

Mathematics Education in the Digital Era

Gila Hanna

David A. Reid

Michael de Villiers *Editors*

Proof Technology in Mathematics Research and Teaching



Springer

Mathematics Education in the Digital Era

Volume 14

Series Editors

Dragana Martinovic, University of Windsor, Windsor, ON, Canada
Viktor Freiman, Faculté des sciences de l'éducation, Université de Moncton,
Moncton, NB, Canada

Editorial Board

Marcelo Borba, State University of São Paulo, São Paulo, Brazil
Rosa Maria Bottino, CNR – Istituto Tecnologie Didattiche, Genova, Italy
Paul Drijvers, Utrecht University, Utrecht, The Netherlands
Celia Hoyles, University of London, London, UK
Zekeriya Karadag, Giresun Üniversitesi, Giresun, Turkey
Stephen Lerman, London South Bank University, London, UK
Richard Lesh, Indiana University, Bloomington, USA
Allen Leung, Hong Kong Baptist University, Kowloon Tong, Hong Kong
Tom Lowrie, University of Canberra, Bruce, Australia
John Mason, Open University, Buckinghamshire, UK
Sergey Pozdnyakov, Saint-Petersburg State Electro Technical University,
Saint-Petersburg, Russia
Ornella Robutti, Università di Torino, Torino, Italy
Anna Sfard, USA & University of Haifa, Michigan State University, Haifa, Israel
Bharath Sriraman, University of Montana, Missoula, USA
Anne Watson, University of Oxford, Oxford, UK
Eleonora Faggiano, Department of Mathematics, University of Bari Aldo Moro,
Bari, Bari, Italy

The Mathematics Education in the Digital Era (MEDE) series explores ways in which digital technologies support mathematics teaching and the learning of Net Gen'ers, paying attention also to educational debates. Each volume will address one specific issue in mathematics education (e.g., visual mathematics and cyber-learning; inclusive and community based e-learning; teaching in the digital era), in an attempt to explore fundamental assumptions about teaching and learning mathematics in the presence of digital technologies. This series aims to attract diverse readers including: researchers in mathematics education, mathematicians, cognitive scientists and computer scientists, graduate students in education, policy-makers, educational software developers, administrators and teachers-practitioners. Among other things, the high quality scientific work published in this series will address questions related to the suitability of pedagogies and digital technologies for new generations of mathematics students. The series will also provide readers with deeper insight into how innovative teaching and assessment practices emerge, make their way into the classroom, and shape the learning of young students who have grown up with technology. The series will also look at how to bridge theory and practice to enhance the different learning styles of today's students and turn their motivation and natural interest in technology into an additional support for meaningful mathematics learning. The series provides the opportunity for the dissemination of findings that address the effects of digital technologies on learning outcomes and their integration into effective teaching practices; the potential of mathematics educational software for the transformation of instruction and curricula; and the power of the e-learning of mathematics, as inclusive and community-based, yet personalized and hands-on.

Submit your proposal:

Book proposals for this series may be submitted per email to Springer or the Series Editors. - Springer: Natalie Rieborn at Natalie.Rieborn@springer.com - Series Editors: Dragana Martinovic at dragana@uwindsor.ca and Viktor Freiman at viktor.freiman@umoncton.ca

Forthcoming volumes

Teaching Mathematics and Financial Education - Research and Practice
Edited by Annie Savard and Alexandre Soares Cavalcante

More information about this series at <http://www.springer.com/series/10170>

Gila Hanna · David A. Reid ·
Michael de Villiers
Editors

Proof Technology in Mathematics Research and Teaching

 Springer

Editors

Gila Hanna
Ontario Institute for Studies in Education
University of Toronto
Toronto, ON, Canada

David A. Reid
AG Didaktik, Fachbereich 3
Universität Bremen
Bremen, Germany

Michael de Villiers
Faculty of Education
Stellenbosch University
Stellenbosch, South Africa

ISSN 2211-8136 ISSN 2211-8144 (electronic)
Mathematics Education in the Digital Era
ISBN 978-3-030-28482-4 ISBN 978-3-030-28483-1 (eBook)
<https://doi.org/10.1007/978-3-030-28483-1>

© Springer Nature Switzerland AG 2019

This work is subject to copyright. All rights are reserved by the Publisher, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, reuse of illustrations, recitation, broadcasting, reproduction on microfilms or in any other physical way, and transmission or information storage and retrieval, electronic adaptation, computer software, or by similar or dissimilar methodology now known or hereafter developed.

The use of general descriptive names, registered names, trademarks, service marks, etc. in this publication does not imply, even in the absence of a specific statement, that such names are exempt from the relevant protective laws and regulations and therefore free for general use.

The publisher, the authors and the editors are safe to assume that the advice and information in this book are believed to be true and accurate at the date of publication. Neither the publisher nor the authors or the editors give a warranty, expressed or implied, with respect to the material contained herein or for any errors or omissions that may have been made. The publisher remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

This Springer imprint is published by the registered company Springer Nature Switzerland AG
The registered company address is: Gewerbestrasse 11, 6330 Cham, Switzerland

Contents

Introduction

Proof Technology: Implications for Teaching	3
--	---

Gila Hanna, David A. Reid and Michael de Villiers

Automatic Theorem Provers

A Fully Automatic Theorem Prover with Human-Style Output	13
---	----

M. Ganesalingam and W. T. Gowers

A Common Type of Rigorous Proof that Resists Hilbert’s Programme	59
---	----

Alan Bundy and Mateja Jamnik

SMTCoq: Mixing Automatic and Interactive Proof Technologies	73
--	----

Chantal Keller

Studying Algebraic Structures Using Prover9 and Mace4	91
--	----

Rob Arthan and Paulo Oliva

Theoretical Perspectives on Computer-Assisted Proving

Didactical Issues at the Interface of Mathematics and Computer Science	115
---	-----

Viviane Durand-Guerrier, Antoine Meyer and Simon Modeste

Issues and Challenges in Instrumental Proof	139
--	-----

Philippe R. Richard, Fabienne Venant and Michel Gagnon

The Contribution of Information and Communication Technology to the Teaching of Proof.	173
---	-----

Maria Alessandra Mariotti

Journeys in Mathematical Landscapes: Genius or Craft?	197
Lorenzo Lane, Ursula Martin, Dave Murray-Rust, Alison Pease and Fenner Tanswell	
Suggestions for the Use of Proof Software in the Classroom	
Using Automated Reasoning Tools to Explore Geometric Statements and Conjectures	215
Markus Hohenwarter, Zoltán Kovács and Tomás Recio	
Computer-Generated Geometry Proofs in a Learning Context	237
Pedro Quaresma and Vanda Santos	
Using 3D Geometry Systems to Find Theorems of Billiard Trajectories in Polyhedra	255
Heinz Schumann	
Classroom Experience with Proof Software	
Learning Logic and Proof with an Interactive Theorem Prover	277
Jeremy Avigad	
Web-Based Task Design Supporting Students' Construction of Alternative Proofs	291
Mikio Miyazaki, Taro Fujita and Keith Jones	
Reasoning by Equivalence: The Potential Contribution of an Automatic Proof Checker	313
Christopher Sangwin	
Virtual Manipulatives and Students' Counterexamples During Proving	331
Kotaro Komatsu and Keith Jones	
Afterword	
Proof Technology and Learning in Mathematics: Common Issues and Perspectives	349
Nicolas Balacheff and Thierry Boy de la Tour	
Author Index	367
Subject Index	375

Contributors

Rob Arthan School of Electronic Engineering and Computer Science, Queen Mary University of London, London, UK

Jeremy Avigad Department of Philosophy, Carnegie Mellon University, Pittsburgh, PA, USA

Nicolas Balacheff Univ. Grenoble Alpes, CNRS, Grenoble INP, LIG, Grenoble, France

Thierry Boy de la Tour Univ. Grenoble Alpes, CNRS, Grenoble INP, LIG, Grenoble, France

Alan Bundy University of Edinburgh, Edinburgh, UK

Michael de Villiers Stellenbosch University, Stellenbosch, South Africa

Viviane Durand-Guerrier IMAG, Univ Montpellier, CNRS, Montpellier, France

Taro Fujita University of Exeter, Exeter, UK

Michel Gagnon École Polytechnique de Montréal, Montréal, Canada

M. Ganesalingam Trinity College, Cambridge, UK

W. T. Gowers Department of Pure Mathematics and Mathematical Statistics, University of Cambridge, Cambridge, UK

Gila Hanna Ontario Institute for Studies in Education, University of Toronto, Toronto, Canada

Markus Hohenwarter Johannes Kepler University of Linz, Linz, Austria

Mateja Jamnik University of Cambridge, Cambridge, UK

Keith Jones School of Education, University of Southampton, Southampton, UK

Chantal Keller LRI, University of Paris-Sud, CNRS UMR 8623, Université Paris-Saclay, Orsay Cedex, France

- Kotaro Komatsu** Institute of Education, Shinshu University, Nagano, Japan
- Zoltán Kovács** The Private University College of Education of the Diocese of Linz, Linz, Austria
- Lorenzo Lane** University of Oxford, Oxford, UK
- Maria Alessandra Mariotti** University of Siena, Siena, Italy
- Ursula Martin** University of Oxford, Oxford, UK
- Antoine Meyer** LIGM (UMR 8049), UPEM, CNRS, ESIEE, ENPC, Université Paris-Est, Marne-la-Vallée, France
- Mikio Miyazaki** Shinshu University, Nagano, Japan
- Simon Modeste** IMAG, Univ Montpellier, CNRS, Montpellier, France
- Dave Murray-Rust** University of Edinburgh, Edinburgh, Scotland
- Paulo Oliva** School of Electronic Engineering and Computer Science, Queen Mary University of London, London, UK
- Alison Pease** University of Dundee, Dundee, Scotland
- Pedro Quaresma** University of Coimbra, Coimbra, Portugal
- Tomás Recio** University of Cantabria, Santander, Spain
- David A. Reid** Universität Bremen, Bremen, Germany
- Philippe R. Richard** Université de Montréal, Montréal, Canada
- Christopher Sangwin** School of Mathematics, University of Edinburgh, Edinburgh, UK
- Vanda Santos** University of Coimbra, Coimbra, Portugal
- Heinz Schumann** University of Education Weingarten, Weingarten, Germany
- Fenner Tanswell** University of St Andrews, St Andrews, Scotland
- Fabienne Venant** Université du Québec à Montréal, Montréal, Canada

Introduction

Proof Technology: Implications for Teaching



Gila Hanna, David A. Reid and Michael de Villiers

Proving is sometimes thought to be the aspect of mathematical activity most resistant to the influence of technological change. While computational methods are well known to have a huge importance in applied mathematics, there is a perception that mathematicians seeking to derive new results are unaffected by the digital era. The reality is quite different. Digital technologies are influencing the way mathematicians work together and the way they go about proving.

Checking billions of cases in extremely large but finite sets, impossible a few decades ago, has now become a standard method of proof. Distributed proving, by teams of mathematicians working independently on sections of a problem, has become very much easier as digital communication facilitates the sharing and comparison of results. Proof assistants and dynamic proof environments have influenced the verification or refutation of conjectures. Techniques from computer science for checking the validity of programs are being used to verify mathematical proofs. In fact, proof-checking by software is seen as a necessity by the mathematician Vladimir Voevodsky (1966–2017), who created groundbreaking tools for computer confirmation of the accuracy of proofs, given that human readers have often failed to detect errors in many important proofs, including his own. It has been his dream to have an electronic proof assistant check any theorem before its publication, as a way to accelerate the reviewing process.

G. Hanna (✉)

Ontario Institute for Studies in Education, University of Toronto, Toronto, Canada
e-mail: gila.hanna@utoronto.ca

D. A. Reid

Universität Bremen, Bremen, Germany
e-mail: dreid@uni-bremen.de

M. de Villiers

Stellenbosch University, Stellenbosch, South Africa
e-mail: profmd1@mweb.co.za

© Springer Nature Switzerland AG 2019

G. Hanna et al. (eds.), *Proof Technology in Mathematics Research and Teaching*, Mathematics Education in the Digital Era 14,
https://doi.org/10.1007/978-3-030-28483-1_1

Mathematics education is faced with many challenges. Educators are expected to equip students with the ability to succeed in a rapidly changing world, to learn methods of reasoning and solving problems that are constantly reshaped by technological advances. These are sometimes called ‘21st century skills’: mental and social flexibility, creativity, collaboration, etc. Technological tools like theorem provers can play a significant role in the development of these skills.

Yet mathematics education has been slow to reflect the substantial and increasing use of proof assistants by practicing mathematicians, though many of the tools are readily available. The goal of this book is to begin a dialogue between mathematics educators and researchers actively developing automatic theorem provers and related tools. This book cannot hope to bridge the gap between these two communities completely, but we hope it will strengthen the awareness of educators to the potential of these powerful tools. It is intended for mathematics educators with an interest in computer tools that support proving, developers of automated theorem provers who are interested in the educational implications of their work, and new scholars in both these fields who will find food for thought and useful starting places for further research. It provides a sampler of current topics in the use of automated reasoning tools. Along with theoretical aspects of computer-assisted proving relevant to the teaching of proof, the book’s chapters explore the most recent developments in the use of technology in proving, notably interactive theorem provers and proof checkers, and go on to provide examples of their potential application in the classroom. Some chapters also present case studies of classroom experience with dynamic geometry software.

The book is organized into an unavoidably rough grouping of four broad parts: (1) Automatic theorem provers, (2) Theoretical perspectives on computer-assisted proving, (3) Suggestions for the use of proof software in the classroom, and (4) Classroom experience with proof software. A separate last chapter provides a discussion of relevant issues and additional topics.

1 Part I: Automatic Theorem Provers

In “A fully automatic theorem prover with human-style output,” M. Ganesalingam and W. T. Gowers show that automated provers do provide a much welcome help to mathematicians, are mathematician-friendly, and can successfully assist mathematicians in proving results. They first describe the state of the art in automated proofs and then explain their own program, which is capable of outputting a reader-friendly script. By using this program students can try out various ways of proving and ask the system to let them know if they are on the right track. The authors also illustrate their program with a few examples. As their abstract indicates, this program was able to solve elementary mathematical problems and present “solutions that are hard to distinguish from solutions that might be written by human mathematicians.” This fact alone is so compelling that it is hard to imagine a mathematics educator, having read the chapter, who would not be impelled to explore the potential of this

program for mathematics education (This chapter appeared in the *Journal of Automated Reasoning* February 2017, 58, 2, 253–291. It is reproduced by permission from Springer).

In their chapter “A common type of rigorous proof that resists Hilbert’s programme” Alan Bundy and Mateja Jamnik show that there is a class of rigorous proofs, which they call “schematic”, which are commonly used in mathematics education but which resist formalization in Hilbert’s sense. This challenges the assumption, widespread in mathematics education, that a rigorous proof is in principle translatable into a formal one, that is, a sequence of formulae each of which is either an axiom or follows from earlier formulae by a rule of inference. Bundy and Jamnik provide a surprising insight by arguing that to be formalizable, schematic proofs require a program more radical than Hilbert’s. They support their claim with convincing evidence, through a detailed analysis of schematic proofs widely discussed in mathematics education, namely Cauchy’s Proof of Euler’s Theorem as discussed by Lakatos (1976), and a few number-theoretic diagrammatic proofs also known as “proofs without words”.

An automatic theorem prover, such as that described by Ganesalingam and Gowers in their chapter, can find proofs autonomously, but coding inaccuracies can lead to errors. Interactive provers, in contrast, are designed to assist mathematicians by checking the proofs they produce. Chantal Keller, in her chapter “SMTCoq: Mixing automatic and interactive proof technologies”, describes the use of an interactive prover to check the output of an automatic theorem prover. SMTCoq allows the Coq interactive theorem prover to work with automatic theorem provers on large combinatorial problems, as well as other areas of mathematics and computer science. She provides examples of how SMTCoq works and outlines its modular and efficient architecture. A key feature is that the automatic theorem provers output the arguments underlying their results, which allows reconstruction and checking by the interactive theorem prover. It is interesting to imagine how such systems could evolve to the point that students could outline a proof approach, use an automatic prover to handle the tedious parts while constructing the interesting parts themselves, and have the whole checked by an interactive prover.

Rob Arthan and Paulo Oliva, in their chapter “Studying algebraic structures using Prover9 and Mace4”, present examples from Heyting algebras of the use of a fully automated theorem prover as well as an automatic counterexample generator. They illustrate how quick and effective Prover9 and Mace4 are in assisting an investigation. The counterexample generator helps to quickly rule out false conjectures, while the theorem prover is sometimes more effective than a human being in verifying the relevant algebraic identities in question. That some of these computer-generated proofs can seem inaccessible to a human reader may provide an ‘intellectual challenge’ for mathematicians: to unpack them and try to rewrite them in an easier, understandable form. Lastly, some brief suggestions are also made about the possible use of such tools in the teaching of algebra.

2 Part II: Theoretical Perspectives on Computer-Assisted Proving

The chapter “Didactical issues at the interface of mathematics and computer science” by Viviane Durand-Guerrier, Antoine Meyer, and Simon Modeste discusses research on the interaction between the two subjects, and reflects on the epistemological and didactical issues that surface. The authors identify proof as a central didactical issue at both high school and undergraduate levels. To illustrate their arguments, they examine and illustrate the links between the concepts of algorithm, proof and computer program.

The notion of instrumental proof is defined and analyzed in the chapter “Issues and challenges in instrumental proof” by Philippe Richard, Fabienne Venant, and Michel Gagnon. Using the theory of mathematical workspaces, they investigate the epistemology and semiotics of instrumental proofs in discursive-graphic, mechanical and algorithmical contexts. Some reflections are also given on the necessary learning conditions for classroom use of computer devices to achieve a healthy balance between informal heuristics and formal validation.

Maria Alessandra Mariotti, in her chapter “The contribution of information and communication technology to the teaching of proof” outlines ways in which computers can allow students not only to prove theorems that would be beyond them without assistance, but also to learn about the *nature* of proof. She cites examples of teaching experiments exploring this possibility, but, more importantly, she outlines a theoretical framework, within the Theory of Semiotic Mediation, that informs and justifies teaching approaches. One focus is the importance of understanding not only that particular results are true but also the context of definitions, principles and ways of reasoning, the ‘theory’, in which they are true. Mariotti sees great potential for the use of computers to help students become aware of the theories in which they are proving results, including the criteria for proofs’ validity. It is interesting to consider the uses of computers in proving outlined in other chapters in the light of this chapter’s ‘meta’ perspectives.

The chapter “Journeys in mathematical landscapes: genius or craft?” by Lorenzo Lane, Ursula Martin, Dave Murray-Rust, Alison Pease, and Fenner Tanswell, focuses on how anglophone mathematicians view their professional activity. The authors examine the idea that the production of mathematics is akin to the exploration of unknown landscapes. They provide a number of quotations in support of the notion that one need not possess a “magic ingredient” to be able to do mathematics. Rather, mathematics as craft is mastered through apprenticeship, practice, and skill-development with the goal of understanding the relationships of individual elements in a proof or a problem, and making the distinction between knowing-how and knowing-that. They also discuss an example of an online mathematical activity, known as “polymath” in which people collaborate over the internet on the solution of problems and the construction of proofs.

3 Part III: Suggestions for the Use of Proof Software in the Classroom

In their chapter “Using automated reasoning tools to explore geometric statements and conjectures” Markus Hohenwarter, Zoltán Kovács, and Tomás Recio describe the integration of an automatic theorem prover into GeoGebra, a dynamic geometry software often used in schools. This allows students to be certain that a proof *exists* for a conjectured relation in a geometric situation. In the past, students could use dynamic geometry software to amass empirical evidence for a conjecture by dragging, but the tool described here is a true theorem-prover. It can also, if a conjecture is not true in general, suggest changes to the constructions that would make it true, thus facilitating the identification of necessary conditions. The authors describe a hypothetical use of the tool in a classroom, in which “the machine behaves like a mentor in the learning process, helping students in the intermediate steps of developing their own explanations of the truth of some geometric facts.”

Pedro Quaresma and Vanda Santos trace the evolution of dynamic geometry systems (DGSs) and geometry automated theorem provers (GATPs) in their chapter “Computer-generated geometry proofs in a learning context”. These tools are now being used by geometry practitioners (and students) not only to explore and test new results and conjectures, but also to generate proofs with natural language and visual rendering. Examples illustrate the use of the GATP features of *GeoGebra* and *Java Geometry Expert* to, respectively, validate the well-known midpoint triangle theorem, and to establish that the two lines are parallel in a particular configuration, by basically giving the output ‘true’, but nothing else. Such validations clearly only provide verification—they do not, for example address mathematicians’ needs for understanding, explanation, clarification and systematization. However, the authors proceed to give several examples of newer software developments that are now starting to address these issues by giving outputs that are more human-readable, in the form of natural language, logic and visual formats.

In his chapter “Using 3D geometry systems to find theorems of billiard trajectories in polyhedra” Heinz Schumann uses the dynamic geometry system Cabri 3D to explore the trajectories of billiards in the cube and other convex polyhedra. Mathematically, the investigation involves shortest paths and is equivalent to the discovery of polygons with minimal perimeter inscribed within polyhedra. As illustrated with many examples, the software provides opportunities for students to dynamically explore, and easily make, many conjectures. The ability of the software to vary parameters of geometric figures by dragging corresponding points assists 3D visualization and heuristic thinking, as well as the gaining of confidence in the validity of observations. A link is also provided to downloadable sketches that can be used by the reader with a free Demo version of Cabri 3D.

4 Part IV: Classroom Experience with Proof Software

Jeremy Avigad's chapter "Learning logic and proof with an interactive theorem prover" advocates the virtues of learning both logic and proof through the use of an interactive theorem prover. Avigad describes an undergraduate course specifically designed to teach students to construct ordinary mathematical proofs using *Lean*, an interactive theorem prover, providing sufficient details on both the contents of the course and the way the course was taught. The chapter maintains that students learn three notations simultaneously: informal mathematical language, formal symbolic logic, and the language of *Lean*, each contributing unique insights, that seem to be helpful to students. Avigad also argues that *Lean* was found to be particularly effective in helping students internalize both the syntax and the rules of logic. He also reveals that students rated the course highly in their evaluations, but grants that a more rigorous evaluation is needed to support the claim that the use of interactive theorem provers is indeed effective in teaching mathematics in general and proof in particular.

The chapter by Mikio Miyazaki, Taro Fujita, and Keith Jones explores the potential of a "Web-based task design supporting students' construction of alternative proofs" especially multiple proofs for a given geometrical statement. The tool provides feedback for different kinds of errors students are likely to make, and a flow-chart visualization of the structure of the proof. The authors describe experience using this tool in classroom contexts in which students could express quite sophisticated thinking about how one proof can be transformed into another, logically distinct of the same statement. The students showed awareness of logical principles (e.g., avoiding circular reasoning) and of the logical connections between statements (e.g., identifying different theorems that can function in equivalent ways in a proof). They also developed some insight into 'aesthetic' aspects of proving, for example seeking proofs that maximize efficiency.

In his chapter "Reasoning by equivalence: the potential contribution of an automatic proof-checker," Christopher Sangwin argues that reasoning by equivalence "is the most important single form of reasoning in school mathematics". He then focuses on defining what he means by reasoning by equivalence, on the extent to which students are asked to provide proofs, and on the design of an automatic proof-checker within the STACK software which he created and which he uses to assess students' responses. One characteristic element of the software is its focus on a whole argument which can be submitted to formal verification; another is the important distinction between reasoning and argumentation. The chapter underscores the fact that through carefully designed tools, it is possible to communicate to students what is and is not important in writing a satisfactory mathematical proof. The chapter gives several examples of student work using STACK and ends with a thorough discussion of the benefits and drawbacks on focusing on equivalence.

The chapter by Kotaro Komatsu and Keith Jones discusses "Virtual manipulatives and students' counterexamples during proving" in an undergraduate geometry course. The chapter is a useful reminder that instruction with technology must always

be planned judiciously. Careful consideration of the goals of teaching and the affordances of the technology are needed, as well as awareness of the teacher's role in the lesson. The authors describe how they used task-design principles developed for virtual manipulatives to reanalyze a dynamic geometry task they had developed using different principles, and, drawing on an example of students' work with that task, they show the relevance of the design principles.

The last chapter, by Nicolas Balacheff and Thierry Boy De La Tour, is an Afterword.

5 Conclusion

We have indicated only a few ways in which the chapters can be used. They can of course be read in a more selective way, highlighting the fruitful connection between new proof technologies and teaching. The chapters also show that the connection between proof technology and mathematics education is a wide field inviting further investigation. We already have some sense that proof assistants greatly diminish the need for verification and justification, but we know almost nothing of their potential contribution to other roles of proof, such as explanation, communication, discovery, and systematization, or how they now may become more relevant as pedagogical motivation for the learning of proof in the classroom.

We hope that this volume will be part of a broader discussion on the value of proof technology in both mathematics and mathematics education. We also hope it will stimulate mathematics educators to learn more about proof technology and to find innovative and effective classroom applications.

Acknowledgements We would like to thank all the authors for contributing their time and expertise to this book. We wish to acknowledge the referees for their thoughtful and constructive reviews. Many authors also served as referees; their double task is highly appreciated.

Special thanks go to Arleen Schenke, Hardy Grant, and Ed Barbeau for their stylistic polishing of some of the chapters and for their most helpful editorial advice.

Thanks are due to the *Journal of Automated Reasoning* for permission to reproduce the article "A Fully Automatic Theorem Prover with Human-Style Output" by M. Ganesalingam and W. T. Gowers.

We wish to acknowledge the generous support of the Social Sciences and Humanities Research Council of Canada.

Automatic Theorem Provers

A Fully Automatic Theorem Prover with Human-Style Output



M. Ganesalingam and W. T. Gowers

1 Introduction

1.1 Overview of the Paper

The main purpose of this paper is to describe a program that solves elementary mathematical problems, mostly but not exclusively in metric space theory, and that presents the solutions in a form that is hard to distinguish from solutions that human mathematicians might write. The following two proofs are examples of the program's output.¹ The first is a proof that if $f : X \rightarrow Y$ is a continuous function and U is an open subset of Y , then $f^{-1}(U)$ is an open subset of X , and the second is a proof that if $f : X \rightarrow Y$ is an injection and A and B are subsets of X , then $f(A) \cap f(B) \subset f(A \cap B)$.

Let x be an element of $f^{-1}(U)$. Then $f(x) \in U$. Therefore, since U is open, there exists $\eta > 0$ such that $u \in U$ whenever $d(f(x), u) < \eta$. We would like to find $\delta > 0$ s.t. $y \in f^{-1}(U)$ whenever $d(x, y) < \delta$. But $y \in f^{-1}(U)$ if and only if $f(y) \in U$. We know that $f(y) \in U$ whenever $d(f(x), f(y)) < \eta$. Since f is continuous, there exists $\theta > 0$ such that $d(f(x), f(y)) < \eta$ whenever $d(x, y) < \theta$. Therefore, setting $\delta = \theta$, we are done.

Let x be an element of $f(A) \cap f(B)$. Then $x \in f(A)$ and $x \in f(B)$. That is, there exists $y \in A$ such that $f(y) = x$ and there exists $z \in B$ such that $f(z) = x$. Since f is an injection,

¹The program produces LaTeX output, which we reproduce verbatim here.

Reprinted with permission from Springer's Journal of Automated Reasoning, February 2017, Volume 58(2), pp 253–291.

M. Ganesalingam (✉)
Trinity College, Cambridge, UK
e-mail: mg262@cam.ac.uk

W. Gowers
Department of Pure Mathematics and Mathematical Statistics, University of Cambridge,
Wilberforce Road, Cambridge CB3 0WB, UK

$f(y) = x$ and $f(z) = x$, we have that $y = z$. We would like to find $u \in A \cap B$ s.t. $f(u) = x$.
But $u \in A \cap B$ if and only if $u \in A$ and $u \in B$. Therefore, setting $u = y$, we are done.

The main challenge we have faced in creating this program, discussed more extensively below, is that it does not seem to be possible to reconstruct genuinely human-like writeups from the proofs produced by automated provers from the machine-oriented tradition. In order to be able to produce such writeups we have had to use much more restricted proof methods than those available to modern provers. This in turn makes it more challenging for the prover to solve any individual problem, and indeed the program does not solve any problems that are beyond the reach of existing fully automatic provers. We should also note that we see this prover as the first stage of a long-term project to write a program that solves more complex problems in a ‘fully human’ way. We shall say a little about this wider project later in the paper.

For the remainder of this section, we shall discuss the human-style output further. We shall explain why we think it is a goal worth pursuing, and expand on the difficulties just mentioned. In Sect. 2 we shall discuss related work. In Sect. 3 we outline the main features of the prover. In Sect. 4 we describe the construction of the human-style writeup. In Sect. 5 we provide more extensive technical details about how the prover works, and in Sect. 6 we present a detailed example to illustrate how these technical elements operate in practice. In Sect. 7, we describe an informal experiment that we carried out in order to test whether the proofs produced by the program were indeed similar to the proofs that a human might write, and in Sect. 8 we provide some practical notes on running the program. Finally, in Sect. 9, we discuss possible future work.

1.2 *Why Bother with Human-Style Output?*

These days there is a thriving subcommunity of mathematicians who use interactive theorem provers such as Coq, HOL, Isabelle and Mizar. However, it is also noticeable that the great majority of mathematicians do not use these systems and seem unconcerned about the possibility that their arguments are incorrect, even though there is plenty of evidence that incorrectness pervades the published literature.

There are several reasons for this, some of which are discussed by Bundy (2011). One is that the mathematical community has simply learnt to deal with the current state of affairs. Mistakes in the literature are less damaging than one might think, because either they are in papers that nobody wishes to build on in future work, or they are in papers that are important and tend therefore to be thoroughly scrutinized. Occasionally a paper by a reputable author makes an important claim that is very hard to verify because the paper is extremely complicated or badly written. In such situations, the paper will usually be treated with caution until some plausible confirmation of its correctness has been provided.

Another reason is that learning to use an interactive theorem prover, though not impossibly difficult, still requires an investment of time that most mathematicians

are not prepared to make, when the reward is likely to be simply to tell them, after substantial effort, that a result is correct that they are already confident is correct. Of course, as just commented, sometimes mathematicians are *not* 100% confident that a result is correct, and in such situations interactive theorem provers have an extremely valuable part to play: among their recent triumphs are verification of the four-colour theorem (Gonthier, 2019) and the Kepler conjecture (Hales et al., 2015). Another remarkable achievement was the formalized proof of the odd order theorem of Feit and Thompson (Gonthier et al., 2013): in that case the correctness of the theorem was not in serious doubt, but the paper of Feit and Thompson was very long and complicated and therefore difficult to check, so there is genuine value in having a formalized proof. This will be especially true if, as the authors hope, it leads to formalizations of other long and complicated proofs, of which there are an ever-increasing number. However, these notable examples are the exception rather than the rule, and the fact remains that most mathematicians do not feel that the benefits of automatic theorem provers are worth the effort when it comes to ‘normal’ mathematics.

So what *would* be beneficial to ‘typical’ mathematicians? One possible answer is to place far less emphasis on proofs as certifications of correctness and far more on proofs as *explanations*. Many mathematicians say that their aim, when looking for proofs, is to achieve understanding rather than to feel confident that a statement is true. Indeed, there are several sociological facts that cannot otherwise be explained. For example, why do mathematicians try to find proofs of results when a combination of experimental evidence and heuristic arguments establishes their truth beyond all reasonable doubt? (Goldbach’s conjecture is a statement that falls into this category). And why are new and different proofs of already known theorems sometimes highly valued? Why do mathematicians have a clear sense that some proofs are ‘better’ than others? It is clear that any answer to these questions, which have attracted considerable attention in the philosophical literature (see for example Mancosu (2008) and the references therein), has to take into account far more than the mere correctness of a proof.

Therefore, for an automatic theorem prover to be useful to mainstream mathematicians, it will be highly desirable for it to produce ‘good’ proofs that ‘explain what is going on’. Achieving this is of course a major challenge, but focusing on how humans find proofs is a good way of trying to meet that challenge, since there seems to be a close connection between the degree to which a proof is judged to be ‘explanatory’ and the ease with which one can see how somebody (a human, that is) might have thought of it. So an automatic theorem prover that imitates the way humans do mathematics is more likely to produce proofs that are appealing to human mathematicians.

For a proof to appeal to human mathematicians, a minimum requirement is that it should be written in a way that is similar to the way humans write. To design a system that produces human-style output one has a choice. One possibility is for the system to operate in a way that closely mirrors the way human mathematicians operate, in which case it is fairly straightforward to convert each step of its reasoning process into a piece of human-style prose, which will form the basis for the human-style

output. The other possibility is for the system to operate in a quite different way from humans, but then to do some highly nontrivial processing to convert its initial logical output into a text that reads like what a human would write. The first of these options looks much easier, but it forces one to pay close attention to many details that would otherwise not matter. This gives rise to many interesting questions that can be regarded as easier cases of some of the fundamental questions in automatic theorem proving.

One such question is an obvious one that many people have asked: how do human mathematicians manage to avoid a combinatorial explosion when they search for solutions to complex problems? While we do not claim to have a satisfactory answer to this, we take the view (which is by no means universally shared, so this is a somewhat speculative claim) that a good way to look for the answer is to focus on exactly the kinds of small technical difficulties that have arisen in our work. In order to design a program that imitates human thought, we have had to place severe restrictions on how it operates; our hope is that if we can get these restrictions right, then we will have the beginnings of a program that can be scaled up to solve more complex problems with only a very modest amount of searching.

1.3 Constraints and Challenges

As noted above, the main challenge we have faced is that we have not found it to be possible to ‘bolt on’ genuinely human-like output to an existing automated theorem prover. This seriously constrains the kind of methods that our prover can use. For example, a simple example of a procedure that is used by many theorem provers, especially those based on resolution, but that would be completely out of the question for us is putting all goals into conjunctive normal form. Not only do human mathematicians not do this, but the resulting search for a proof is quite unlike the kind of search that human mathematicians undertake, to the point where converting a successful search into a human-style write-up would be scarcely any easier than obtaining the write-up had been before the program started. The difficulty faced here is essentially the same difficulty found in decompilation of compiled code. In each case the high-level description (a human-style mathematical proof, a human-readable program) can always be converted into a low-level equivalent (a logical derivation, machine code). The translation is formally lossless, but at the same time some kind of higher-level structure that is important to humans is ‘lost in translation’ so that reversing the process, that is, converting machine code to a high-level program or a logical derivation to a human-style proof, is extremely difficult.

A less obvious example is the use of modus ponens. One might expect this to be fundamental to any theorem prover, but in fact pure modus ponens is almost never used by human mathematicians. Although generations of mathematicians are told that $P \implies Q$ means the same as $\neg P \vee Q$, this does not in fact reflect the way mathematicians think or write (which causes genuine confusion to many first-year undergraduates). The implications that mathematicians actually use are almost

always quantified ones—that is, implications of the form $\forall x (P(x) \implies Q(x))$ —although the quantification may not be explicitly mentioned.

Suppose, for example, that a human mathematician wishes to show that $3^{996} \equiv 1 \pmod{997}$. A typical proof might read as follows: “Fermat’s little theorem states that if p is a prime and a is not a multiple of p , then $a^{p-1} \equiv 1 \pmod{p}$. Since 997 is prime, and 3 is not a multiple of 997, it follows that $3^{996} \equiv 1$.” A theorem prover that performed the deduction as follows would normally be regarded as acceptable:

1. $\forall a, p \in \mathbb{N} (p \text{ is prime} \wedge p \nmid a \implies a^{p-1} \equiv 1 \pmod{p})$.
2. 997 is prime.
3. $997 \nmid 3$.
4. $997 \text{ is prime} \wedge 997 \nmid 3 \implies 3^{997-1} \equiv 1 \pmod{997}$.
5. $3^{997-1} \equiv 1 \pmod{997}$.
6. $997 - 1 = 996$.
7. $3^{996} \equiv 1 \pmod{997}$.

In the above chain of reasoning, statements 1–3 are given as initial assumptions, 4 is obtained from 1 by universal instantiation, 5 is obtained from 2 to 4 by (unquantified) modus ponens, and 7 is obtained from 5 and 6 by equality substitution.

If we were to convert this reasoning directly into prose, it might end up something like this.

For every $a, p \in \mathbb{N}$, if p is prime and a is not a multiple of p , then $a^{p-1} \equiv 1 \pmod{p}$. Therefore, if 997 is prime and 3 is not a multiple of 997, then $3^{997-1} \equiv 1 \pmod{997}$. But 997 is prime and 3 is not a multiple of 997. Therefore, $3^{997-1} \equiv 1 \pmod{997}$. But $997 - 1 = 996$. Therefore, $3^{996} \equiv 1 \pmod{997}$.

This is noticeably different from what a human would write, because of the second sentence, which corresponds to step 4. A human mathematician will deduce 5 from 1 to 3 using quantified modus ponens, so if we want to produce realistic human-style write-ups, we either have to emulate this or suppress the universal instantiation step when it comes to the write-up. But the latter option produces inappropriate proofs in other cases: for example, if steps occur between the universal instantiation and modus ponens, then the writeup can end up implicitly using a fact which has not been mentioned for some time, which contravenes rules regarding the coherence of discourse and so generates an unnatural writeup.

Another striking and important feature of human mathematical reasoning, which is reflected in some but not all theorem provers, is that it tends to take place at as high a level as possible, at least when it is done well. For example, an inexperienced undergraduate, when asked to prove that the graph of the function $f(x) = x^2$ is a closed subset of \mathbb{R}^2 , might well choose a point (x, y) in the complement of the graph and laboriously find $\epsilon > 0$ such that no point (z, w) within ϵ of (x, y) belonged to the graph. A more experienced mathematician would give a high-level proof such as this.

Let $g(x, y) = x^2 - y$ for each $x, y \in \mathbb{R}$. Then g is a continuous function and the graph of f is the inverse image $g^{-1}(\{0\})$. Since the singleton $\{0\}$ is closed, so is $g^{-1}(\{0\})$.

This short argument illustrates several of the kinds of phenomena that concern us. The main one we wish to highlight is that it does not make any use of the definition of ‘is closed’ beyond the fact that in order to apply a general result about inverse images of closed sets one needs a match for the hypothesis ‘ A is closed’. In particular, at no point does the argument perform a definition expansion of the statement ‘ $\{0\}$ is closed’.

Another feature of the argument is that although it is logically necessary to justify the statement that g is continuous, it will be acceptable to many mathematicians not to do so, since the continuity of g is ‘more basic’ than the statement being proved (Similarly, if one *were* writing out a proof of this easier result, one would normally rely on yet more basic statements such as the continuity of $f(x) = x^2$ and of the coordinate projections from \mathbb{R}^2 to \mathbb{R}). A similar remark applies to the statement that the graph of f is equal to $g^{-1}(\{0\})$ and the statement that $\{0\}$ is closed.

A different phenomenon is illustrated by the last sentence: it uses the result that the inverse image of a closed set under a continuous function is closed without explicitly mentioning it. It really does feel to a mathematician as though if a function g is continuous and a subset A of its range is closed, then those two facts together imply that $g^{-1}(A)$ is closed. This implication is ‘mathematical’ rather than ‘purely logical’. Of course, one can analyse it logically by explicitly stating the theoretical result being used and applying quantified modus ponens, but that is a temptation one needs to resist if the program is to produce human-style write-ups without an undue amount of processing.

There are several phenomena like these, and our main challenge has been to identify and take account of them when writing our program. A subsidiary challenge has been ordering the different tactics that humans use in order of attractiveness; if this is not done correctly then the prover will still find proofs but those proofs will read as ‘odd’, because they follow a line that seems strange to a human.

There is one further constraint that needs some justification, which is that our program does not have any ability to backtrack: if it tries out a sequence of steps and gets stuck, then it simply stops. This might seem to be a flaw with the program, since human mathematicians definitely *do* backtrack. However, we took the decision to concentrate on routine problems, which we define informally as precisely those for which humans do not consciously backtrack, since it seems important to have a program that performs well on routine problems before one tackles the formidable question of just how humans limit their searches when the problems become more difficult. Fortunately, there are many proofs that an experienced mathematician will remember as ‘Just do the obvious thing at each stage and it all works out.’ Our initial aim (which we are still a long way from achieving) was to write a program that would solve all routine problems in the way that humans would solve them, which in particular means without backtracking.

2 Related Work

2.1 Systems with Natural-Language Output

Several systems have been developed that use natural language to a greater or lesser extent. An early example with some similarity to ours is that of Felty and Miller (1987). They start with a proof tree and convert it into a more readable form. Their system can also make significant changes to how a proof is presented. The following is an example of output from their system: it is a proof that there are infinitely many primes. The function f mentioned in the proof can be taken to be the function defined by the formula $f(n) = n! + 1$: then the beginning of the proof is asserting some properties of this function that are subsequently used in the proof (so any other function with those properties would do just as well).

Assume $\forall x(f(x) > x) \wedge \forall x \forall y(\text{div}(x, f(y)) \supset (x > y)) \wedge \forall x(\neg \text{prime}(x) \supset \exists y(\text{prime}(y) \wedge \text{div}(y, x)))$. We have two cases. Case 1: Assume $\neg \text{prime}(f(a))$. By modus ponens, we have $\exists y(\text{prime}(y) \wedge \text{div}(y, f(a)))$. Choose b such that $\text{prime}(b) \wedge \text{div}(b, f(a))$. By modus ponens, we have $(b > a)$. Hence, $(b > a) \wedge \text{prime}(b)$. Thus, $\exists x((x > a) \wedge \text{prime}(x))$. Case 2: Assume $\text{prime}(f(a))$. Hence, $(f(a) > a) \wedge \text{prime}(f(a))$. Thus, $\exists x((x > a) \wedge \text{prime}(x))$. Thus, in either case, we have $\exists x((x > a) \wedge \text{prime}(x))$. Since a was arbitrary, we have $\forall n(\exists x((x > n) \wedge \text{prime}(x)))$.

They describe their mechanism for converting the original tree-structured deductions into readable natural-language text as very simple. It is clear that with some small changes they could have improved the readability. For example, they could have replaced $\text{prime}(x)$ by ‘ x is prime’, $\text{div}(x, y)$ by $x|y$ and the symbols for connectives by English words. However, the result would still have had some slightly odd characteristics—for instance, no human mathematician would bother to write ‘by modus ponens’—that would have betrayed its mechanical origins.

Another program that produced readable text was written by Holland-Minkley, Barzilay, and Constable (1999). Their aim was to create natural-language output from the Nuprl system. This is an interactive system based on tactics that is designed to mimic human reasoning. The output from the Nuprl system is not at all easy for the untrained mathematician to read. However, they could convert it into language that was considerably closer to what a human mathematician might write, as the following sample demonstrates (We have slightly modified what they wrote, replacing pairs of minus signs by the cutoff subtraction symbol $\dot{-}$, which seems to be what was intended).

Theorem: For integers a and b and natural number c , $(a \dot{-} b) \dot{-} c = a \dot{-} (b + c)$. Consider that a and b are integers and c is a natural number. Now, the original expression can be transformed to $\text{imax}(\text{imax}(a - b; 0) - c; 0) = \text{imax}(a - (b + c); 0)$. From the add com lemma, we conclude $\text{imax}(-c + \text{imax}(a + -b; 0); 0) = \text{imax}(a + -b + -c; 0)$. From the imax assoc lemma, the goal becomes $\text{imax}(\text{imax}((a + -b) + -c; 0 + -c); 0) = \text{imax}(a + -b + -c; 0)$. There are 2 possible cases. The case $0 + -c \leq 0$ is trivial. Consider $0 < 0 + -c$. Now, the original expression can be transformed to $\text{imax}((a + -b) + -c; 0 + -c) = \text{imax}(a + -b + -c; 0)$. Equivalently, the original expression can be rewritten as $\text{imax}((a + -b) + -c) = \text{imax}(a + -b + -c; 0)$. This proves the theorem.

In places this looks like the kind of continuous prose that a mathematician would write, though as with Felty and Miller’s system there are a number of telltale signs of the mechanical origins of the text. For instance, the first sentence is not quite grammatical: a human would write, ‘Let a and b be integers and let c be a natural number.’ There is also the trivial point that mathematicians would write ‘max’ rather than ‘imax’ (trivial because it would be very easy to change this). There is also a repetitive quality to the prose that gives it an automatically generated feel: for instance, two sentences open with ‘Now, the original expression can be transformed to’.

There are further differences that highlight a point that will be important to us later. For example, when proving that $A = B$ via a string of intermediate inequalities, mathematicians will normally express the proof in the form $A = A_1 = \dots = A_k = B$. From the write-up above, it is clear that Nuprl prefers to deal with equivalences between statements: a typical step might be to reduce the original goal $A = B$ to the goal $A = A_k$, for instance.

Another difference is that the proof makes use of terms that a human would consider too ugly and unnatural to write. For example, no human mathematician would ever write “The case $0 + -c \leq 0$,” instead writing the condition in the more obvious way as “ $c > 0$ ”.

It is perhaps helpful to distinguish between two kinds of unnaturalness of a program’s output: roughly speaking, unnaturalness in how the program expresses its thoughts, and unnaturalness of the thoughts themselves. Writing ‘imax’ and $a + -b$ are examples of the former, while avoiding equational reasoning and considering expressions with strange terms in them are examples of the latter.

A further example of the latter, which we touched on in the previous section, is use of tiny lemmas. Presumably ‘the imax assoc lemma’ is the statement that the binary operation of taking the maximum of two numbers is associative. This statement belongs to a troublesome class of statements that human mathematicians will normally never state explicitly, even when they first meet the basic definitions. The precise reason for this is unclear, but it appears to be something like that we visualize the maximum of a finite set of numbers as its rightmost element on the number line and can immediately see that that rightmost element will win a knockout competition however the competition is organized.

Another example of a statement in this class is the general principle that if a binary operation \circ is associative, then the expression $x_1 \circ \dots \circ x_k$ is unambiguous (More precisely, however you bracket it, you will get the same result). The associativity assumption itself is just the case $k = 3$, and yet generations of mathematicians happily use the more general principle that brackets are unnecessary without ever stopping to prove it.

A third system that has aims that are in certain respects similar to ours is the Theorema system of Buchberger et al. (2006). However, we ask significantly more of our output than the developers of Theorema: for them it is sufficient that the output should be comprehensible to mathematicians, whereas we ask that it should be hard to distinguish from what a human might write. For example, this is the statement of a lemma in the Theorema language (quoted in the paper referred to).

Lemma (“coprime”, any $[a, b]$ with $[\text{nat}[a] \wedge \text{nat}[b]]$, $2b^2 = a^2 \implies \neg \text{coprime}[a, b]$) A human rendition of this lemma would read more like this.

Lemma Let a and b be natural numbers satisfying the equation $2b^2 = a^2$. Then a and b are not coprime.

In other words, it would be written in continuous prose, albeit prose of a characteristic ‘semi-formal’ kind, which makes it significantly easier to read.²

This leads us to an important general point about all three of the systems discussed. The output from these systems is comprehensible to mathematicians, in the sense that they can decipher it. But it is hard work, enough so to prevent most mathematicians from using these systems (It seems likely that one becomes used to reading proofs like the ones above with exposure. But we suspect that this same factor leads those who use theorem provers regularly to underestimate the uphill struggle a typical mathematician faces when reading the proofs like the ones above).

Part of the difficulty in reading these proofs is due to an overreliance on symbols; in proofs written by humans, symbolic material plays a very specific and limited role (Ganesalingam, 2013), which does not include many of the uses in the example above. But a larger part relates to higher-level structure: none of the proofs presented above *flow* like real mathematical proofs. Mathematical proofs are like natural language texts—such as the document you are reading—in that sentences link together to form coherent discourses. From a mathematician’s perspective, this is what allows a proof to read as presenting a coherent argument rather than merely a sequence of facts. The principles behind discourse structure have been extensively studied by linguists (see e.g. Asher and Lascarides (2003)). We do not know of a way of tacking on this kind of structure, which is key to readability, to the proofs presented in this section; in order to produce it we have had to design our prover to support it from the ground up. In particular, the prover operates in a way that models a human closely enough to let it suppress irrelevant information and produce much larger blocks of discourse-coherent text than in the proofs given above.

2.2 Other Related Work

The constraints involved in producing human-like writeups mean that our prover functions in a very similar way to some older human-oriented provers produced by Woody Bledsoe and his students (Bledsoe, 1971, 1977a, 1977b, 1983, 1995; Bledsoe, Boyer, & Henneman, 1972; Ballantyne & Bledsoe, 1977; Bledsoe & Hodges, 1988). His *Non-resolution theorem proving* (Bledsoe, 1977a) outlines the main concepts involved in human-oriented theorem proving; a number of these are very relevant to

²At a deeper level there are further differences. For example the Theorema system will perform a certain amount of backtracking even on routine problems such as that of showing that a limit of the sum of two sequences is the sum of the limits. Thus, while there are many points in common between our aims and those of the Theorema project, there are also important differences.

the work we describe below, including the use of rewrite rules, forward chaining, (a kind of) typing, a reluctance to expand definitions unless necessary, and the use of ‘natural systems’. Further, as we shall describe below, our system shares the ‘waterfall architecture’ used in the provers created by Bledsoe’s students Boyer and Moore (1979).

Although human-oriented proving has largely fallen out of fashion, there have been some more recent systems that mimic natural proof styles. The most notable of these is Weierstrass (Beeson, 1998), which is capable of generating ϵ - δ proofs. One distinctive feature of this system is as follows:

The intention [of Weierstrass] is, to produce a proof that can be read and checked for correctness by a human mathematician; the standard to be met is “peer review”, just as for journal publication. By contrast, the purpose of Weierstrass is *not* to produce formal proofs in a specified formal system.

As we shall discuss more extensively in Sect. 3, the prover we are describing here has exactly the same goals as Weierstrass: it aims to produce proofs for human consumption, not proofs that are formally certified correct.

The article just cited does not provide any actual ϵ - δ proofs, noting simply that “the strict length limit does not permit the inclusion of the actual output of Weierstrass”. Similarly an article on the irrationality of e (Beeson, 2001) does not contain any actual output. We have also been unable to obtain the program and are therefore not able to comment on how human-like the output of Weierstrass actually is.

Another recent human-oriented system is Analytica (Clarke & Zhao, 1992), a theorem prover built inside Mathematica. The output of this system contains some words, but it is much less human-like than even the systems described in the previous section; essentially it prints out a long sequence of equations connected by phrases such as ‘reduces to’ and ‘rewrites as’. The result is not written in sentences, let alone being grammatical, so we do not classify Analytica with the systems described in the previous section (This is not a criticism of Analytica, as its authors make no claim to produce human-readable output).

We should emphasize that the work we describe is not meant to be competitive with any of the provers described here when considered as a prover; as noted above, the additional constraints involved in generating human-readable writeup are important to our project, but they also rule out many sound and effective tactics. Thus, our system is operating at a disadvantage, though it is our hope that ultimately the extra work we have to do now will help us see how to design systems that are more powerful than existing systems at solving the kinds of problems that human mathematicians are good at.

There is also an extensive literature on systems that accept natural language-like *input*, including MIZAR (Trybulec, 1978), NaProChe (Kuhlwein, Cramer, Koepke, & Schröder, 2009), ForTheL (Vershinin & Paskevich, 2000) and MathNat (Humayoun & Raffalli, 2010); we will not discuss these further here because the acceptance of human-like input and generation of human-like output are, from a linguistic perspective, almost unrelated problems. Additionally most systems of this kind focus

on checking that proofs are correct, which is (as discussed above) not a concern for most human mathematicians. Thus, they serve a different purpose from that of the work we are describing here.

3 Key Features of the Prover

The basic architecture of our prover is, as we have mentioned, very similar to the ‘waterfall’ architecture used in the Boyer-Moore provers. The program contains a number of heuristics that transform the goal in a sound manner. These heuristics are ranked by attractiveness. The program operates fully automatically by repeatedly applying the most attractive heuristic that has an effect. The underlying logic is essentially first-order logic with equality, though there are some caveats to be made here (in particular involving structural sharing of formulae and the use of metavariables) which we will outline as we go through the paper.

Notwithstanding the overall waterfall architecture, the details of the operation of the program are actually better described with reference to the goal-oriented tactic-based style of proof used in some interactive LCF-style theorem provers (Paulson, 1987). The correspondence may be drawn simply by regarding each heuristic as a tactic; the program can then be thought of as repeatedly applying a single tactic, which is itself constructed by taking a list of subsidiary tactics and applying the first that can be applied. In the remainder of the paper we will refer to tactics rather than heuristics, not least because we do not expect the waterfall architecture to be the last word on selecting tactics for a system of this kind.

We should stress that there is a key respect in which our system is not LCF-like. Like the Weierstrass system described in Sect. 2, it is not attempting to certify proofs correct in the way that LCF provers do; there is no small ‘safe’ kernel. Although we have every respect for the goal of certification, our concern in this paper is the generation of human-like proofs.

While it would certainly be possible to produce a program that generated human-like proofs and certified them correct, this is not an easy exercise for a number of reasons. One of these relates to the different number systems in use in mathematics: most work in theorem proving treats, say, the natural number 3 as being distinct from the rational number $3/1$, whereas human mathematicians consistently treat them as the same object. One way of seeing this is to consider a standard definition like

$${}^n C_r = \frac{n!}{r!(n-r)!}$$

in which the definiendum is a natural number but the definiens is a rational number. Handling the number systems in a human-like fashion in a theorem prover is far from straightforward because under standard accounts, the natural numbers are *used* to define the integers and thence the rational numbers. A full treatment requires an

analysis of the operation of identification, which in turns requires modifications to the underlying logic (Ganesalingam, 2013).

In the remainder of this section, we highlight some other significant respects in which our approach differs from more typical approaches.

3.1 *Targets, and the Structural Sharing of Goals*

Rather than having a list or stack of goals, we have a single goal, which may be regarded as a structurally shared version of the collection of goals found in a prover such as HOL. This structural sharing is necessary both for efficiency and in order to model human reasoning processes closely enough to produce human-like write-ups. A human reasoner will generally keep track of a list of background ‘assumptions’ which are known to be true (and likely to be helpful) and a separate list of statements that need to be proved. Under many circumstances the human will reason forwards from the assumptions without knowing in advance which of the statements to be proved will be deduced. If each goal were represented independently, this kind of forwards reasoning would have to be performed more than once. Thus, in our system the goal consists of a list of assumptions and a list of statements to be deduced from those assumptions, which we refer to as *targets*. If we represent the goal as a sequent $A_1, \dots, A_m \vdash B$, where B is of the form $B_1 \wedge \dots \wedge B_k$, then the targets are the statements B_i .

We should emphasise that our targets are not the same as the consequents C_1, \dots, C_k of a sequent $A_1, \dots, A_m \vdash C_1, \dots, C_k$; consequents are interpreted disjunctively, whereas targets are to be interpreted conjunctively. Thus our sequents always have a single consequent, and the conjuncts of that consequent are the targets. This is a good illustration of our priorities. Consequents may be more attractive from a logical point of view for symmetry reasons, but the convention we adopt, where we list the assumptions and targets, is in our judgment closer to how humans would think of ‘the current state of play’ when they are in the middle of solving a problem, and that is more important to us than logical neatness.

3.2 *The Library*

Like many other systems, the program also has a library of statements that it may assume. However, the role that our library plays is very different. With an interactive program whose main aim is proof verification, the library will typically be a huge database of statements that have already been fully checked and can therefore be used safely.

By contrast, for us the library is provided by the user and represents a body of results and definitions that a human mathematician would know and feel free either to quote or to use silently when trying to solve the problem at hand. Thus, the question

of whether or not a statement is appropriate for our library is one that has to be considered carefully, and the answer varies from problem to problem. For example, if the system is asked to prove the associativity of set intersection, then we do not want it to say, ‘This result is already in the library, so we are done.’ But for a more advanced problem, we want the associativity of set intersection to be available to the program (or perhaps even built in to its reasoning processes) rather than requiring the program to formulate and prove it as a lemma. As ever, this is because we are trying to imitate human mathematical thought and so generate human-like output: one of the things an undergraduate mathematician has to learn is what it is appropriate to assume when proving a result.

Thus, to use the program, it is not usually enough simply to input the goal and hope that the program will manage to prove it. Typically, the user has to add appropriate background results to the library first.

The library that we have used with the problems we have worked on is small, so we have not had to face the potential problem of the program using a result that is ‘more advanced’ than what it is supposed to be proving. However, when this problem does arise, as it inevitably will, we plan to order our library results using a relation ‘is more advanced than’, so that for each problem we can simply instruct the library not to use results beyond a certain point in the ordering.

We have also adopted a policy of being sparing about what we allow in the library. Human mathematicians do not store all the true mathematical statements that they come across in their brains. Rather, they seek out statements or techniques that have a kind of general quality. Exactly what this quality is, we do not claim to understand. It may be that for a more sophisticated program one should not expect to be able to judge the appropriateness of a statement in advance, but should store promising statements and then gradually forget about them if they turn out not to be especially useful.

Two examples of results that we have in the library we currently use are the statement that a closed subset of a metric space contains its limit points, and transitivity of $<$. Actually, it is convenient to store four separate transitivity statements such as

$$\begin{array}{l} a < b \\ b \leq c \\ \hline a < c \end{array}$$

one for each way of choosing $<$ and \leq . This saves the program from rederiving these variants from the transitivity of $<$. Once again, this is because we are modelling human thought: a typical human mathematician will internalize all four transitivity statements and use them without thinking. It would look very strange in a write-up if the program stopped to prove the above transitivity statement as a lemma.

A statement such as the triangle inequality would be most usefully stored in the library in the following form.

$$d(x, y) < \alpha$$

$$d(y, z) < \beta$$

$$\alpha + \beta \leq \gamma$$

$$d(x, z) < \gamma$$

In this form it is appropriate for most simple deductions, and reflects quite well how human mathematicians use the inequality. However, we do not yet have a satisfactory understanding of the process whereby human mathematicians are told the triangle inequality in the usual form

$$d(x, z) \leq d(x, y) + d(y, z)$$

and then quickly use it to have thoughts such as, ‘I need $d(x, z)$ to be less than γ , so it will be enough to ensure that $d(x, y) < \gamma/2$ and $d(y, z) < \gamma/2$.’ That is not to say that we cannot think of a mechanical method that would manage to make a deduction like this: the difficulty comes when we try to understand (in order to imitate) how humans do it.

At the moment, the library contains four kinds of data:

1. Results, i.e. facts that the program can utilise while constructing a proof.
2. Definitional expansions.
3. Term rewrite rules.
4. Constructions.

Results and (definitional) expansions are the key to reasoning at a high level, which is in turn one of the keys to producing human-like proofs. Term rewrite rules and instructions play a much smaller role in the program. The former are used to change e.g. $(f \circ g)(x)$ into $f(g(x))$, which is an instinctive and automatic step for any human mathematician. Constructions generally ‘finish off’ a problem by supplying an object with certain properties that are being looked for. For example, suppose the program needs to find x s.t. $x \leq a$ and $x \leq b$; the library can be used to construct $\min\{a, b\}$, which has the necessary properties.

3.3 *Removal of Assumptions*

Another unusual aspect of our approach involves paying close attention to the removal or deletion of assumptions. We include tactics for removing assumptions not because they have a significant effect on the output of the program, but because of our commitment to modelling how human mathematicians think. The particular phenomenon we are trying to model is that when humans work out proofs, they often find it obvious that a statement has been ‘used up’ and will have no further role to play in

the proof. We have modelled this by incorporating tactics that, under appropriate circumstances, remove assumptions from the list of all assumptions.

One reason we are interested in the removal of assumptions is that it forces us to think about a relatively simple special case of a hard and important general problem in theorem proving, namely the problem of deciding which statements are likely to be relevant to the solution of a given problem. It is possible to write programs that search through very large numbers of statements until they find something that magically works, but humans do not do this. So we feel that any efforts we make to understand this aspect of how humans do mathematics will pay dividends when we try to scale up to systems that will find more complex proofs. Another advantage of including tactics that remove assumptions is that it makes it considerably easier to debug cases where the program is stuck, by removing a lot of the ‘noise’ that makes it hard to understand intermediate goals.

4 Writing Up

In general, natural language generation is a complex process. It involves multiple levels of planning, which draw on both domain knowledge and models of the intended audience, and also a phase when the actual text is generated, which draws on syntactic, morphological and lexical information. An overview of the process may be found in Reiter and Dale (2000). Because of this complexity, building a fully fledged natural language generation system is a major task. Furthermore, since mathematics contains not just English words but also a large array of distinctive symbols used in distinctive ways, it is not at all straightforward to use off-the-shelf systems.

Fortunately, mathematical language has properties that make the task considerably simpler than it is for the English language in general. Foremost among these is the fact that mathematical proofs almost always have a particularly simple rhetorical structure. To some degree this is because the domain of discourse includes only timeless facts, which itself rules out a large proportion of the rhetorical relations found in general text. But the main reason is that there is a strong convention that further constrains the rhetorical structure of proofs. A proof proceeds by the presentation of a sequence of assertions, each of which follows from the premises of the theorem being proved or from previous assertions. This structure is not accidental; it is a direct reflection of the fact that mathematicians process proofs by reading and verifying one sentence at a time, and would not expect the justification of a fact presented in one sentence to be deferred to a later sentence (We are talking here about proofs of the level of simplicity of the proofs discussed in this paper. For more complicated arguments, facts may sometimes be used before they have been proved, but in good mathematical writing this will be carefully flagged up to make it as easy as possible for the reader to check that the resulting argument is complete and not circular).

This convention gives us an easy way to produce write-ups of our proofs. An obvious strategy is to allow each application of a tactic to generate some number of sentences (possibly zero), and then to concatenate the output from the different

tactics to produce the final text. Note that this strategy is viable only because we are absolutely rigorous about requiring our tactics to reflect steps in human reasoning; in effect, the strategy is mimicking a human who is carefully writing down a proof while coming up with it, which is quite straightforward for an experienced mathematician (Again, this becomes less true if the proofs are more difficult). As we shall see below, this simple strategy produces surprisingly good results, though with a weakness that needs to be dealt with by a postprocessing phase that turns out to be straightforward.

Because we have a fixed list of tactics, implementing the strategy only requires us to specify which sentences (if any) are produced for the applications of each tactic. A very simple way to do this is to use *template generation*: each tactic is associated with a *template*, or ‘piece of text with holes’, and the holes are filled in with concrete information about the facts and objects used in the actual application. So, for example, forwards reasoning may be associated with a very simple template ‘since $\langle \text{facts} \rangle$, $\langle \text{deduced fact} \rangle$ ’. Instantiating this template would produce text like

Since A is open and $x \in A$, there exists $\eta > 0$ such that $u \in A$ whenever $d(x, u) < \eta$.

Note that individual facts are expressed in idiomatic ways, rather being displayed in a way that directly reflects the underlying predicate calculus; thus we have ‘ A is open’ and ‘ $\eta > 0$ ’ rather than ‘ $\text{open}(A)$ ’ and ‘ $\text{greater_than}(\eta, 0)$ ’. The same is true of objects: we display ‘ $f \circ g$ ’ rather than $\text{compose}(f, g)$, and so on. Similarly quantification is expressed idiomatically using words like ‘whenever’, where possible, rather than using more stilted phrases like ‘for all’, which would more directly reflect the underlying predicate calculus.

An example of the text produced by this method is as follows:

Let x be an element of $A \cap B$. Since $x \in A \cap B$, $x \in A$ and $x \in B$. Since A is open and $x \in A$, there exists $\eta > 0$ such that $u \in A$ whenever $d(x, u) < \eta$. Since B is open and $x \in B$, there exists $\theta > 0$ such that $v \in B$ whenever $d(x, v) < \theta$. We would like to find $\delta > 0$ s.t. $y \in A \cap B$ whenever $d(x, y) < \delta$. But $y \in A \cap B$ if and only if $y \in A$ and $y \in B$. We know that $y \in A$ whenever $d(x, y) < \eta$. We know that $y \in B$ whenever $d(x, y) < \theta$. Assume now that $d(x, y) < \delta$. Since $d(x, y) < \delta$, $d(x, y) < \eta$ if $\delta \leq \eta$. Since $d(x, y) < \delta$, $d(x, y) < \theta$ if $\delta \leq \theta$. We may therefore take $\delta = \min\{\eta, \theta\}$. We are done.

The main problem with this text is that it suffers a lack of coherence, in the sense defined in Knott (1996): the sentences are individually acceptable, but they do not combine to form an idiomatic discourse. The principal reason for this is that the text repeats information unnecessarily. For example, in

Since $x \in A \cap B$, $x \in A$ and $x \in B$. Since A is open and $x \in A$, there exists $\eta > 0$ such that $u \in A$ whenever $d(x, u) < \eta$.

the repetition of the underlined phrase is awkward. Because it is introduced by the sentence immediately preceding the ‘since’ clause, it is awkward to have it spelt out explicitly within that clause. Similarly, consider:

Since $d(x, y) < \delta$, $d(x, y) < \eta$ if $\delta \leq \eta$. Since $d(x, y) < \delta$, $d(x, y) < \theta$ if $\delta \leq \theta$.

Here, having two identical ‘since’ clauses in consecutive sentences is again awkward: the repetition of material is unwieldy and unidiomatic.

We are of the opinion that (Knott, 1996) correctly diagnoses the underlying problem here: spelling out rhetorical relations, or aspects of rhetorical relations, that can easily be inferred from the context violates Grice’s maxim of quantity (Grice, 1975). Often the solution is to substitute an appropriate and less explicit *cue phrase*. For example, ‘since A is open and $x \in A$, ...’ is better replaced by ‘therefore, since A is open, ...’. The cue phrase ‘therefore’ (which assumes that the relevant reason has just been given) is less explicit than the cue phrase ‘since’ (which subordinates an explicitly stated reason), so it avoids spelling out information that is clear from the context. In other cases repetition can be avoided by combining sentences; thus the previous example may be changed into

Since $d(x, y) < \delta$, $d(x, y) < \eta$ if $\delta \leq \eta$ and $d(x, y) < \theta$ if $\delta \leq \theta$.

The initial ‘sentence by sentence’ process described above is followed by a series of transformations that manipulate pairs of consecutive sentences in order to resolve the issues just mentioned (Needless to say, the transformations operate on a structural level rather than on the literal text). Applying this series of transformations to the example text above yields:

Let x be an element of $A \cap B$. Then $x \in A$ and $x \in B$. Therefore, since A is open, there exists $\eta > 0$ such that $u \in A$ whenever $d(x, u) < \eta$ and since B is open, there exists $\theta > 0$ such that $v \in B$ whenever $d(x, v) < \theta$. We would like to find $\delta > 0$ s. t. $y \in A \cap B$ whenever $d(x, y) < \delta$. But $y \in A \cap B$ if and only if $y \in A$ and $y \in B$. We know that $y \in A$ whenever $d(x, y) < \eta$ and that $y \in B$ whenever $d(x, y) < \theta$. Assume now that $d(x, y) < \delta$. Then $d(x, y) < \eta$ if $\delta \leq \eta$ and $d(x, y) < \theta$ if $\delta \leq \theta$. We may therefore take $\delta = \min\{\eta, \theta\}$ and we are done.

One particular point worth emphasising is that the write-up process is deterministic: it will always produce the same output text for any given proof. This is for two reasons. First, if any non-determinism had been present we would have had to evaluate many outputs for any given proof, which would have made iterative improvement and fine-tuning of the write-ups considerably slower. Secondly, and more importantly, if the process were nondeterministic, our claim that the program produced human-like output would be suspect, in that we would have been able to run the program several times and ‘cherry pick’ output. Unfortunately, this determinism has an undesirable (but fully anticipated) side-effect. When one compare several proofs produced by the program, the write-ups are much more similar than those a human would produce. For example, most proofs produced by the program end with the phrase ‘we are done’. In the long run, we will undoubtedly need to introduce nondeterministic stylistic variation, allowing the program to vary the text generated for a particular step in just the way human would, despite the difficulties that will cause.

Finally, it is worth noting that during the evaluation process described in Sect. 7, we collated a wealth of data on how humans write up proofs. We anticipate using this data in combination with carefully chosen natural language processing techniques to create substantially improved versions of the write-up procedure.

5 Technical Details

5.1 Formalism

The formulae used in the program are essentially formulae of first-order logic, augmented with metavariables. More formally, we have:

1. A collection of *functions*, each with a distinct name and an (implicit) arity. An example is *compose*.
2. A collection of *predicates*, each with a distinct name and an (implicit) arity. An example is *less_than*. Equality is represented by a distinguished predicate.
3. A collection of *mathematical types*, each with a distinct name. An example is *positive real number*. At the moment the collection of types is specialized for use in problems involving metric spaces.
4. A *variable* is specified by a name (typically a single character), a mathematical type and a ‘variable type’, which indicates whether or not the variable is a normal variable or one of two kinds of metavariable, discussed in Sect. 5.8. Variables are also annotated with information indicating that certain variables are independent of other variables, which constrains inference.
5. A *term* is either a variable or a function together with a list of terms of appropriate arity.
6. An *atomic formula* consists of a predicate together with a list of terms of appropriate arity.
7. A *formula* consists of one of the following, where v_i are variables and F_i formulae:
 - An atomic formula.
 - $\neg F_1$
 - $F_1 \vee F_2$
 - $F_1 \wedge F_2$
 - $\forall v_1 \dots v_k. F_1$
 - $\forall v_1 \dots v_k. (F_1 \wedge F_2 \wedge \dots \wedge F_n \Rightarrow F_{n+1})$
 - $\exists v_1 \dots v_k. F_1$

As discussed in Sect. 3, the structural sharing of goals means that the higher-level datatypes used by the program are different from those used in e.g. an LCF-style prover. The key datatypes are defined recursively as follows:

1. A *box* is either a *nontrivial box* or the special box \top .
2. A *nontrivial box* consists of a list of variables, a list of formulae (called *assumptions*) and a list of *targets*.
3. A *target* consists of either a formula or a list of boxes.

In the absence of variables, the box consisting of variables $v_1 \dots v_k$, assumptions $H_1 \dots H_n$ and targets $T_1 \dots T_m$ corresponds to the aim of proving that the formula

$$\forall v_1 \dots v_k. (H_1 \wedge \dots \wedge H_n \Rightarrow T_1 \wedge \dots \wedge T_m)$$

holds. Where metavariables are present the corresponding quantifiers need to be existential rather than universal.

Targets that consist of a list of boxes correspond to a disjunction of the formulae corresponding to the individual boxes.

The goal consists of a single box, and tactics are functions that map boxes to boxes. In the rest of this document, we will display the box consisting of variables $v_1 \dots v_k$, assumptions $H_1 \dots H_n$ and targets $T_1 \dots T_m$ as follows:

$$\begin{array}{c} H_1 \\ \vdots \\ H_n \\ \hline T_1 \\ \vdots \\ T_m \end{array}$$

Note that we suppress the variables as they tend to clutter the exposition; they are however present in the debug code produced by the program. Where a target consists of one or more boxes, we draw a rectangle around each box to delineate it.

We use the term *statement* to refer to a formula that appears either as an assumption or a target in some box. Statements and boxes may be tagged with additional information; for example, when a statement has been used together with certain other statements by a tactic, it is tagged to indicate this. Tags never affect the logical interpretation of the tagged material, but are used when the program is evaluating whether tactics are permissible. In particular, tags are used to prevent repeated application of the same tactic to the same statements.

Both the human-like output and debugging code prettify the output in conventional ways, for example by writing $a < b$ for the atomic formula *less_than*(a, b). We adopt such conventions throughout this document. In cases of ambiguity, quantifiers should always be read as taking the widest scope possible.

5.2 Terminology

When a statement is built out of atomic statements using connectives and quantifiers, the program classifies it according to the operations that appear at the top of its parse tree. For example, the statement

$$\exists x (x \in A \wedge d(x, y) < \epsilon)$$

is an *existential* statement, whereas the statement

$$x \in A \wedge d(x, y) < \epsilon$$

is *conjunctive*. It is often useful to look more than one level down the parse tree. For example, the program would call the statement

$$\forall x (x \in A \Rightarrow x \in B)$$

a *universal conditional* statement (Because we do not allow ‘bare’ conditional statements, this is a particularly important category). Similarly, the existential statement above can be further classified as an *existential conjunctive* statement.

Finally, many atomic statements can be expanded into statements that are no longer atomic. For example, the statement $A \subset B$ expands to the universal conditional statement above. It is often useful to know what a statement will become after it is expanded: to specify this the program uses the prefix ‘pre-’. Thus, the statement $A \subset B$ is pre-universal conditional. Similarly, the statement “ x has an inverse” is pre-existential because it expands (in a suitable context) to

$$\exists y xy = yx = 1$$

and the statement “ A is unbounded” is pre-universal because it expands to

$$\forall C \exists a \in A a > C.$$

An expansion is *elementary* if it does not introduce a quantifier. For example, the expansion of $A \subset B$ is not elementary, whereas the expansion of

$$x \in A \cap B$$

as

$$x \in A \wedge x \in B$$

is elementary.

5.3 *Substantive Hypotheses and Background Conditions*

Consider the following definition. If A is a subset of a metric space X and $x \in X$, then x is an *accumulation point* of A if

$$\forall \epsilon > 0 \exists a \in A d(a, x) < \epsilon.$$

An obvious way of rewriting this is

$$\forall \epsilon (\epsilon > 0 \implies (\exists a (a \in A \wedge d(a, x) < \epsilon))).$$

However, that does not reflect how a mathematician will think about the definition. The real number ϵ is not something that might conceivably be negative but happens to have a useful property if it is positive. Rather, when it comes to selecting ϵ , the universe from which we select it is the set of positive real numbers. So the condition $\epsilon > 0$ is not a ‘substantive’ assumption, but more like a background condition. By contrast, the statement $a \in A$ is substantive: it is an element of X that has the further interesting property of belonging to A .

We capture this in our program by representing background conditions through our type system. That is, instead of explicitly saying that $\epsilon > 0$, we will take ϵ to have the type ‘positive real number’.

Note that this is important for the categorization of statements discussed in the previous subsection. For example, we want to think of a statement such as

$$\forall \epsilon > 0 \exists N \forall n \geq N d(a_n, x) < \epsilon$$

as a universal existential statement and not a universal conditional statement. This is achieved by having the program represent it as

$$\forall \epsilon \exists N \forall n (n \geq N \implies d(a_n, x) < \epsilon)$$

and having the type system capture the fact that ϵ is a positive real number.

It is also important for deciding when a deduction is likely to be relevant. Suppose that we have a universal-conditional statement in the library that has several premises that match our assumptions. To mimic human reasoning, we would usually like this to count as evidence that the library statement is relevant, but not if the assumptions are merely background statements: if a library result requires a positive real number, we will not get excited just because we have a positive real number floating around.

5.4 The Waterfall

The following lines are taken directly from the program’s code: they list, in order of priority, the names of the tactics it can use (The names of the tactics do not always match the terminology used in this paper, which we have standardized to match the standard terminology used in interactive provers as far as we can). In Sects. 5.5–5.9, we shall describe each tactic in turn.

```
--Deletion
deleteDone,
deleteDoneDisjunct,
deleteDangling,
deleteUnmatchable,
```

```

--Tidying
  peelAndSplitUniversalConditionalTarget,
  splitDisjunctiveHypothesis,
  splitConjunctiveTarget,
  splitDisjunctiveTarget,
  peelBareUniversalTarget,
  removeTarget,
  collapseSubtableauTarget,
--Applying
  forwardsReasoning,
  forwardsLibraryReasoning,
  expandPreExistentialHypothesis,
  elementaryExpansionOfHypothesis,
  backwardsReasoning,
  backwardsLibraryReasoning,
  elementaryExpansionOfTarget,
  expandPreUniversalTarget,
  solveBullets,
  automaticRewrite,
--Suspension
  unlockExistentialUniversalConditionalTarget,
  unlockExistentialTarget,
  expandPreExistentialTarget,
  convertDiamondToBullet,
--EqualitySubstitution
  rewriteVariableVariableEquality,
  rewriteVariableTermEquality

```

We stress here that because our system is fully automatic and intended to model human thought processes, our efforts have been concentrated less on the tactics themselves and more on how the program chooses which tactic to apply. For this program, the general form of the answer is as follows: it just chooses a tactic of the first type it can apply from the list above. Thus, if a tactic of type `deleteDone` can be performed, it performs it. If not, but a tactic of type `deleteDoneDisjunct` can be performed, then it performs that. Otherwise, it tries `deleteDangling`. And so on. In this way our architecture is similar to the ‘waterfall’ architecture used by Boyer and Moore in their provers NQTHM and ACL2. Like them we have tended to give higher priority to ‘lower-risk’ tactics, since this appears to correspond well to the way humans choose what to do; one of the challenges in generating human-like proofs is to assess correctly the risk of the different tactics. We discuss this point more fully in Sect. 5.10.

In the remainder of this section we shall describe the tactics, splitting them into some broad categories. One general preliminary remark is that a large proportion of the tactics contain minor ‘postprocessing’ steps, applying operations such as existential elimination. This is appropriate in precisely the cases where a human would

apply such an operation in so automatic a fashion that it would not have any effect on the writeup. We will not mention these postprocessing steps in the tactic descriptions below as to do so would make them much longer (Readers may of course consult the source code for full technical details).

5.5 Removal Tactics

These tactics remove assumptions (cf. Sect. 3.3) and targets. They never directly contribute material to the writeup.

deleteDone

There are several situations where a tactic results in a target being replaced by \top because it has been proved. Once this has happened, the program immediately uses this tactic to remove it. Thus, the aim of the program is to reach a goal with no targets.

deleteDoneDisjunct

If a target is disjunctive and one of its disjuncts is \top , then the entire target is removed.

deleteDangling

If an assumption has previously been used and contains a variable that is not involved in any other statement, then the program removes the assumption.

deleteUnmatchable

Roughly speaking, this tactic removes assumptions that have previously been used by a tactic and that have no obvious use.

For example, suppose that we have the statements $x \in A$ and $A \subset B$ as assumptions. The expansion of $A \subset B$ is $\forall u (u \in A \implies u \in B)$. If we substitute x for u , then the premise of this statement becomes $x \in A$, which is identical to the assumption. We say that $x \in A$ *matches* the premise of (the expansion of) $A \subset B$. We call a statement *unmatchable* if there are no available matches for it.

The program is not allowed to substitute the same variable twice into the same assumption (This is partly because no human would ever do so, and partly to avoid non-termination). This can create further circumstances where an assumption is unmatchable. For example, suppose we apply forwards reasoning to the statements $x \in A$ and $A \subset B$ to deduce that $x \in B$. Then we can no longer use the match between $x \in A$ and $A \subset B$, so $x \in A$ becomes unmatchable (assuming that there is no other statement that matches it). Since it has been used, it will therefore be removed. If no other statement matches $A \subset B$, then that statement too is unmatchable and will therefore be removed, since it too has been used.

5.6 Tidying Tactics

Tidying tactics are tactics that do not substantially change the content of a target, but put it into a more convenient form.

peelAndSplitUniversalConditionalTarget

If the target takes the form $\forall x (P(x) \Rightarrow Q(x))$, then this tactic removes it and adds instead a variable x , a new assumption $P(x)$ and a new target $Q(x)$. This corresponds to the human tactic of saying (or thinking), ‘Let x be such that $P(x)$; we need to show that $Q(x)$.’

The tactic could be thought of as a composition of two more basic tactics, one an application of universal generalization and the other an application of implication introduction, but again our avoidance of bare conditionals demands that we treat it as unitary.

If there is more than one target, then this tactic has to be modified, since we cannot use the new assumption $P(x)$ to help us prove a different target. In that situation, we create a target that consists of a box.

That is, if we have a goal of the form

$$\begin{array}{c}
 H_1 \\
 \vdots \\
 H_n \\
 \hline
 \forall x (P(x) \implies Q(x)) \\
 R
 \end{array}$$

then the tactic will transform it to

$$\begin{array}{c}
 H_1 \\
 \vdots \\
 H_n \\
 \hline
 \boxed{
 \begin{array}{c}
 P(x) \\
 \hline
 Q(x)
 \end{array}
 } \\
 R
 \end{array}$$

The program can then use $P(x)$ to prove $Q(x)$ but not to prove R . The assumptions in the outermost box (which in the diagram above we do not enclose) can be used to prove both statements.

This tactic generates a representation which (if in isolation) would eventually be transformed into text like “Let x be such that $P(x)$.”

splitDisjunctiveHypothesis

If there is an assumption of the form $P \vee Q$, then the program removes the disjunction by replacing the target(s) by two new targets. One of these targets is a box with P as

an assumption and the original targets as its targets, and the other is a box with Q as an assumption and the original targets as its targets.

This tactic and its counterpart `splitDisjunctiveTarget` constitute work in progress; they are important for generating certain human-like proofs, but (because they split the proof into cases) they are not compatible with the incremental, concatenative tactic-by-tactic generation of writeups. We intend to extend the writeup generation mechanism to accommodate these tactics in future versions. Note that neither tactic is used in the problems we evaluate on.

splitConjunctiveTarget

If there is a target of the form $P \wedge Q$, then it is replaced by two targets P and Q . This tactic does not contribute to the writeup.

splitDisjunctiveTarget

If there is a target that consists of a formula of the form $P \vee Q$, then it is replaced by a target consisting of boxes corresponding to P and Q .

This tactic exists for technical reasons: one reason is that the program sometimes likes to attach tags to statements, for example to record whether they have been used (which affects the deletion rules), but it has no facility for attaching tags to parts of statements. Therefore, if we want to use a tag to record information about one disjunct of a disjunctive target, we need to ‘split’ the target first.

Also see the note regarding `splitDisjunctiveHypothesis` above.

peelBareUniversalTarget

If the target is of the form $\forall x P(x)$ and P is not a conditional statement, then this tactic removes the universal quantifier.

This tactic generates a representation which (if in isolation) would eventually be transformed into text like “Take $\epsilon > 0$.”

removeTarget

This tactic is formed from a family of related sub-tactics whose common element is that under appropriate circumstances they replace a target with \top . The most obvious example is when a target equals an assumption (and that assumption is allowed to be used to prove the target). A more complicated example is when the target is of the form $\exists u (P(u) \wedge Q(u))$ and there are assumptions $P(x)$ and $Q(x)$. The other circumstances are similar.

The sub-tactics generate representations that are eventually transformed into text like “Therefore, setting $\delta = \theta$, we are done.”

collapseSubtableauTarget

If a target consists of a single box B that has no assumptions and contains no metavariables (see Sect. 5.8 for a discussion of these), then the target is replaced by the targets of B . This tactic does not contribute to the writeup.

5.7 Applying Tactics

An applying tactic is any tactic that corresponds to what human mathematicians would call applying an assumption, result or definition.

forwardsReasoning

The most basic form of this tactic is universal modus ponens: that is, a combination of substitution into a universal-conditional assumption followed by an implication elimination that uses the resulting conditional statement. In other words, one uses assumptions of the form $\forall u (P(u) \implies Q(u))$ and $P(x)$ to obtain a new assumption $Q(x)$.

A slight variant of this tactic that is worth highlighting is illustrated by the following simple piece of reasoning: if we know that $x \in A$ and that $A \subset B$ then we can deduce that $x \in B$. Humans will make this deduction in one step rather than first expanding the statement $A \subset B$ as $\forall u (u \in A \implies u \in B)$, then substituting x for u , and finally applying forward chaining. In order to produce a human-like writeup, our program does the same. In general, for each type of reasoning tactic that involves a universal conditional assumption, there is a variant that does the same thing to the expansion of a pre-universal conditional assumption.

This tactic also handles deductions that involve more than one premise, such as using assumptions of the form $P(x)$, $Q(x)$, and $\forall u (P(u) \wedge Q(u) \implies R(u))$ to obtain the assumption $R(x)$.

This tactic generates a representation which (if in isolation) would eventually be transformed into text like “Since $x \in A$ and $A \subset B$, $x \in B$.”

forwardsLibraryReasoning

This is reasoning that is ‘mathematical’ rather than ‘purely logical’. For example, from the statements ‘ (a_n) is a sequence in A ’, ‘ A is closed’ and ‘ $a_n \rightarrow a$ ’ one can deduce that $a \in A$. Experienced human mathematicians will perform this deduction in one step, because their mental library will contain the result that whenever a sequence in a closed set tends to a limit, then the limit belongs to the closed set as well. Mimicking this behaviour is very important for producing a write-up that is not cluttered with an inappropriate amount of detail.

Logically speaking, one could unify forwards library reasoning with ordinary forwards reasoning by adding the entire (allowable) content of the library to our list of assumptions. However, there are one or two aspects of library reasoning that give it a different flavour. The main one is that library results contain no free variables: they are general facts that apply universally. This distinguishes them from assumptions, which are more contingent. A second difference is that forwards library reasoning is normally used to deduce an atomic assumption from other atomic assumptions. A universal conditional statement is involved, but it is in the library and is not an assumption. For these reasons, reasoning that uses library results tends to be used by humans only when other forms of forwards reasoning are not available. Therefore, for the program it is important *not* to unify the two forms of reasoning, so that library reasoning can be given a lower priority.

It is also important to place some kind of restriction on forwards library reasoning to stop the program making obviously irrelevant deductions. For instance, if it knows that H is a subgroup of a group G and $x \in H$, and if a suitable result belongs to the library, then an unrestricted forwardsLibraryReasoning would allow it to deduce that $x^{-1} \in H$, but this deduction may well be of no help in solving the problem at hand and a step that no human would think of taking. As a safeguard against this, we forbid the program to apply the forwardsLibraryReasoning tactic if it creates a new term. This aspect of the program is not stable: it seems that although human mathematicians are indeed reluctant to create new terms in this way, they sometimes do so, even in some fairly straightforward problems. More work is needed to understand the circumstances under which such ‘speculative’ reasoning occurs.

This tactic generates a representation which (if in isolation) would be transformed into text like “Since (a_n) is a sequence in A , A is closed and $a_n \rightarrow a$, $a \in A$.”

expandPreExistentialHypothesis

As its name suggests, this means replacing a pre-existential assumption by its definition expansion (Recall that a statement is pre-existential if its definition expansion is an existential statement). What the name of the tactic does not reveal is that this expansion is followed immediately by an existential elimination. So for example expansion of the hypothesis ‘ A is bounded’ might result in the introduction of a variable M and a new hypothesis $\forall x (x \in A \Rightarrow |x| \leq M)$. We do this because human mathematicians almost always do it without comment, so our program should do so as well. Although these features have no effect on which problems the program can solve, they have a significant effect on the writing-up stage, saving the program from having to judge that certain steps do not need to be spelt out.

This tactic generates a representation which (if in isolation) would be transformed into text like “Since A is bounded, it follows that there exists M such that $|x| \leq M$ whenever $x \in A$.” Note that, following standard human practice, in the rest of the write-up the variable x would automatically be treated as an entity one could refer to: the program is like a human in considering this kind of instance of ‘there exists M such that ...’ as equivalent to ‘let M be such that ...’ (Also note that a human typically suppresses the domain of quantification of M in this case, i.e. a human does not write $M \in \mathbb{R}$, and the program does the same.)

elementaryExpansionOfHypothesis

This takes a assumption that has an elementary expansion (recall that this means an expansion that does not begin with a quantifier) and replaces it by that expansion. This is sometimes combined with some tidying. For example, if the assumption in question is $x \in A \cap B$, then the elementary expansion is $x \in A \wedge x \in B$, but this expansion is immediately converted into the two assumptions $x \in A$ and $x \in B$ and does not itself appear in any intermediate goal—again so that the write-up will be suitably concise.

This tactic generates a representation which (if in isolation) would be transformed into text like “Since $x \in A \cap B$, $x \in A$ and $x \in B$.”

backwardsReasoning

This is the obvious backwards counterpart of forwards reasoning, using universal modus tollens instead of universal modus ponens. The most basic form is thus that we are given a target $Q(x)$ and an assumption $\forall u (P(u) \Rightarrow Q(u))$, and we replace the target by $P(x)$.

More generally, if we have a target $Q(x)$ and an assumption $\forall u (P_1(u) \wedge \dots \wedge P_k(u) \Rightarrow Q(u))$, then it is logically sound to replace the target $Q(x)$ by the k targets $P_1(x), \dots, P_k(x)$, and many provers will happily do so. Our program is allowed to do this more complex backward chaining only under tightly constrained circumstances: we require all but one of the statements $P_1(x), \dots, P_k(x)$ to be an assumption, so that only one new target is created. This is another pragmatic decision: it is a crude way of deciding whether applying the assumption $\forall u (P_1(u) \wedge \dots \wedge P_k(u) \Rightarrow Q(u))$ is likely to be the right thing to do, and the severity of the test is intended to stop the program making ‘speculative’ deductions that risk leading to combinatorial explosion. It is clear that humans sometimes violate this rule, but more work is needed in order to understand when they do so.

As with forwards reasoning, there is a simple variant where the role of the universal conditional assumption is played by a pre-universal conditional assumption instead. For example, given a target $x \in B$ and an assumption $A \subset B$ the program could use this variant to replace the target by $x \in A$.

The contribution of this move to the writeup is complex. An example of the output it can produce is “We know that $y \in A$ whenever $d(x, y) < \eta$.”

backwardsLibraryReasoning

This is backwards reasoning that makes use of a general result in the library. However, it is slightly subtler than forwards library reasoning, because it always uses assumptions as well as a target. The precise rule is that if there are assumptions $P_1(x), \dots, P_{k-1}(x)$, a library result $\forall u (P_1(u) \wedge \dots \wedge P_k(u) \Rightarrow Q(u))$ and a target $Q(x)$, then the target can be replaced by $P_k(x)$. (Of course, the premises of the library result do not have to be stated in the order P_1, \dots, P_k).

An example of this kind of reasoning would be to say, “It is sufficient to prove that B is open,” if one wished to prove that $A \cap B$ was open and knew that A was open. This would be making use of the result that an intersection of two open sets is open.

Once again, the restriction we place is for pragmatic reasons: we do not want the program to make highly speculative transformations of the goal that introduce several new targets, since humans are usually reluctant to do this, especially when solving simple problems of the kind we are focusing on. But this is another situation where we would hope to improve the program in a future version, since humans do sometimes introduce multiple targets and then tick them off one by one.

The contribution of this move to the writeup is complex. An example of the output it can produce is “Since $d(x, y) < \delta$, $d(x, y) < \eta$ if $\delta \leq \eta$.”

elementaryExpansionOfTarget

This replaces a target by an elementary expansion of that target, if it has one. In the absence of metavariables, it generates a representation that will eventually be transformed into text like “we would like to show that A is open, i.e. that ...”. In the presence of metavariables, it generates a representation that will eventually be transformed into text like “we would like to show that $xy \in H \cap K$, i.e. that $xy \in H$ and $xy \in K$ ”.

expandPreUniversalTarget

This replaces a pre-universal target by its expansion. This tactic will be followed by one of the tidying tactics `peelAndSplitUniversalConditionalTarget` or `peelBareUniversalTarget`. It is usually the first tactic that the program applies when faced with a naturally stated problem.

This tactic does not generate any write-up text.

solveBullets

As we are just about to discuss in more detail, we sometimes convert a variable w into a metavariable. The metavariable needs at some stage to be chosen in such a way that the problem can be solved. If the variable only ever appears in targets, then one simple way in which this can often be done is to identify another variable x with the property that if we substitute x for w , then every target that involves w is equal to an assumption. In that situation, all those targets are replaced by \top .

This tactic generates a representation that will (after postprocessing) be transformed into text like “We may take $\epsilon = \dots$ and we are done.”

automaticRewrite

There are a few term rewriting rules stored as data in the library. An example is that the term $(g \circ f)(x)$ is rewritten as $g(f(x))$. These rewriting rules are intended to represent operations that are so automatic that a human would not comment on them, and accordingly this tactic does not contribute to the writeup.

5.8 Creation of Metavariables

We now come to a class of tactics alluded to earlier: tactics that help us deal with existential targets when it is not immediately clear what to substitute for the existentially quantified variable. A standard technique for this, which is essentially the technique we use, is to form *metavariables*. The rough idea of a metavariable is that one reasons with it as though it had been chosen, deferring the actual choice until later when it becomes clearer what choice will make the argument work. Mathematicians often use this trick: a classic example is the ‘ 3ϵ -argument’ used to prove that a uniform limit of continuous functions is continuous.

We have found it convenient to introduce two kinds of metavariable, to model two styles of reasoning that are logically similar but psychologically quite different.

As ever, this mimicking of human reasoning is necessary to produce a human-like writeup. These are displayed with diamonds or bullets, as described below.

unlockExistentialUniversalConditionalTarget

To illustrate this, suppose we have a target such as $\exists \delta \forall y (d(x, y) < \delta \Rightarrow f(y) \in B)$, and also a assumption $\forall u u \in A \Longrightarrow f(u) \in B$. Then it is easy to see that we can reduce the target to $\exists \delta \forall y (d(x, y) < \delta \Longrightarrow y \in A)$. However, this operation is not open to the program because it is not allowed to ‘reason inside quantifiers’. This is a matter of convenience: such operation are logically valid, but it is tedious to specify appropriate variants of several of the reasoning tactics listed above. Instead, we introduce a metavariable, which effectively moves aside the existential quantifier and allows the program to reason as normal with the statements inside it.

More precisely, what the program does to is replace the statement with a box whose variables include the metavariable that is being introduced. In the example above, it would have no assumptions and a single target $\forall y (d(x^\blacklozenge, y) < \delta \Rightarrow f(y) \in B)$. The diamond on the (meta)variable x indicates that x needs to be chosen.

It is important for the program not to interchange quantifiers accidentally. For this reason, we tag the box just created with the variable x^\blacklozenge , to indicate the scope of the existential quantification over x .

After ‘unlocking’ the statement, the program applies the `peelAndSplitUniversalConditionalTarget` tactic inside the box. After that, we have a box that looks like this.

$$\boxed{\begin{array}{c} d(x^\blacklozenge, y) < \delta \\ \hline f(y) \in B \end{array}}$$

Once we have done this, the statement $f(y) \in B$ has become an internal target and the program is free to apply backwards reasoning to it.

This tactic generates a representation that will (after postprocessing) be transformed into text like, “We would like to find x s.t. $P(x)$ whenever $Q(x)$.”

unlockExistentialTarget

This tactic replaces a target of the form $\exists x P(x)$ with a box that has the variable x^\blacklozenge , no assumptions and a single target $P(x^\blacklozenge)$.

This tactic will never be applied to an existential universal conditional target, since that will have been dealt with by `unlockExistentialUniversalConditionalTarget`. The main reason we have two separate tactics here is that we prefer to bundle the unlocking together with the `peelAndSplitUniversalConditionalTarget` tactic when that is possible.

To see what `unlockExistentialTarget` allows the program to do, suppose that we have a target of the form $\exists x (Q(x) \wedge R(x))$ and also a assumption of the form $\forall u (P(u) \Rightarrow Q(u))$. In this situation we would like to be able to do backwards reasoning inside the existential quantifier to reduce the target to $\exists x (P(x) \wedge R(x))$.

However, the program does not have a tactic for this. Instead, it unlocks the existential target, so that it has a box with a target $Q(x^\blacklozenge) \wedge R(x^\blacklozenge)$. The tidying tactic `splitConjunctiveTarget` can now turn this new target into two targets, and once it has done that, the applying tactic `backwardsReasoning` can be used to replace the target $Q(x^\blacklozenge)$ by $P(x^\blacklozenge)$.

As another example of the use of unlocking, suppose that we wished to prove that $A \cap B$ is non-empty and had the assumptions $x \in A$ and $x \in B$. The program cannot see that x is a witness to the non-emptiness of $A \cap B$ without doing some processing. An obvious first step is to expand the target into the statement $\exists u u \in A \cap B$. However, the program is not then allowed to do an elementary expansion inside the quantifier. Instead, it unlocks u so that there is a new target $u^\blacklozenge \in A \cap B$. This can now be expanded and split into the two targets $u^\blacklozenge \in A$ and $u^\blacklozenge \in B$, which `solveBullets` can then match with the assumptions.

This may seem a little circuitous, but it actually models quite closely how humans think. A human might say, ‘I want to show that $A \cap B$ is non-empty, so I need to find some u that belongs to $A \cap B$. In other words, I need u to be in A and in B . Aha, I can take x .’ The program’s unlocking models the silent disappearance of the existential quantifier before the second sentence of the above, which we need to model to produce a human-like writeup.

This tactic generates a representation which will (after postprocessing) be transformed into text like “We would like to find x s.t. $P(x)$.”

expandPreExistentialTarget

This does exactly what it says: it replaces a pre-existential target by its expansion. It generates a representation that will eventually be transformed into text like “We would like to show that”, explicitly presenting the expansion.

convertDiamondToBullet

There are certain tactics that the program will not apply to a ‘diamonded’ metavariable. In particular, it will not do any reasoning with an assumption that involves such a metavariable: for that it needs another kind of metavariable, roughly speaking corresponding to the human operation of ‘pretending that a variable has been chosen’ and then reasoning with it. Logically this is not an important difference, but it is a useful one for us because it reflects a difference in the way human mathematicians think and write. This helps the program to produce more convincing write-ups. When we convert a ‘diamonded’ variable into a full metavariable in this way, we change the diamond to a bullet.

We do not need separate tactics for reasoning that involves assumptions with bulleted metavariables: we just allow the reasoning tactics described above to handle such metavariables.

An important technicality is that if we postpone the choice of a metavariable, we must keep track of which other variables it is allowed to depend on. However, what we actually do is note which variables it is *not* allowed to depend on. This is for two reasons. First, it seems to reflect more accurately how human mathematicians think about such variables, and secondly, it is more economical: there are typically many

fewer variables on which a bulleted variable is not allowed to depend than variables on which it is allowed to depend.

This tactic generates a representation that will (after postprocessing) be transformed into text like “Assume now that”, explicitly stating all assumptions involving the relevant metavariable.

5.9 *Equality Substitution*

If we are told that two objects are equal, then we can eliminate all mention of one object in favour of the other. The precise rules governing when and how mathematicians tend to avail themselves of this opportunity are not obvious. The rules below are best regarded as a temporary solution: they do not always result in realistically human choices, and we intend to replace them by more satisfactory rules when we understand better what humans do.

rewriteVariableVariableEquality

If there is an assumption of the form $x = y$, then this tactic replaces all occurrences of y by x and eliminates the assumption.

This tactic generates a representation that will eventually be transformed into text like “Since $x = y$, ...”.

rewriteVariableTermEquality

If there is an assumption of the form $v = t$ or $t = v$, where v is a variable and t is a term, then this tactic replaces all occurrences of t by v .

This tactic generates a representation that will eventually be transformed into text like “Since $v = t$, ...”.

5.10 *Justification for the Order of Priority*

As we have already said, the tactics we use above are all either standard in themselves or simple combinations of standard tactics (with the possible exception of our distinction between ‘diamonded’ variables and more standard metavariables). Our main concern is not the set of tactics available to the program, but the way the program chooses which tactic to apply to any given goal. We have attempted to design this so that the program can produce a realistically human style of output in an incremental fashion. That is, each tactic needs to produce a list of human-like English sentences, or more accurately a list of elements of a datatype that correspond to such sentences. The postprocessing described in Sect. 4 does not change the fact that the output of the program very closely matches its inner workings. This feature of the program has governed many of the design decisions we have made.

How should the program decide which tactic to use in any given situation? Our methodology for answering this question was to work through large numbers of problems ourselves, making a note of which tactics seem appropriate. After a while we were in a position to make a first guess at a suitable method for choosing tactics. We then tried the method out on more problems, adjusting it when it led to inappropriate choices. After several iterations of this, we arrived at the order of priority of the tactics that we set out in the previous section.

If our only justification for the order of priority were that it leads to good results for the problems we have tried so far, it would not be very strong: what gives us any confidence that the order of priority will be appropriate for other problems that may be quite different from the ones we have looked at so far? However, there is an informal guiding principle that explains quite a lot (though not all) of the order of priority, which is that the program prefers “safe” tactics to “dangerous” tactics. As we mentioned earlier, the same is true of the order of priority chosen by Boyer and Moore in their ‘waterfall’ architecture (see Boyer and Moore (1979), p. 90).

Broadly speaking, a tactic is safe if the risk that it will lead to an undesirable result, such as a dead end or a step that is completely irrelevant to the eventual proof, is small. For example, tidying tactics are safe in this sense: by expressing the goal in a more convenient form, they open up new options without closing off any old ones. Since they are so safe, they come first in the order of priority. By contrast, expanding a definition is substantially less safe: sometimes it is possible to reason in a high-level way without expanding, and since we do not allow ‘de-expansion’ in this program (and in general allowing it would be highly problematic because of the danger of an infinite loop), expanding a definition is closing off the option of such high-level arguments, so we are reluctant to do it unless we have convinced ourselves that high-level arguments are not available. For example, if there is an assumption of the form ‘ (a_n) is Cauchy’, then we do not want our program to expand out the definition of Cauchy unless it has checked that it is not thereby closing off the option of a high-level deduction such as

$$\begin{array}{l} X \text{ is complete} \\ (a_n) \text{ is Cauchy} \\ \hline (a_n) \text{ converges} \end{array}$$

which would be a piece of forwards library reasoning in the program.

Thus, expansion has a fairly low priority. Having said that, some expansions, such as elementary expansions or expansions of pre-existential assumptions, are considerably safer, so those ones have higher priority.

Somewhere in between are the other reasoning tactics. Here it becomes more complicated to apply the general principle, even as an informal guide, because the

safety of a tactic depends heavily on context. In particular, forwards reasoning is in general fairly unsafe—if you have a lot of information and do not know which statements are relevant, then the probability that any given deduction will form part of the eventual proof may be quite small—but it is much safer when it comes to routine problems, which tend not to suffer from the problem of irrelevant information.

It seems that *when it is safe*, humans tend to prefer forwards reasoning to backwards reasoning (Sweller, Mawer, & Ward, 1983; Owen & Sweller, 1985), though this appears to be a question more of style than of problem-solving efficacy: we tend to prefer not to keep track of a moving target if we do not have to. Since forwards reasoning tends to be safe for the highly routine problems our program tackles, we have given all forwards reasoning a higher priority than all backwards reasoning. This also has the beneficial effect of making the program reluctant to switch direction—too much switching from forwards to backwards or vice versa would again be bad mathematical style.

This aspect of our program is, however, unstable, for the reason just given. When humans are faced with several possibilities for forwards reasoning, they will often switch to backwards reasoning in order to lessen the risk of making irrelevant deductions, but our program does not yet have any facility for making this kind of judgment.

One other feature of the ordering of reasoning tactics is that we prefer pure reasoning tactics to library reasoning tactics. That is because in general an assumption is more likely to be relevant than a library statement, though if enough of the premises of a library statement are present as assumptions, that is a fairly strong argument for its relevance.

At the bottom of the list of priorities is the process of creating metavariables. That is because humans tend to regard it as a last resort. When mathematicians need to prove statements of the form $\exists x P(x)$, then by and large they prefer to transform the goal using other tactics until a suitable candidate x_0 for x becomes obvious and it remains to carry out the relatively easy task of verifying that $P(x_0)$. Only when this straightforward approach fails do we take the more drastic step of pretending that x has been chosen.

We will not say much more here about how we chose the priority order, but we have two brief further points. First, although our reasons are not completely precise, we found that in practice they were adequate, in the sense that they suggested an order before we started, and we found that we did not have to modify the order when we tried further problems (though, as commented above, there are certain aspects of the architecture that will need to be changed in future versions). Secondly, when it comes to the finer detail of the ordering, there may not be that much to choose between different tactics. However, conflicts rarely arise between different tactics that are not distinguished by any of the above criteria, so in practice these finer details have little if any effect on what the program actually does.

6 Example of Operation: An Intersection of Two Open Sets Is Open

Now that we have discussed how the program works, let us look at another example, which involves most of the tactics we have discussed and shows how the order of priority works in practice. The problem to be solved is the following.

Problem 1 Let A and B be open subsets of a metric space X . Prove that $A \cap B$ is open.

The initial goal is represented as follows.

$$\begin{array}{l} A \text{ is open} \\ B \text{ is open} \\ \hline A \cap B \text{ is open} \end{array}$$

No reasoning tactics are possible, so we end up having to expand. The highest priority tactic we can do is `expandPreUniversalTarget`, which, after the tidying `peelAndSplitUniversalConditionalTarget`, has the following effect.

$$\begin{array}{l} A \text{ is open} \\ B \text{ is open} \\ x \in A \cap B \\ \hline \exists \delta \forall y (d(x, y) < \delta \Rightarrow y \in A \cap B) \end{array}$$

Recall that we do not explicitly specify here that $\delta > 0$, but instead take the positivity of δ to be part of its type. This is an example of why that is important: by suppressing background conditions such as $\delta > 0$, we make it much easier for the program not to pay undue attention to them, and therefore easier for us to define our priority order in a unified way.

At this point, the program is trying to prove a statement that existentially quantifies over δ . The nuclear option would be to convert the variable δ to a metavariable, but this operation has a low priority, so the program does as much forwards reasoning as it possibly can before resorting to it. It begins with `elementaryExpansionOfHypothesis`, applied to the third assumption.

$$\begin{array}{l}
A \text{ is open} \\
B \text{ is open} \\
x \in A \\
x \in B \\
\hline
\exists \delta \forall y (d(x, y) < \delta \Rightarrow y \in A \cap B)
\end{array}$$

This allows it apply forwardsReasoning twice. After the first application, the goal is as follows.

$$\begin{array}{l}
A \text{ is open} \\
B \text{ is open} \\
x \in A \\
x \in B \\
\forall u (d(x, u) < \eta[x] \Rightarrow u \in A) \\
\hline
\exists \delta \forall y (d(x, y) < \delta \Rightarrow y \in A \cap B)
\end{array}$$

Note that the last assumption is in a sense generated by a combination of subtactics: the first is forwardsReasoning (using the assumptions $x \in A$ and ‘ A is open’) and the second is an existential elimination (to get rid of $\exists \eta$ that would otherwise occur at the beginning of the statement). However, the latter is so automatic that it is not listed as one of our tidying tactics: instead, it is considered as part of any other tactic that potentially generates an existential assumption.

It is important to keep track of the fact that η depends on x , which is what is signified by $\eta[x]$.

After this, deleteUnmatchable causes the program to remove the statements $x \in A$ and ‘ A is open’. This is because both statements have been used and because it is no longer permissible to substitute x into ‘ A is open’. The resulting goal is as follows.

$$\begin{array}{l}
B \text{ is open} \\
x \in B \\
\forall u (d(x, u) < \eta[x] \Rightarrow u \in A) \\
\hline
\exists \delta \forall y (d(x, y) < \delta \Rightarrow y \in A \cap B)
\end{array}$$

The program then runs through a similar process for B (it does not yet have the capacity to recognise that the problem is symmetric in A and B and say, ‘Similarly ...’). After that process, it arrives at the following.

$$\begin{array}{l} \forall u (d(x, u) < \eta[x] \Rightarrow u \in A) \\ \forall v (d(x, v) < \theta[x] \Rightarrow v \in B) \\ \hline \exists \delta \forall y (d(x, y) < \delta \Rightarrow y \in A \cap B) \end{array}$$

It has now reached the point where conversion of δ to a metavariable is the only option it has. In the first instance, it uses the tactic `unlockExistentialUniversalConditionalTarget`. The result is as follows.

$$\begin{array}{l} \forall u (d(x, u) < \eta[x] \Rightarrow u \in A) \\ \forall v (d(x, v) < \theta[x] \Rightarrow v \in B) \\ \hline \boxed{\begin{array}{l} d(x, y) < \delta^\blacklozenge[\bar{y}] \\ \hline y \in A \cap B \end{array}} \end{array}$$

The notation $\delta^\blacklozenge[\bar{y}]$ signifies that δ is not allowed to depend on y .

The highest priority tactic the program can now apply is `elementaryExpansionOfTarget`, so it does that, and automatically splits the resulting conjunctive statement (rather than using the tactic `splitConjunctiveTarget`).

$$\begin{array}{l} \forall u (d(x, u) < \eta[x] \Rightarrow u \in A) \\ \forall v (d(x, v) < \theta[x] \Rightarrow v \in B) \\ \hline \boxed{\begin{array}{l} d(x, y) < \delta^\blacklozenge[\bar{y}] \\ \hline y \in A \\ y \in B \end{array}} \end{array}$$

This allows it to apply `backwardsReasoning` twice. After the two deductions it reaches the following state (It does them separately, so we are jumping a step here).

$$\begin{array}{l} \forall u (d(x, u) < \eta[x] \Rightarrow u \in A) \\ \forall v (d(x, v) < \theta[x] \Rightarrow v \in B) \\ \hline \boxed{\begin{array}{l} d(x, y) < \delta^\blacklozenge[\bar{y}] \\ \hline d(x, y) < \eta[x] \\ d(x, y) < \theta[x] \end{array}} \end{array}$$

It then uses `deleteUnmatchable` to remove the two assumptions it has just used.

$$\boxed{\begin{array}{c} d(x, y) < \delta^\blacklozenge[\bar{y}] \\ \hline d(x, y) < \eta[x] \\ d(x, y) < \theta[x] \end{array}}$$

At this point, there is not much that the program can do, because it is not allowed to reason with the diamonded variable δ^\blacklozenge . So the highest-priority tactic it can apply is `convertDiamondToBullet`. Also, since there are no assumptions above the main line, it replaces the goal by the inner box.

$$\begin{array}{c} d(x, y) < \delta^\bullet[\bar{y}] \\ \hline d(x, y) < \eta[x] \\ d(x, y) < \theta[x] \end{array}$$

Now it applies `backwardsLibraryReasoning`. The result in the library is that if $a < b$ and $b \leq c$, then $a < c$. Applying that with the assumption and the first target results in the following goal.

$$\begin{array}{c} d(x, y) < \delta^\bullet[\bar{y}] \\ \hline \delta^\bullet[\bar{y}] \leq \eta[x] \\ d(x, y) < \theta[x] \end{array}$$

The removal tactics do *not* allow the program to remove the assumption we have just used (and this is a good example of a situation where deletion would be a very bad idea). However, it cannot use the assumption with the new target. The program then uses `backwardsLibraryReasoning` again and this time it does remove the assumption, on the grounds that the variable x that appears in the assumption does not appear in any other statement. After that, it has reached the following state.

$$\begin{array}{c} \hline \delta^\bullet[\bar{y}] \leq \eta[x] \\ \delta^\bullet[\bar{y}] \leq \theta[x] \end{array}$$

This is a ‘standard’ existence problem, whose solution is stored as a construction in the library. The program uses this and declares the problem solved. It is here that the

background information that δ , η and θ are positive is used, since the library result is that the minimum of two positive real numbers a and b is a positive real number that is less than or equal to both a and b .

7 Testing the Write-Ups

Once the program had generated the write-ups for several problems, we wanted to test whether they could pass for write-ups written by a human mathematician. In this section we describe an informal experiment that we carried out for this purpose.

We began by asking two mathematicians, one an undergraduate and one a PhD student, to write out proofs for five problems for which our program had generated proof write-ups. We did not tell either of them why we were making this unusual request, and we did not ask them to make their write-ups as good as possible. One of the problems was to show that the inverse image of an open set under a continuous function is open, and one of our volunteers decided to prove the converse, so that he could use the topological definition of continuity to prove another of the assertions—that a composition of continuous functions is continuous. We had to ask him to rewrite the latter and give the epsilon-delta proof, since we wanted the differences between the write-ups to be a matter of style rather than substance.

We had another problem of this kind, which was that both our volunteers made frequent use of open balls. For example, their expansion of ‘ $A \cap B$ is open’ was ‘for every $x \in A \cap B$ there exists $\delta > 0$ such that $B_\delta(x) \subset A \cap B$.’ This made some of their arguments neater than the ones produced by our program. We contemplated getting the program to redo the problems using open-balls expansions, but in the end decided that it would be ‘cheating’ to make changes to its output in response to the human write-ups we had solicited, so we left things as they were.

The program’s write-ups were not designed to be indistinguishable from human write-ups: we merely wanted them to be acceptable as human write-ups. Therefore, we left in certain features, such as ending every proof with the words, ‘we are done’, that we could with a little trouble have changed (See the brief discussion of non-determinism at the end of Sect. 4). For this reason, we did not want to ask people to guess which write-ups were by the program. Instead, we presented all fifteen write-ups—two by humans and one by the program for each of the five problems—on the second author’s blog, and asked readers of the blog to comment on them in any way they liked. We also asked them to award points for clarity and style. The orders of the write-ups were chosen randomly and independently (The precise mechanism was to decide on a one-to-one correspondence between the set $\{1, 2, 3, 4, 5, 6\}$ to the set of permutations of the set $\{1, 2, 3\}$, then to find a website that produced random dice rolls). So that answers would be as independent as possible, all comments and ratings were sent to the blog’s moderation queue and published only after the experiment was finished and comments on the blog post were closed.

The post can be found at <http://gowers.wordpress.com/2013/03/25/an-experiment-concerning-mathematical-writing/>, together with all the comments and ratings, but the real point of the experiment was to see whether anybody noticed that not all the write-ups were by humans. Nobody expressed the slightest suspicion of this kind.

Having said that, we should also remark that many commenters were highly critical of the program's output. Three criticisms in particular stand out. First, as we expected, the fact that the program did not use open balls was unpopular: many people commented that this made the write-ups unwieldy. Secondly, several of the human write-ups stated the new target when the initial one had been stripped of universal quantifiers and conjunctions. Several readers commented that they found this helpful, and criticized our program for not doing it. And thirdly, commenters did not like the way the program spelt out in detail how it thought of the right variable to substitute into existential targets (such as choosing $\min\{\eta, \theta\}$ for δ in the intersection-of-open-sets problem).

It would be easy to modify the program so that none of these criticisms apply, so they do not point to fundamentally non-human aspects of how it thinks. To change the first, we would just have to use a library containing open-balls expansions of definitions such as 'A is open' and 'f is continuous'. To change the second, we could alter the rule for what the write-up does when we remove quantifiers and conjunctions, so that it states the new target (preceded by a phrase such as 'We need to show that'). The third criticism would be harder to deal with, but in future versions, we plan to switch to having two styles of write up: a 'proof write-up' and a more detailed 'proof-discovery account'. For the first style we will let the program work out the values of bulleted variables, then simply declare those values when the variable is first mentioned after being converted to a metavariable. This will correspond to the human practice of writing something like 'Let $\delta = \min\{\eta, \theta\}$ ' or 'Consider the sequence (b_n) defined by $b_n = a_n/(1 + a_n)$,' which 'magically' does exactly what it needs to do later in the proof.

Although our program's output came in for quite a bit of criticism, so did the write-ups by the undergraduate and PhD student—it seems that the readers were harsh judges. However, for most of the problems, the human write-ups were found preferable to the program's.

After the success (as we considered it) of this experiment, we dared to try a direct test. We published a new post, this time explaining that one proof was by a program, one by an undergraduate and one by a PhD student, and inviting readers to vote on which one they thought was by the program. For each problem, the write-ups were numbered (a), (b) and (c). There were seven options for the voting: one could choose between (a), (b) and (c), but also choose between 'The computer-generated output is definitely *' and 'I think the computer-generated output is * but am not certain'; the seventh option was 'I have no idea which write-up was computer generated.' Again there was the opportunity to comment, for those who wanted to explain the reasons for their choices.

We did not reveal the results of the voting so far, or anybody's comments, until the experiment was ended and the voting was closed. However, there was a different

kind of dependence between answers, which was that people had the opportunity to look for clues that two different write-ups were from the same source. Given that we had not tried to remove stylistic ‘tics’ from our program’s write-ups, this put the program at a significant disadvantage. It was clear from the comments that many people had noticed that for each problem exactly one write-up ended with the words ‘we are done’.

Despite this, the program did reasonably well at fooling people that it was human. The typical pattern was that roughly half the voters would correctly guess which output was by the program, with slightly under half of that half saying that the output was definitely by the program. The undergraduate would always ‘come second’, and there would always be a fair number of people who said that they had no idea which output was written by the computer. There were surprisingly many votes for ‘The computer-generated output is definitely *,’ when * was the wrong answer. The total number of votes was always at least 300, and for the first problem listed (the intersection of open sets is open) it was over 1000. One slight complication was that after a day or two the post was listed on the front page of Hacker News. The result was that the number of votes doubled in a couple of hours, and it may be that the profile of a typical voter changed. Fortunately, we had noted down the voting numbers just before this happened, so we presented those results as well as the final numbers. In the end, however, the proportions did not change very much. The detailed numbers can be found here: <http://gowers.wordpress.com/2013/04/14/answers-results-of-polls-and-a-brief-description-of-the-program/>.

One thing this experiment could not tell us, except to a limited extent through the comments, was whether the program was good at fooling *mathematicians* that it was human. It could be that the more mathematically experienced readers found the program’s output easy to distinguish, while the votes for the human write-ups came from people who were not used to reading mathematical proofs. However, we feel justified in concluding that the program’s output is not *obviously* written by a computer program, and that was our objective.

8 Running the Program

The prover was written in Haskell, and contains about 3300 lines of source code. Readers who wish to replicate the evaluation or try the prover on other problems can obtain the source code at <https://github.com/mg262/research/raw/master/robotone.zip>; the readme file inside the archive contains instructions on compiling and running the prover. Note that although the problems and library are logically separated from the rest of the program, they are currently stored as pure data in a Haskell module

and compiled with the rest of the code.³ Output is produced as L^AT_EX source which is then compiled to two human-readable PDFs; one of these simply contains the proofs, and the other displays the step-by-step operation of the program with goals displayed at each stage.

Note that the shell script that invokes the prover also runs L^AT_EX on its output, and that this accounts for nearly all of its running time; the actual Haskell program runs too fast to measure (<1 ms) on the eight test problems included with the source code. This speed is a consequence of our aim of solving routine problems without backtracking or extensive search, just as a human does (Sect. 1.3).

Readers who wish to try the prover on other problems should be warned that the library *must* be tailored to the problem being solved.⁴ It is not possible to create a general, problem-independent library (without significantly modifying the program) because then the prover will use “more advanced” results to prove simpler ones. For example, if one were simply to fill a library with every available result about real analysis and then ask the prover to show that sin is continuous, it could well deduce this from the fact that sin is differentiable and the fact that differentiable functions are continuous. But this is clearly an absurd proof.

This point may be illustrated with a problem tried by a referee, namely to show that a preimage of a closed set under a continuous function is closed. This problem was tried with the default library, which does not contain the requisite body of facts about sequences. In particular, the program contains the expansion

$$\text{in}(x, \text{preimage}(f, U)) \text{ --> } \text{in}(\text{applyfn}(f, x), U)$$

which allows ‘ $x \in f^{-1}(U)$ ’ to be expanded into ‘ $f(x) \in U$ ’. As with many other expansions, this rule has a direct analogue for sequences (and one for families, one for sets, etc.). Once that rule,

$$\text{sequencein}(x, \text{preimage}(f, U)) \text{ --> } \text{sequencein}(\text{applyfnpointwise}(f, x), U)$$

which allows ‘ $(a_n) \in f^{-1}(U)$ ’ to be expanded into ‘ $f((a_n)) \in U$ ’, has been added to the library, the prover produces a solution:

Let (a_n) and a be such that (a_n) is a sequence in $f^{-1}(U)$ and $a_n \rightarrow a$. Then $f(a_n)$ is a sequence in U . We would like to show that $a \in f^{-1}(U)$, i.e. that $f(a) \in U$ and since U is closed, $f(a) \in U$ if $f(a_n) \rightarrow f(a)$. Let $\epsilon > 0$. We would like to find N s.t. $d(f(a), f(a_n)) < \epsilon$ whenever $n \geq N$. Since f is continuous, there exists $\delta > 0$ such that $d(f(a), f(a_n)) < \epsilon$ whenever $d(a, a_n) < \delta$. Since $a_n \rightarrow a$, there exists N' such that $d(a, a_n) < \delta$ whenever $n \geq N'$. Therefore, setting $N = N'$, we are done.

³Using a Haskell module has allowed us to leverage Haskell’s excellent type-checking system to validate much of the input, and has also made it easy to construct specific libraries using higher-order operations during testing, with a considerable reduction in redundancy relative to a text format.

⁴All of the data used by the prover, including the library used with the supplied problems, can be found in TestData.hs.

This solution is unsatisfactory in that it is operating at too low a level by re-proving from scratch the fact that a continuous function preserves limits. Adding that result to the library gives a more satisfactory proof:

Let (a_n) and a be such that (a_n) is a sequence in $f^{-1}(U)$ and $a_n \rightarrow a$. Then $f(a_n)$ is a sequence in U . We would like to show that $a \in f^{-1}(U)$, i.e. that $f(a) \in U$. Since f is continuous and $a_n \rightarrow a$, we have that $f(a_n) \rightarrow f(a)$. Therefore, since U is closed and $f(a_n)$ is a sequence in U , we have that $f(a) \in U$ and we are done.

Note that if the library contains the fact that a continuous function preserves limits, then the prover will generate a trivial proof when asked to prove that fact.

In cases where the program fails to solve a problem, the most likely cause is that the supplied library is not appropriate. Examining the final goal presented in the detailed output of the program usually makes it clear what fact(s) one has forgotten to include. Note that this is a benefit of our strategy of not backtracking: there is a definite single final state in which the program is 'stuck', and examining that state is invariably helpful.

9 Future Work

In the short term, we would like to make a number of small improvements to the program so that it handles a greater range of problems satisfactorily. In the longer term, we would like to enlarge significantly the set of problems that our program, or some new version of it, is capable of solving. To do this, we will have to enable the program to handle certain kinds of deductions that it currently handles either not at all or only in a rather rudimentary way. In particular, an immediate target is to give the program the means to deal with second-order quantification, which would allow it to solve easy compactness problems, and also problems that require the construction of 'obvious' sequences.

At a more basic level, the program does not currently solve problems that involve proof by contraposition or contradiction. It is not hard to add tactics that allow it to cope with a few problems of this kind, but it is trickier to do so while not letting it apply those tactics in inappropriate contexts. More work is needed to understand what triggers the 'contradiction move' in human mathematicians, but we expect to be able to add this facility in the near future.

The program is also not as good as we would like at handling equality substitutions. The situation here is similar: we can obviously add tactics that perform such substitutions (and have done so in the current version of the program), but it is more challenging to understand when humans make such substitutions. It is also tricky to come up with a general understanding of how they choose which out of two equal variables or complex terms to eliminate. At its most general, the problem of how to handle equality is well known to be hard, but our immediate aim would be a program that can handle the easy cases of that problem competently and in a human way.

Related to this, we need to do more work in order to enable the program to solve problems that require the arithmetic structure of the real numbers, rather than just the order structure. For example, the prover does not yet solve problems such as showing that the limit of the sum of two convergent sequences is the sum of the limits.

In the longer term, we would of course like the program to be able to solve non-routine problems. A major step up in problem-solving sophistication is needed when one is required to carry out mathematical constructions, especially when they are far from unique. This is true even for very easy problems. Consider for example the problem of finding an infinite set of positive integers that contains no three distinct numbers x , y and z with $x + y = z$. One obvious example is to take the set of all odd numbers. Another that works for a different reason is to take the set of all powers of 2. Yet another, $\{1, 2, 4, 7, 12, 20, \dots\}$ is obtained by taking each new element to be one more than the sum of the two largest elements so far. All these examples feel like ones that a human might conceivably come up with in response to the problem. We have ideas about how these kinds of simple (for humans) existence proofs are discovered, but implementing those ideas in a program will be a great deal of work.

Acknowledgements Research supported by a Royal Society 2010 Anniversary Research Professorship.

References

- Asher, N., & Lascarides, A. (2003). *Logics of conversation*. Cambridge: Cambridge University Press.
- Ballantyne, A. M., & Bledsoe, W. W. (1977). Automatic proofs of theorems in analysis using non-standard techniques. *Journal of ACM*, 24(3), 353–374.
- Beeson, M. (1998). Automatic generation of epsilon-delta proofs of continuity. In J. Calmet & J. A. Plaza (Eds.), *Proceedings of the International Conference on Artificial Intelligence and Symbolic Computation (AISC '98)* (pp. 67–83). London, UK: Springer.
- Beeson, M. (2001). Automatic derivation of the irrationality of e . *Journal of Symbolic Computation*, 32(4), 333–349.
- Bledsoe, W. W. (1971). Splitting and reduction heuristics in automatic theorem proving. *Artificial Intelligence*, 2(1), 55–77.
- Bledsoe, W. W. (1977a). Non-resolution theorem proving. *Artificial Intelligence*, 9(1), 1–35.
- Bledsoe, W. W. (1997b). Set variables. In: *Proceedings of the 5th International Joint Conference on Artificial Intelligence* (Vol. 1, pp. 501–510). San Francisco, CA: Morgan Kaufmann Publishers Inc. <http://dl.acm.org/citation.cfm?id=1624548>.
- Bledsoe, W. W. (1983). Using examples to generate instantiations for set variables. *Proceedings of IJCAI*, 83, 892–901.
- Bledsoe, W. W. (1995). A precondition prover for analogy. *BioSystems*, 34(1), 225–247.
- Bledsoe, W. W., Boyer, R. S., & Henneman, W. H. (1972). Computer proofs of limit theorems. *Artificial Intelligence*, 3, 27–60.
- Bledsoe, W. W., & Hodges, R. (1988). A survey of automated deduction. In H. Shrobe (Ed.), *Exploring artificial intelligence* (pp. 483–541). San Mateo, CA: Morgan Kaufmann Publishers Inc.
- Boyer, R. S., & Moore, J. S. (1979). *A computational logic*. ACM Monograph, Cambridge: Academic Press.

- Buchberger, B., Crăciun, A., Jebelean, T., Kovács, L., Kutsia, T., Nakagawa, K., et al. (2006). Theorema: Towards computer-aided mathematical theory exploration. *Journal of Applied Logic*, 4(4), 470–504.
- Bundy, A. (2011). Automated theorem provers: A practical tool for the working mathematician? *Annals of Mathematics and Artificial Intelligence*, 61(1), 3–14.
- Clarke, E., & Zhao, X. (1992). *Analytica—A theorem prover in Mathematica*. Berlin: Springer.
- Felty, A., & Miller, D. (1987). *Proof explanation and revision*. Technical Report MS-CIS-88-17, University of Pennsylvania.
- Ganesalingam, M. (2013). *The language of mathematics*. Berlin: Springer.
- Gonthier, G. (2019). A computer-checked proof of the four colour theorem. <http://research.microsoft.com/en-US/people/gonthier/4colproof.pdf>.
- Gonthier, G., Asperti, A., Avigad, J., Bertot, Y., Cohen, C., Garillot, F., et al. (2013). A machine-checked proof of the odd order theorem. In S. Blazy, C. Paulin-Mohring & D. Pichardie (Eds.), *Interactive theorem proving* (pp. 163–179). Berlin: Springer.
- Grice, H. P. (1975). Logic and conversation. In P. Cole & J. L. Morgan (Eds.), *Syntax and semantics* (Vol. 3, pp. 41–58). New York: Academic Press.
- Hales, T., Adams, M., Bauer, G., Dang, D.T., Harrison, J., Hoang, T.L., et al. (2015). A formal proof of the Kepler conjecture. <http://arxiv.org/abs/1501.02155>.
- Holland-Minkley, A. M., Barzilay, R., Constable, R. L. (1999). Verbalization of high-level formal proofs. In *Proceedings of Sixteenth National Conference on Artificial Intelligence* (pp. 277–284).
- Humayoun, M., & Raffalli, C. (2010). MathNat—Mathematical text in a controlled natural language. In *Special issue: Natural Language Processing and its Applications* (Vol. 46, pp. 293–307).
- Knott, A. (1996). A data-driven methodology for motivating a set of coherence relations. Ph.D. thesis, University of Edinburgh.
- Kuhlwein, D., Cramer, M., Koepke, P., & Schröder, B. (2009). The Naproche system. <http://www.naproche.net/downloads/2009/emergingsystems.pdf>.
- Mancosu, P. (Ed.) (2008). Mathematical explanation: Why it matters. In *The Philosophy of Mathematical Practice* (pp. 134–149). Oxford: Oxford University Press.
- Owen, E., & Sweller, J. (1985). What do students learn while solving mathematics problems? *Journal of Educational Psychology*, 77(3), 272–284.
- Paulson, L. C. (1987). *Logic and computation: Interactive proof with Cambridge LCF*. Cambridge: Cambridge University Press.
- Reiter, E., & Dale, R. (2000). *Building natural language generation systems*. Cambridge: Cambridge University Press.
- Sweller, J., Mawer, R. F., & Ward, M. R. (1983). Development of expertise in mathematical problem solving. *Journal of Experimental Psychology: General*, 112(4), 639–661.
- Trybulec, A. (1978). The Mizar-QC/6000 logic information language. *ALLC Bulletin (Association for Literary and Linguistic Computing)*, 6(2), 136–140.
- Vershinin, K., & Paskevich, A. (2000). Forthel—The language of formal theories. *International Journal of Information Theories and Applications*, 7(3), 120–126.

A Common Type of Rigorous Proof that Resists Hilbert's Programme



Alan Bundy and Mateja Jamnik

1 Introduction

Hilbert's Programme is defined in (Zach, 2009, p. 9) as:

The main goal of Hilbert's program was to provide secure foundations for all mathematics. In particular this should include:

- A formalization:** of all mathematics; in other words all mathematical statements should be written in a precise formal language, and manipulated according to well defined rules.
- Completeness:** a proof that all true mathematical statements can be proved in the formalism.
- Consistency:** a proof that no contradiction can be obtained in the formalism of mathematics. This consistency proof should preferably use only "finitistic" reasoning about finite mathematical objects.
- Conservation:** a proof that any result about "real objects" obtained using reasoning about "ideal objects" (such as uncountable sets) can be proved without using ideal objects.
- Decidability:** there should be an algorithm for deciding the truth or falsity of any mathematical statement.

The problems with the goals of Completeness, Consistency and Decidability, that were revealed by Gödel's incompleteness theorems (Gödel, 1931), have been well documented, but in this chapter, we are focused on the goal of Formalisation.

A. Bundy (✉)
University of Edinburgh, Edinburgh, UK
e-mail: A.Bundy@ed.ac.uk

M. Jamnik
University of Cambridge, Cambridge, UK
e-mail: mateja.jamnik@cl.cam.ac.uk

Formalisation has become important in Computer Science as the basis of *automated theorem proving*, which has important practical applications, for instance, in the verification of the correctness of computer software and hardware (Robinson & Voronkov, 2001). Using one of many formal logics, the axioms and rules of inference of a formal mathematical theory are represented as data-structures in an automated theorem prover. Programs are then written to construct formal proofs by deriving new formulae from old by applying these rules to them. In some provers, humans can interact and guide the development of the proof; in others, the development is completely automatic.

Zach (2009, p. 10) summarises the post-Gödel, modified form of the Formalisation goal of Hilbert's Programme as:

Although it is not possible to formalize all mathematics, it is possible to formalize essentially all the mathematics that anyone uses. In particular Zermelo-Fraenkel set theory, combined with first-order logic, gives a satisfactory and generally accepted formalism for essentially all current mathematics.

Our claim in this chapter is that there is a form of proof, which we will call schematic proof, that resists even this, generally accepted, Formalisation goal of the Hilbert Programme.

As evidence to support our claim we will analyse Cauchy's Proof of Euler's Theorem as discussed, for instance, in Lakatos (1976). Paraphrasing Hilbert (1930), we will take Hilbert's view of proof to be:

A proof is a sequence of formulae each of which is either an axiom or follows from earlier formulae by a rule of inference.

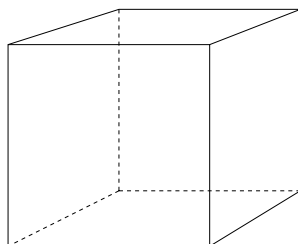
There is a widespread assumption in the Mathematics community that the rigorous¹ proofs one finds in Mathematics papers and textbooks are an abstraction of this ideal. Instead of stating every formula in the sequence, some of them are omitted, the assumption being that a typical reader will be able to skip several formal steps at a time. De Bruijn's factor even gives a numerical value of approximately 4 to the typical ratio of the logical to the rigorous proof steps (de Bruijn, 1980). This was calculated by surveying and analysing a number of proofs produced in different automated theorem provers. Carrying out this formalisation part of Hilbert's Programme then consists of filling in the gaps: agreeing on an axiomatic system, such as Zermelo-Fraenkel set theory, deriving all the elementary lemmas that might be assumed in a proof, and interleaving the rigorous proof with the missing steps.

We will show that schematic proofs contradict this assumption and that a much more radical programme is required to formalise them—provided they are not faulty.

Schematic proofs are of interest to Mathematics educators for both, positive and negative reasons.

- On the positive side, they are more intuitive, natural and accessible, and can engender a deeper understanding of why a theorem holds than a Hilbertian proof can.

¹The word 'informal' is sometimes used instead of 'rigorous'. We avoid 'informal', as we will claim, in Sect. 4, that schematic proofs can also be formalised.



In a cube there are 8 vertices, 12 edges and 6 faces. So, $V - E + F = 8 - 12 + 6 = 2$.

Fig. 1 Euler's Theorem in the case of the cube

- On the negative side, they are error prone. A proof of their correctness for all cases is required, but is often omitted. If omitted, counter-examples can go undetected.

2 Cauchy's 'Proof' of Euler's Theorem

Polyhedra are the 3D version of polygons: objects whose faces are polygons. Examples include the five regular Platonic polyhedra: tetrahedron, cube, octahedron, dodecahedron and icosahedron, as well as many other semi-regular and regular objects. Euler's 'Theorem'² states that in any polyhedron, the following holds, $V - E + F = 2$, where V is the number of vertices, E the number of edges and F the number of faces. Figure 1 illustrates this 'theorem' in the case of the cube.

In (1976), Imre Lakatos uses a rational reconstruction of the history of Euler's 'Theorem' as a vehicle to argue that the methodology of mathematics had evolved and become more sophisticated. He describes a fictional classroom in which a teacher leads a (very bright!) class through this history. The teacher starts by presenting Cauchy's 'proof'³ of the theorem. The students then confront it with various counter-examples and suggest ways to cope with them.

Cauchy's 'proof' is couched as the following 'thought experiment', quoted from Lakatos (1976, pp. 7–8):

- “Step 1:** Let us imagine the polyhedron to be hollow, with a surface made of thin rubber. If we cut out one of the faces, we can stretch the remaining surface flat on the blackboard, without tearing it. The faces and edges will be deformed, the edges may become curved, but V and E will not alter, so that if and only if $V - E + F = 2$ for the original polyhedron, $V - E + F = 1$ for this flat network—remember we have removed one face (Fig. 2 top left shows the flat network for the case of a cube).”

²The scare quotes indicate that there are issues with proving Euler's 'Theorem'. These issues are the subject of this section. It would have been more accurate to call it 'Euler's Conjecture'.

³Again, the scare quotes indicate that there are issues with this alleged proof.

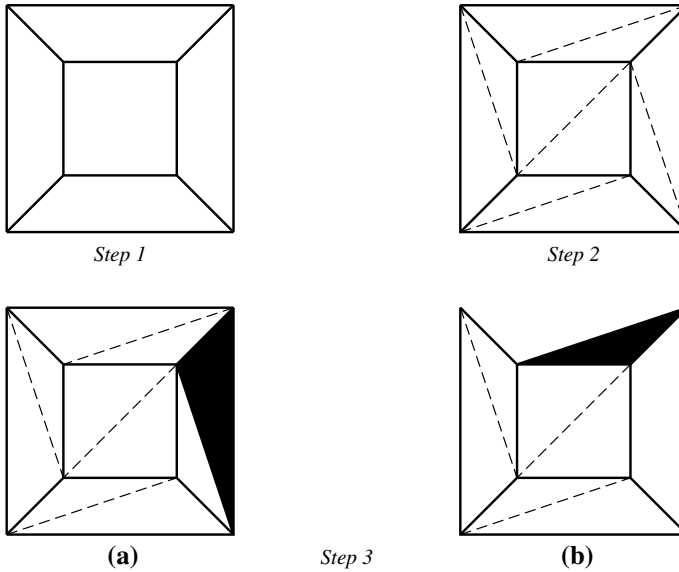


Fig. 2 Cauchy’s ‘proof’ applied to the cube

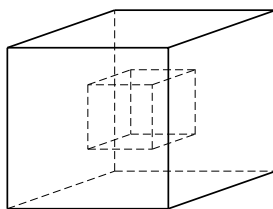
“Step 2: Now we triangulate our map—it does indeed look like a geographical map. We draw (possibly curvilinear) diagonals in those (possibly curvilinear) polygons which are not already (possibly curvilinear) triangles. By drawing each diagonal we increase both, E and F by one, so that the total $V - E + F$ will not be altered (Fig. 2 top right).”

“Step 3: From the triangulated network we now remove the triangles one by one. To remove a triangle we either remove an edge—upon which one face and one edge disappear (Fig. 2 bottom left), or we remove two edges and a vertex—upon which one face, two edges and one vertex disappear (Fig. 2 bottom right). Thus if $V - E + F = 1$ before a triangle is removed, it remains so after the triangle is removed. At the end of this procedure we get a single triangle. For this $V - E + F = 1$ holds true. Thus we have proved our conjecture.”

The bulk of Lakatos (1976) consists of the presentation of various counter-examples to Euler’s ‘Theorem’, followed by discussions of how these can be dealt with either by ruling them out as polyhedra or adapting the proof and/or theorem. Figures 3 and 4 give four such counter-examples.

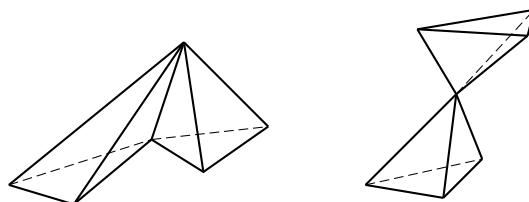
3 Schematic Proof

Cauchy’s ‘proof’ in Sect. 2, quoted from Lakatos (1976, pp. 7–8), has some unusual properties. It is an example of what we will call *schematic proof*. The hypothesis of this chapter is that:



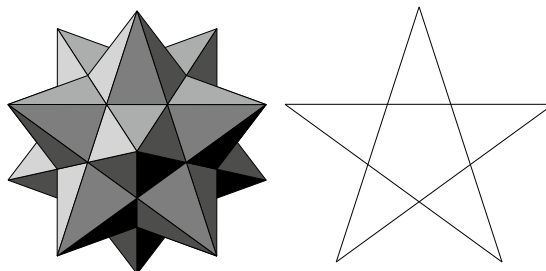
The hollow cube is a cube with a cubical hole in the middle. The values of V , E and F are all doubled. So $V - E + F = 16 - 24 + 12 = 4$.

Fig. 3 The hollow cube: a counter-example to Euler’s ‘Theorem’



$$V - E + F = 5 - 10 + 8 = 3$$

$$V - E + F = 7 - 12 + 8 = 3$$



$$V - E + F = 12 - 30 + 12 = -6$$

Fig. 4 Three more counter-examples to Euler’s Theorem

Schematic proofs resist Hilbert’s Programme to unpack rigorous proofs into logical ones.

These unusual properties are:

1. The ‘proof’ is not Hilbertian, that is, it is not a sequence of formulae each of which is either an axiom or follows from earlier formulae by a rule of inference. Rather, it is a *procedure*, which is to be applied to any polyhedron, during which a property ($V - E + F = 1$) remains invariant.

2. This procedure will produce a different sequence of steps for each polyhedron, for example, when applied to a tetrahedron, the ‘map’ produced by step 1 will consist of three triangles, in contrast to the five quadrilaterals produced for a cube. Step 2 will then not be necessary and there will be fewer applications of step 3. Indeed, we can see the procedure as generating a *different* proof for each polyhedron.
3. The ‘proof’ is error prone, that is, some steps either cannot be applied to some polyhedra, or they produce unanticipated effects. There is no attempt to prove that the procedure will apply to all polyhedra, or that it will output a proof of Euler’s ‘Theorem’ for each polyhedron.
4. The ‘proof’ is carried out in the absence of any definition of polyhedra. In a Hilbertian proof, definitions must precede proofs. The absence of a definition quickly becomes apparent. In the argument as to whether the hollow cube (Fig. 3) is a polyhedron, two rival definitions are proffered (Lakatos, 1976, p. 14). In one, a polyhedron is a solid, and the hollow cube is a polyhedron. In the other, a polyhedron is a collection of surfaces, and the hollow cube falls apart into two nested polyhedra. Even these definitions are not sufficient, as they still contain undefined terms, such as ‘surface’.
5. The ‘proof’ was accepted for some time before counter-examples were discovered. Even then, according to Lakatos’ account, there was considerable confusion about exactly what was wrong with the ‘proof’ and how it should be fixed. How is this possible? Hilbertian proofs can be mechanically checked and any errors will quickly show up, for instance, a formula that does not follow by a rule of inference from earlier formulae. Even before the advent of computers, humans could be employed to check even a long proof.

The reason we call this ‘proof’ *schematic* is because of point 2 above. To present it as a single proof we need to abstract from the varying number of steps it produces. For this we need abstraction devices, such as ellipsis (cf. Fig. 6 in Sect. 5.1 and Fig. 10 in Sect. 5.2 below).

4 Formalisation of Schematic Proofs

We have argued that schematic proofs are not Hilbertian and that this presents an obstacle to their formalisation within the usual assumptions of Hilbert’s Programme, that is, the ‘filling in the gaps’ process outlined in Sect. 1. However, we will argue that it *is* possible to give a logical account of schematic proofs, albeit a more complex one. This logical account will use the *constructive ω -rule* (Shoenfield, 1959).

The ω -rule for the natural numbers 0, 1, 2, ... is:

$$\frac{\phi(0), \phi(1), \phi(2), \dots}{\forall x.\phi(x)}$$

Mathematical Induction	Philosophical Induction	ω -rule
$\frac{\phi(0), \forall n \phi(n) \implies \phi(n+1)}{\forall n. \phi(n)}$	$\frac{\phi(n_1), \phi(n_2) \dots \phi(n_m)}{\forall n. \phi(n)}$	$\frac{\phi(0), \phi(1) \phi(2) \dots}{\forall n. \phi(n)}$

Fig. 5 Difference between mathematical induction, philosophical induction and the ω -rule

that is, we can infer that $\phi(x)$ for all natural numbers x provided we can prove $\phi(n)$ for $n = 0, 1, 2, \dots$. The ω -rule is clearly not a very practical rule of inference, since it requires the proof of an infinite number of premises to prove its conclusion. A Hilbertian proof using it would consist of an infinite sequence of formulae. Its use is usually confined to theoretical discussions, for instance, in Gödel’s incompleteness theorems (Gödel, 1931).

The constructive ω -rule is a refinement of the ω -rule that *can* be used in practical proofs. It has the additional requirement that the $\phi(n)$ premises be proved in a *uniform* way, that is, that there exists an effective procedure, $proof_\phi$, which takes a natural number n as input and returns a proof of $\phi(n)$ as output. We will write this as $proof_\phi(n) \vdash \phi(n)$. The procedure $proof_\phi$ formalises our notion of schematic proof. Applied to the domain of polyhedra, rather than natural numbers, it could be used to formalise Cauchy’s ‘proof’ of Euler’s ‘theorem’ given in Sect. 2. In particular, the procedure, which given a polyhedron, proves Euler’s ‘theorem’ for it, can be interpreted as the procedure $proof_\phi$, but for polyhedra rather than natural numbers.

The constructive ω -rule has been automated to generate schematic proofs in the domain of natural numbers. Siani Baker’s Ph.D. thesis (Baker, 1993) automated the schematic proofs of theorems of Peano Arithmetic. Mateja Jamnik’s Ph.D. thesis (Jamnik, 2001) automated the proofs in Nelsen (1993) (see Sect. 5.1). These automated provers start with a theorem to prove and one or two instances of the proof of this theorem for particular numbers: essentially just calculations of concrete instances of the theorems. These proof instances are then generalised to a procedure $proof_\phi$.

Both of these automated provers then do something omitted in Cauchy’s ‘proof’: the meta-proof that the procedure generated a proof for each $\phi(n)$. This is done by mathematical induction⁴ on the natural numbers, that is, the proof that:

$$\begin{aligned}
 &proof_\phi(0) \vdash \phi(0) \\
 &proof_\phi(n) \vdash \phi(n) \implies proof_\phi(n+1) \vdash \phi(n+1)
 \end{aligned}$$

Not only was a comparable meta-proof omitted by Cauchy, it’s hard to see how he could have provided it. The natural numbers have a recursive structure, which lends itself to inductive proof. The set of polyhedra has no known recursive structure, so a similar proof plan is not available. As we have seen in Sect. 2, the absence of this meta-proof is a fatal flaw. It allows the possibility that there is a polyhedron *poly*

⁴Not to be confused with, the regrettably similarly named, *philosophical induction*, which is a rule of conjecture, not deduction, or with the ω -rule. See Fig. 5.

say, for which:

$$\text{proof}_\phi(\text{poly}) \not\vdash \phi(\text{poly})$$

where ϕ is Euler's 'Theorem'. As we saw in Sect. 2, there are many such polyhedra. This illustrates the negative side of schematic proof, which we mentioned at the end of Sect. 1, that educators need to be aware of.

The situation with the extraction of schematic proofs is somewhat reminiscent of the way that computer programs are typically developed. Programmers start with a few development examples of the input/output relationship that the program is intended to exhibit, and the series of steps it should go through in these concrete instances. They then generalise from the steps for concrete instances into a general procedure. The generalised procedure is then tested on further examples. This testing can only cover a finite subset of what is usually a potentially infinite set. So, there is no guarantee that the procedure will not subsequently fail, especially on a kind of example that the programmers did not think of. Ideally, the programmers would verify the correctness of the program via the kind of meta-proof discussed above. In Computing, this is called a *verification proof*. Unfortunately, verification proofs require a high skill level and are time consuming. As a result, they are rarely undertaken, except in highly safety or security critical applications. When they are undertaken, then they, like Cauchy's 'proof', often consist of showing that some invariant is preserved at each step of the procedure. This invariant might, for instance, be a safety or security property, or it may assert the absence of deadlock or of other kinds of error.

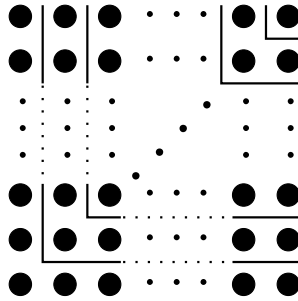
5 Schematic Proofs are Common

Our claim in Sect. 1, that schematic proofs resist Hilbert's Programme, would have little force if the set of schematic proofs was very small, for example, consisting only of Cauchy's faulty proof. In this section, we argue that they are commonplace. We will present two forms of evidence:

- Nelsen's "Proofs without words" (Nelsen, 1993); and
- An analysis of human reasoning about recursive programs (Jamnik & Bundy, 2005; Fugard, 2005).

5.1 Nelsen's Proofs Without Words

In (1993), Roger Nelsen shows how the truth of a mathematical theorem can often be convincingly demonstrated with a well chosen diagram. Some of these theorems are about the natural numbers, one of which is illustrated in Fig. 6. It is these proofs that Mateja Jamnik automated in Jamnik (2001) using the constructive ω -rule. Her



The diagram gives a proof of the theorem $n^2 = 1 + 3 + \dots + (2n - 1)$. The diagram can be viewed as describing both, the left and right hand sides of the equation. The whole square represents n^2 . Each of the L-shapes represents one of the odd numbers summed on the right-hand side.

Fig. 6 A proof without words

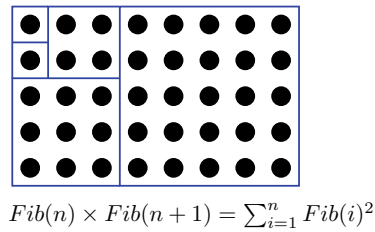
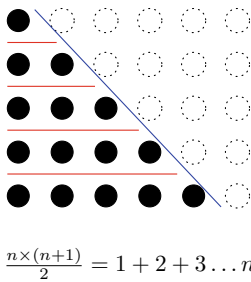


Fig. 7 Further examples of proofs without words

Diamond program used the procedure outlined in Sect. 4. That is, a proof-generating procedure was abstracted from the proofs of a couple of special cases, say $n = 3$ and $n = 4$, then a meta-proof showed that this procedure would generate a correct proof for each natural number n . The success of her project shows that some of Nelsen’s proofs without words can be characterised as schematic. Further examples of such schematic proofs are given in Fig. 7.

Nelsen’s diagrammatic proofs illustrate the positive side of schematic proofs that we argued at the end of Sect. 1. Many educators have long found them an intuitive and accessible alternative to Hilbertian proofs.

5.2 Human Use of Schematic Proof

In his M.Sc. project (Fugard, 2005), Andy Fugard reported on some experiments to observe human reasoning about recursive programs. Participants were presented with conjectures, some of which were true and some false. For instance, he asked participants about the (true) *rotate-length conjecture*:

$$\text{rot}(\text{len}(l), l) = l \tag{1}$$

where l is a list, such as $[a, b, c]$, $\text{len}(l)$ is the length of the list, say 3, and $\text{rot}(n, l)$ rotates n times the list l , where one rotation is to move the first element to the end of the list, for example $[a, b, c]$ to $[b, c, a]$. The formal recursive definition of rot is:

$$\begin{aligned} \text{rot}(0, l) &= l \\ \text{rot}(n + 1, []) &= [] \\ \text{rot}(n + 1, [h|t]) &= \text{rot}(n, \text{app}(t, [h])) \end{aligned}$$

where app appends one list to the end of another and $[h|t]$ is a list where h is the first element and t is the rest of the list. In Fugard's experiments the function names were replaced with arbitrary letters so that participants were not influenced by the meanings implied by names such as rot , len and app .

The Hilbertian proofs of theorems about recursive programs use mathematical induction. Fugard did not observe any of his participants attempting an inductive proof. Had they done so, they would have encountered a difficulty. The rotate-length conjecture (1) is surprisingly tricky to prove. It requires an intermediate lemma. For instance, the following generalised rotate-length conjecture works as an intermediate lemma:

$$\text{rot}(\text{len}(l), \text{app}(l, k)) = \text{app}(k, l) \tag{2}$$

where k is also a list. Lemma (2) is (surprisingly) easier to prove than (1) and, in fact, (1) is an immediate corollary from (2) by setting k to be the empty list. Fugard's participants, however, found it harder to see that (2) was true and harder to prove than (1).

The participants' reasoning often used diagrams consisting of a succession of lists being rotated, in which the last list was the same as the first. These lists often started as concrete ones for particular lists, but were then converted into schematic ones via the use of ellipsis, or similar abstraction devices. Diagrams often showed signs of alteration, where a concrete list was converted into a more schematic one. Figure 8 shows an example. These diagrams approximated the more formal accounts of a concrete and schematic proof given in Figs. 9 and 10, respectively.

One can speculate why people might prefer schematic proofs. Firstly, it's a common observation of Mathematics and Computing teachers that students find recursion and mathematical induction puzzling and difficult when they first encounter them. They are thought, for instance, to be circular: defining a function in terms of itself;

assuming the conclusion you want to prove. On the other hand, most people are familiar at an early age with philosophical induction (see Fig. 5) and the generalisation from concrete examples to a general one. They are used to developing general procedures to deal with new situations by generalising from successful concrete cases. It is, thus, very natural to apply this technique to mathematical problems—even though it is error prone.

As we argued at the end of Sect. 1, educators will, therefore, often find that students understand schematic proofs more readily than they do Hilbertian ones.

The contrast between the accessibility of schematic and Hilbertian proofs is acutely illustrated in Lakatos (1976, Chap. 2). It relates Poincaré’s Hilbertian proof of a special case of Euler’s ‘Theorem’. Not only is the proof extremely technical, but the uncertainty about its correctness is transferred from the proof itself to whether the encoding of polyhedra into incidence matrices is faithful.

6 Discussion

We have defined a common class of proofs, which we call *schematic*, that we claim resist the part of Hilbert’s Programme that asserts that all proofs can be readily formalised as a sequence of formulae that are either axioms or follow from earlier formulae by a rule of inference. Schematic proofs *can* be formalised—using the constructive ω -rule—but not as Hilbertian proofs. Rather, their formalisation consists of a procedure that generates a different Hilbertian proof for each concrete case, that is, different in the sense that each such concrete case consists of a different sequence of proof steps.

Moreover, schematic proofs are error prone in that they can yield unexpected counter-examples. Schematic proofs can be shown to be error free if a meta-proof is



A concrete diagram was first drawn on the left and then extended on the right using ellipsis and annotation to indicate generality.

Fig. 8 Example diagram drawn by one of Fugard’s participants for rotate-length conjecture

$$\begin{aligned}
 \text{rot}(\text{len}([a_1, a_2, a_3]), [a_1, a_2, a_3]) &= \text{rot}(\text{len}([a_2, a_3]), [a_2, a_3, a_1]) \\
 &= \text{rot}(\text{len}([a_3]), [a_3, a_1, a_2]) \\
 &= \text{rot}(\text{len}([\]), [a_1, a_2, a_3]) \\
 &= \text{rot}(0, [a_1, a_2, a_3]) \\
 &= [a_1, a_2, a_3]
 \end{aligned}$$

Fig. 9 A concrete proof of the rotate-length theorem for $n = 3$

$$\begin{aligned}
\text{rot}(\text{len}([a_1, a_2, a_3, \dots, a_n]), [a_1, a_2, a_3, \dots, a_n]) &= \text{rot}(\text{len}([a_2, a_3, \dots, a_n]), [a_2, a_3, \dots, a_n, a_1]) \\
&= \text{rot}(\text{len}([a_3, \dots, a_n]), [a_3, \dots, a_n, a_1, a_2]) \\
&\vdots \\
&= \text{rot}(\text{len}([]), [a_1, a_2, a_3, \dots, a_n]) \\
&= \text{rot}(0, [a_1, a_2, a_3, \dots, a_n]) \\
&= [a_1, a_2, a_3, \dots, a_n]
\end{aligned}$$

Fig. 10 A schematic proof of the rotate-length theorem

conducted to prove that the proof generation procedure produces correct proofs for all inputs. Unfortunately, this part of the process is often omitted. Also, schematic proofs can be conducted in the absence of definitions that would be crucial in a Hilbertian context. This is possible because, apart from the meta-proof, schematic proving is just based on an analysis and generalisation of the concrete proofs of a few examples. It is possible to agree that these are examples of a concept without formally defining that concept. If the meta-proof is omitted, as it often is, the issue of definitions only arises when potential counter-examples are discovered, and it is necessary to decide whether they are really examples of the concept.

The formalisation part of Hilbert's Programme has, here-to-fore, largely avoided the criticism that the Completeness, Consistency and Decidability goals have attracted because of Gödel's incompleteness theorems. We have argued that this Formalisation goal of the Programme is also not as straightforward as it is sometimes assumed to be.

Acknowledgements The research reported in this chapter is based on Bundy et al. (2005), Jamnik and Bundy (2005), Bundy (2012). It was supported by EPSRC grants GR/S01771, GR/S31099 and EP/N014758/1. Many thanks to Predrag Janičić and Alison Pease for drawing some of the images and to Andy Fugard for permission to use a diagram drawn by one of the participants in his study. Thanks to Gila Hanna and Andy Fugard for comments on an earlier version.

References

- Baker, S. (1993). *Aspects of the constructive omega rule within automated deduction*. Unpublished Ph.D. thesis, Edinburgh.
- de Bruijn, N. G. (1980). A survey of the project Automath. In J. P. Seldin & J. R. Hindley (Eds.), *To H. B. Curry; Essays on combinatory logic, lambda calculus and formalism* (pp. 579–606). Academic Press.
- Bundy, A. (2012). Reasoning about representations in autonomous systems: What Pólya and Lakatos have to say. In D. McFarland, K. Stenning, & M. McGonigle-Chalmers (Eds.), *The complex mind: An interdisciplinary approach*, chapter 9 (pp. 167–183). Palgrave Macmillan.
- Bundy, A., Jamnik, M., & Fugard, A. (2005). What is a proof? *Philosophical Transactions of the Royal Society A*, 363(1835), 2377–2392.
- Fugard, A. J. B. (2005). *An exploration of the psychology of mathematical intuition*. Unpublished M.Sc. thesis, School of Informatics, Edinburgh University.

- Gödel, K. (1931). Über formal unentscheidbare sätze der principia mathematica und verwandter systeme i. *Monatshefte für Mathematik und Physik*, 38, 173–198. English translation in [van Heijenoort, 1967].
- Hilbert, D. (1930). Die Grundlebung der elementahren Zahlenlehre (Vol. 104). *Mathematische Annalen*.
- Jamnik, M., & Bundy, A. (2005). Psychological validity of schematic proofs. In *Volume LNCS 2605 of lecture notes in computer science* (pp. 321–341). Springer-Verlag GmbH.
- Jamnik, M. (2001). *Mathematical reasoning with diagrams: From intuition to automation*. Stanford, CA: CSLI Press.
- Lakatos, I. (1976). *Proofs and refutations: The logic of mathematical discovery*. Cambridge University Press.
- Nelsen, R. B. (1993). *Proofs without words: Exercises in visual thinking*. The Mathematical Association of America.
- Robinson, A., & Voronkov, A. (Eds.). (2001). *Handbook of automated reasoning*, 2 volumes. Elsevier.
- Shoenfield, J. R. (1959). On a restricted ω -rule. *Bulletin de l'Académie Polonaise des Sciences: S'erie des sciences mathematiques, astronomiques et physiques*, 7, 405–407.
- van Heijenoort, J. (1967). *From Frege to Gödel: A source book in mathematical logic, 1879–1931*. Cambridge, Mass: Harvard University Press.
- Zach, R. (2009). Hilbert's program. *Stanford Encyclopedia of Philosophy*.

SMTCoq: Mixing Automatic and Interactive Proof Technologies



Chantal Keller

1 Introduction

Mechanization of mathematical reasoning can be seen as starting from two somewhat opposite applications (Mackenzie, 1995).

On the one hand, *interactive theorem provers* (also known as *proof assistants*) aim at checking (even complex) mathematical proofs with great confidence. Theorems and proofs should be stated and written interactively by mathematicians, with the help of the system to deduce facts, discharge automatically trivial sub-goals, and check the actual proof. To achieve confidence, these systems rely on a *kernel* that is a piece of code, as small as possible, implementing a proof checker for a well-defined logic (Harrison et al., 1996). Among the most successful current interactive theorem provers, one can cite the HOL family (Gordon, 2000) (HOL4, HOL Light, Isabelle/HOL), the type-theoretical family (Agda (Norell, 2009), Coq (Huet & Herbelin, 2014), Lean (de Moura, Kong, Avigad, van Doorn, & Jakob von Raumer, 2015), Matita (Asperti, Ricciotti, Coen, & Tassi, 2011), ...) and many other systems such as PVS (Owre, Rushby, & Shankar, 1992), Mizar (Bancerek et al., 2015) or Nuprl (Allen, Constable, Eaton, Kreitz, & Lorigo, 2000). Interactive theorem provers often come with high-level *tactics* that translate the interaction with the user into low-level proofs that are checked by the kernel. These tactics offer the possibility of having safe automation by performing complex reasoning while relying on the kernel. Most of the time, such tactics are dedicated decision procedures (Grégoire & Mahboubi, 2005; Besson, 2006): they can automatically solve problems that belong to a recognized fragment of a logic.

On the other hand, *automated theorem provers* aim at finding proofs fully automatically. Theorems should be stated in a logic accepted by the system which may, in return, prove it, give a counter-example, or fail, if the problem falls into an undecidable fragment or the proof search exceeds some heuristic limit. While the algorithms are shown to be correct on paper, actual implementations involve fast automatic proof

C. Keller (✉)

LRI, University of Paris-Sud, CNRS UMR 8623, Université Paris-Saclay,
Bât 650 Ada Lovelace Université Paris Sud, 91405 Orsay Cedex, France
e-mail: Chantal.Keller@lri.fr

© Springer Nature Switzerland AG 2019

G. Hanna et al. (eds.), *Proof Technology in Mathematics Research and Teaching*, Mathematics Education in the Digital Era 14,
https://doi.org/10.1007/978-3-030-28483-1_4

searches and may thus contain bugs (Brummayer & Biere, 2009). They are very powerful tools in proving automatically generated goals, for instance in the context of proof of programs (Filliâtre & Paskevich, 2013; Swamy et al., 2016). More recently they have been used to settle combinatorial problems such as the Erdős Discrepancy Conjecture (for discrepancy up to 3) (Konev & Lisitsa, 2015) or the Boolean Pythagorean Triples problem (Heule, Kullmann, & Marek, 2016). Currently, the most two successful approaches rely on satisfiability (Biere, Heule, van Maaren, & Walsh, 2009), in particular with Conflict-Driven Clause Learning SAT (Silva, Lynce, & Malik, 2019) and SMT (Satisfiability Modulo Theories) provers (zChaff (Fu, Marhajan, & Malik, 2007), CVC4 (Barrett et al., 2011), Z3 (de Moura & Bjørner, 2008), veriT (Bouton, de Oliveira, Déharbe, & Fontaine, 2009), ...), and saturation-based resolution and superposition (Bachmair & Ganzinger, 1998) with first-order provers (SPASS (Weidenbach et al., 2019), Vampire (Riazanov & Voronkov, 2002), E (Schulz, 2013), ...).

The idea to deploy both human interaction and expressive automation in a single tool started in the '90s. One can in particular cite NQTHM and its successor ACL2 (Kaufmann & Moore, 1996), which implements an interactive prover on top of a powerful automated prover. Confidence was achieved by implementing ACL2 in ACL2 itself, with the possibility of establishing properties on the system.

Subsequently, the independent success of both interactive and automated provers led again to the need for reconciling both worlds. A first approach, called the *autarkic* approach, close to what has been done for ACL2, was to implement and prove correct automatic provers inside proof assistants (Lescuyer, 2011; Hurd, 2005). This approach formally establishes the correctness of the underlying algorithms, but has the major drawback of fixing an implementation that would be very difficult to enhance without re-doing most of the proof work. More recently, the *skeptical* approach makes use of external solvers that, in addition to a yes/no answer, can output a *certificate*, that is to say the arguments underlying the proof they found, that allows proof reconstruction in the proof assistants (Böhme & Weber, 2010; Paulson & Blanchette, 2010; Blanchette, Kaliszyk, Paulson, & Urban, 2016; Armand et al., 2019). This approach actually scales since it allows the use of state-of-the-art external solvers, which may evolve independently. Moreover, proof checking is faster and easier rather than proof search.

The skeptical approach has had a major success with the Isabelle/HOL *sledgehammer* tactic (Paulson & Blanchette, 2010), that employs multiple external solvers in parallel and reconstructs an Isabelle/HOL proof script based in particular on autarkic solvers. It allows users of this interactive prover to sketch the interesting part of mathematical proofs (induction, intermediate lemmas) and leave the remaining automatic. This idea has been recently ported to the Coq proof assistant (Blanchette et al., 2016).

In this chapter, we present SMTCoq¹ (Armand et al., 2019; Ekici et al., 2016, 2017), a Coq plugin to interact with SAT and SMT external solvers via certificates. The objective is to provide a generic and efficient proof checker for automated

¹SMTCoq is available at <https://smtcoq.github.io>.

provers, with the same degree of confidence as Coq itself. This tool can be used to take advantage of automation in Coq (Sect. 4.3). Care has also been taken to ensure efficiency of proof checking (Sect. 3), which allows for the certification of big certificates that arise when proving large combinatorial problems (Sects. 4.1 and 4.2). We start by explaining the kind of problems that SMTCoq handles (Sect. 2).

2 The Satisfiability and Satisfiability Modulo Theories Problems

SAT solvers are automated provers to decide the satisfiability of (quantifier-free) Boolean formulas. They rely on an efficient exploration of the possible models of such formulas. By nature, they are powerful tools to solve combinatorial problems (Konev & Lisitsa, 2015; Heule et al., 2016).

On top of them, SMT solvers (standing for *Satisfiability Modulo Theory* solvers) automatically determine whether first-order formulas living in a combination of theories are satisfiable. Theories often include equality, arithmetic over various domains (integers, rationals, reals), and representation of data-structures (algebraic data-structures, arrays, bit vectors, ...). SMT solvers try to satisfy SMT formulas by the interaction of a SAT solver with theory reasoners (see (Nieuwenhuis, Oliveras, & Tinelli, 2006) for a detailed explanation), with the possibility to instantiate quantified hypotheses (de Moura & Bjørner, 2007; Barbosa, 2016), making the logic very expressive.

2.1 Examples

Let us illustrate the kind of problems SAT and SMT can be used for on the combinatorial example of the Erdős Discrepancy Conjecture.

Conjecture 1 (Erdős Discrepancy Conjecture) *For any infinite sequence $\langle x_1, x_2, \dots \rangle$ of ± 1 integers and any integer C , there exist integers k and d such that*

$$\left| \sum_{i=1}^k x_{i \times d} \right| > C$$

To prove this conjecture for a particular C_0 , one has to find a length l of sequences such that the formula

$$\forall (x_1, x_2, \dots, x_l), \forall k, \forall d, \left| \sum_{i=1}^k x_{i \times d} \right| \leq C_0$$

is unsatisfiable. Konev and Lisitsa proposed a non trivial encoding of this problem into SAT, allowing to prove the conjecture up to $C = 3$ with modern SAT solvers (Konev & Lisitsa, 2015). More naively, the problem can be easily encoded into SMT using the theory of Linear Integer Arithmetic.

Example 1 For $l = 6$, the formula can be encoded in SMT by the conjunction of:

- $(x_1 = -1 \vee x_1 = 1) \wedge \cdots \wedge (x_6 = -1 \vee x_6 = 1)$ (domain of the sequence)
- $(-C \leq x_1 + x_2 \leq C) \wedge \cdots \wedge (-C \leq x_1 + x_2 + x_3 + x_4 + x_5 + x_6 \leq C)$ (sums for $d = 1$)
- $(-C \leq x_2 + x_4 \leq C) \wedge (-C \leq x_2 + x_4 + x_6 \leq C)$ (sums for $d = 2$)
- $(-C \leq x_3 + x_6 \leq C)$ (sum for $d = 3$)

Most SMT solvers supporting integer arithmetic are able to prove the conjecture for $C_0 = 1$ by choosing the encoding for $l = 12$.

Theorem 1 *Any sequence of length at least 12 has discrepancy at least 2.*

The proof of such a theorem relies on (a) an encoding of the original problem into a SMT formula, encoding which must be automatically generated for $C \geq 2$ since the formula becomes very large and (b) an automatic proof from a SAT or SMT solver. To increase confidence, one can formally establish such theorems in a proof assistant by (a) proving the correctness of the encoding and its generator and (b) proving the correctness of the SMT answer, using the autarkic or skeptical approach.

SMTCoq is, in particular, a way to formally establish (b) for the Erdős Discrepancy Conjecture, based on the skeptical approach. In comparison to similar work in checking these proofs in Coq (Cruz-Filipe & Schneider-Kamp, 2017), there was no need to implement and prove correct a dedicated checker: SMTCoq is generic and efficient enough to encompass such proofs.

The expressivity of SMTCoq makes it possible to formally and efficiently check SAT and SMT proofs coming from any kind of problem. Indeed, SAT and in particular SMT are very expressive and can encode problems coming from multiple areas of mathematics and computer science. We illustrate this aspect by two examples coming from program testing, where one has to generate inputs satisfying the preconditions of a program (Example 2), and program proving, where one has to establish properties for all the possible runs of a program (Example 3).

Example 2 The problem of automatically generating sorted integer arrays of a given length is a satisfiability problem in the combination of the theories of arrays and integer arithmetic. For instance, for length 4, it can be formulated as such: find a value for the variable a (belonging to the sort of arrays) such that:

- $\text{length } a = 4$
- $a[0] \leq a[1] \wedge a[1] \leq a[2] \wedge a[2] \leq a[3]$

Example 3 To prove the correctness of the mergesort algorithm on arrays, one should be able to establish² that if:

²This corresponds to proving the invariant of the merge loop stating that the array is sorted up to a certain point.

- $\forall k_1 \leq k_2 < k, a[k_1] \leq a[k_2]$
- $\forall k_1 < k, a[k_1] \leq x$
- $a[k] = x$

then:

- $\forall l_1 \leq l_2 \leq k, a[l_1] \leq a[l_2]$

where a is an array and k_1, k_2, k and x are integers. The validity of this formula can be encoded as the unsatisfiability of the negation of the conclusion under the same hypotheses, which corresponds to the following SMT problem: check that the conjunction of

1. $\forall k_1 \leq k_2 < k, a[k_1] \leq a[k_2]$
2. $\forall k_1 < k, a[k_1] \leq x$
3. $a[k] = x$
4. $l_1 \leq l_2 \leq k \wedge a[l_1] > a[l_2]$

is unsatisfiable, that is to say that there is no concrete instance for the variables a, k, l_1 and l_2 that satisfy the four formulas.

2.2 SAT and SMT Proof Evidence

SMTCoq considers SMT solvers as black boxes that input a SMT problem and output evidence of the satisfiability or unsatisfiability of the problem (or nothing or a partial evidence if it was not able to solve it). The input format has been standardized in the SMT-LIB project (Barrett, de Moura, Ranise, Stump, & Tinelli, 2010) and is thus common to most state-of-the-art SMT solvers. However, the output format currently differs from one system to another.

If the problem is satisfiable, most provers return as evidence an instance of the variables that satisfies it, called a *model*.

Example 4 There exists a sequence of length 11 of discrepancy 1:

$$\langle 1, -1, -1, 1, -1, 1, 1, -1, -1, 1, 1 \rangle$$

So the SMT encoding of the Erdős Discrepancy Conjecture for $C_0 = 1$ and $l = 11$ is satisfiable and a possible model is:

$$\{x_1 \mapsto 1; x_2 \mapsto -1; x_3 \mapsto -1; x_4 \mapsto 1; x_5 \mapsto -1; x_6 \mapsto 1;$$

$$x_7 \mapsto 1; x_8 \mapsto -1; x_9 \mapsto -1; x_{10} \mapsto 1; x_{11} \mapsto 1\}$$

Example 5 The problem of Example 2 is satisfiable and a possible model is:

$$\{a \mapsto \boxed{-2} \boxed{3} \boxed{17} \boxed{42}\}$$

However, if the problem is unsatisfiable, while generic formats have been proposed (Stump, 2009; Besson, Fontaine, & Théry, 2011), the evidence output by various SMT solvers may differ a lot, particularly in terms of granularity of the proof. To interact with various solvers at small cost, SMTCoq is based on a certificate format inspired by Besson et al. (2011) that can represent most existing SMT reasoning, and is also modular to be easily extensible with new theories or proofs with a different level of details.³ This will make SMTCoq easy to extend at small cost, as detailed in Sect. 3.1.

The idea of this format is to combine independent steps. A *step* can be any deduction that transforms a (possibly empty) set of clauses into a clause that is implied: a step must preserve satisfiability of clauses. A *clause* consists of a disjunction of literals, where a *literal* can be any formula (positive literal) or its negation (negative literal).⁴

Example 6 The problem of Example 1 consists of 14 clauses:

- 6 clauses of two positive literals each (e.g. a positive literal is $x_4 = -1$) for the domain of the sequence;
- 5 clauses of one positive literal each for the sums for $d = 1$;
- 2 clauses of one positive literal each for the sums for $d = 2$;
- 1 clause of one positive literal for the sums for $d = 2$.

The four formulas corresponding to the problem of Example 3 are four clauses with a single positive literal (which is the formula itself).

The first clause deduced in Example 7:

$$\neg (a[l_1] > a[l_2]) \vee \neg (a[l_1] = a[l_2])$$

contains two negative literals.

A certificate then combines steps to deduce, in the end, the empty clause from the initial problem. Since the empty clause is unsatisfiable, and each step must preserve satisfiability, it implies that the initial problem is indeed unsatisfiable.

Example 7 The problem of Example 3 is unsatisfiable and a possible certificate starts with the following steps⁵:

Step	Deduced clause	Premises	Justification
5	$\neg (a[l_1] > a[l_2]) \vee \neg (a[l_1] = a[l_2])$	–	LIA
6	$\neg (l_1 = l_2) \vee (a[l_1] = a[l_2])$	–	Congruence
7	$l_1 \leq l_2 \leq k$	4	\wedge Projection
8	$a[l_1] > a[l_2]$	4	\wedge Projection
9	$\neg (l_1 = l_2)$	5, 6, 8	Resolution
...

³This format has been designed together with the veriT (Bouton et al., 2009) proof production engine.

⁴This definition of a clause is more general than the usual one: a literal can be any formula, even containing logical connectives.

⁵It corresponds to the certificate given by veriT, stable version of 2016.

This piece of certificate reads as follows. The first four steps (which are not written) consist of taking the four input clauses (in Example 3) as known clauses. Step 5 deduces a new clause from no premise (hence the clause must be a tautology) in the theory of Linear Integer Arithmetic. Step 6 again produces a tautology by congruence of equality with respect to array lookup. Steps 7 and 8 project the \wedge from the fourth input clause, respectively on the left-hand-side and on the right-hand-side. Finally, step 9 produces a new clause from steps 5, 6 and 8 by resolution (meaning that literals appearing both positively and negatively can be simplified out).

Example 8 Similarly, a certificate⁶ for the SMT problem corresponding to Theorem 1 is a proof of the empty clause obtained by combining, using resolution, the initial problem with tautologies in Linear Integer Arithmetic such as:

$$(1 \leq x_1 \wedge 1 \leq x_2) \Rightarrow x_1 + x_2 > 1$$

The main idea for modularity is that steps need only agree on the representation of formulas, but otherwise can be completely independent from each other. In particular, they independently deal with the various theories: as illustrated in the example, propositional reasoning is represented by resolution and connective steps (Tseitin, 1970); equality reasoning, by congruence (and transitivity) steps (Besson, Cornilleau, & Pichardie, 2019), ...etc. Moreover, they can have a different granularity: resolution is very fine-grained but nothing prevents a step from representing a full SAT solving step.

Notice that results of unsatisfiability are the main use of SMTCoq: as illustrated in Example 3, by contradiction, a formula is valid (i.e. always true) if and only if its negation is unsatisfiable. The remaining of the chapter thus focuses on this part. Nonetheless, checking the satisfiability given a model is much simpler.⁷

3 A Certified, Efficient and Modular Checker for SMT

3.1 A Modular Checker

The choice of the certificate format naturally induces a modular checker based on the architecture given in Fig. 1.

To each kind of step corresponds what we call a *small checker*, whose task is to check this particular kind of step independently of the other possible steps. The role of the *main checker* is simply to dispatch each step to the corresponding small checkers, and check in the end that the empty clause has been deduced.

⁶The certificate given by veriT, stable version of 2016, contains 178 steps.

⁷In Coq, one simply needs to assign the variables using the model and compute that the formula reduces to the true formula.

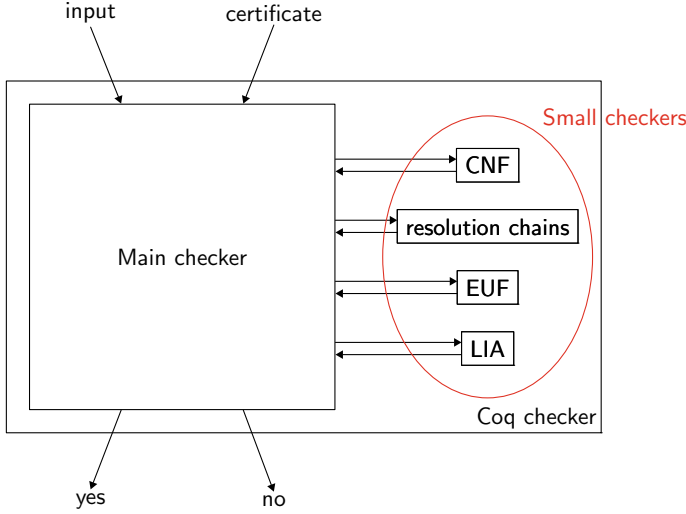


Fig. 1 Architecture of the SMTCoq checker

Small and main checkers operate over a state \mathcal{S} . This state initially contains the problem whose unsatisfiability is to be verified (Steps 1–4 in Example 7), and is, throughout the process, augmented with the clauses that are deduced by the small checkers (Steps 8 and beyond in Example 7). The data-structure used for states will be explained in the next subsection. One crucial aspect is that states can be embedded into Coq terms by a Coq function $\llbracket \bullet \rrbracket_\rho : \mathcal{S} \rightarrow \text{bool}$ that interprets the state as the conjunction of the interpretation of each formula, and for each formula, interprets each syntactic connective and operator by its Coq counterpart (we refer the reader to (Armand et al., 2019) for a detailed explanation of this interpretation function). As standard, the valuation ρ is a mapping of the variables to Coq terms. The property $\forall \rho, \llbracket s \rrbracket_\rho = \text{false}$ thus means that a state $s \in \mathcal{S}$ is unsatisfiable.

As explained in the previous section, a small checker takes as input a (possibly empty) set of clauses and returns a new clause that is implied, in the sense of satisfiability. Concretely, a small checker is thus given by:

- a function $\text{sc} : \mathcal{S} \rightarrow \text{step} \rightarrow \mathcal{S}$ that, given a state and a step, returns the state augmented with the deduced clause;
- a proof that this function preserves satisfiability:

$$\text{sc_ok} : \forall (s:\mathcal{S}) (p:\text{step}), \\ \forall \rho, \llbracket s \rrbracket_\rho = \text{true} \Rightarrow \llbracket \text{sc } s \text{ p} \rrbracket_\rho = \text{true}$$

Adding a new small checker consists of providing such a function and its proof of correctness, independently of existing small checkers, making SMTCoq extensions to new checkers easy.

As the figure suggests, various small checkers have already been implemented for major SMT theories: the initial development of SMTCoq (Armand et al., 2019)

implemented small checkers for propositional reasoning (via CNF computation and resolution), Equality of Uninterpreted Functions, and Linear Integer Arithmetic, and implementations of small checkers for the theories of bit vectors and arrays have been recently added (Ekici et al., 2016), confirming the modularity of SMTCoq.

3.2 An Efficient Checker

For the skeptical approach to be practical, certificate checking must be far cheaper than proof search. This is theoretically the case for most concrete SAT and SMT problems, and SMTCoq has been designed to be as efficient as possible while being implemented and proved correct inside Coq.

The SMTCoq checker has been designed to run in a branch of Coq, called `native-coq`⁸ (Boespflug, Dénès, & Grégoire, 2019), that in particular lifts in Coq native data-structures such as machine integers and mutable arrays (with history), while preserving soundness. SMTCoq makes intensive use of these data-structures to be efficient.

As an example, formulas are hash-consed using mutable arrays instead of being represented by a standard recursive algebraic datatype: each sub-formula is stored in a cell of the array, and is referred to by its index in the array (which is a machine integer). Literals are encoded as follows: the positive literal associated to the formula at index i is represented by $2 \times i$, and the negative literal, by $2 \times i + 1$. This encoding enjoys the following aspects:

- it is efficiently represented in memory, since it has maximal sharing;
- computations that appear often in SMT checking are really fast: for instance, checking if a literal l is the negation of a literal m is computed by the bitwise operations $l \oplus m = 1$.

Example 9 The fourth formula of Example 3 is the formula at index 2 in the following array:

Index	0	1	2
Sub-formula	$l_1 \leq l_2 \leq k$	$a[l_1] > a[l_2]$	$0 \wedge 2$

It corresponds to $0 \wedge 2$ since $0 = 2 \times 0$ is the positive literal associated to the sub-formula at index 0 and $2 = 2 \times 1$ is the positive literal associated to the sub-formula at index 1.

Another place where native data-structures are crucial is in the presentation of states. As detailed above, states start with the initial problem and are “augmented” with new clauses that are deduced. Simply keeping all the clauses is infeasible in

⁸The `native-coq` branch of Coq is progressively being integrated into the main version.

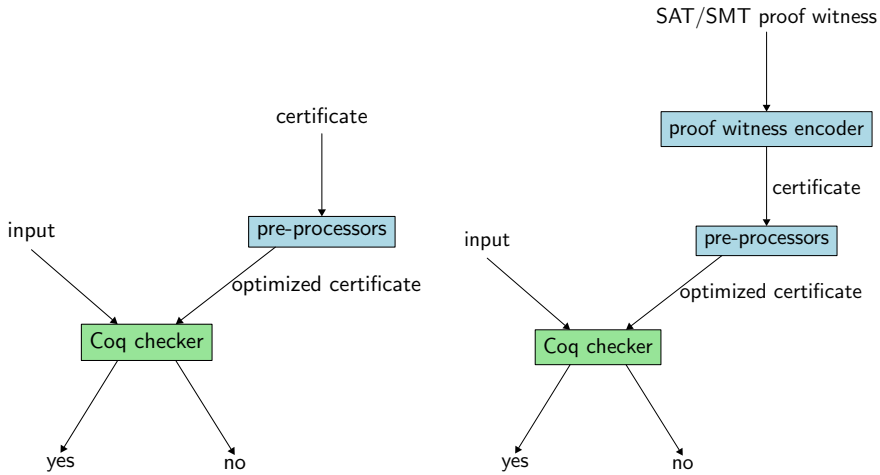


Fig. 2 A certificate can be arbitrarily transformed before being validated by the Coq checker

practice, since a certificate may produce thousands of them. Thus, a state is a mutable array, whose length is (at least) the number of clauses that are alive at the same time: once all clauses that are implied from a clause c have been deduced, clause c is not useful anymore.

Example 10 In Examples 3 and 7, the fourth clause will not be useful anymore after Steps 7 and 8, and can thus be removed from the state.

It is necessary to know in advance, before certificate checking, how many clauses are alive at the same time and in which cell to allocate each clause (in order to overwrite clauses that will not be used anymore). This is done by another nice property of the skeptical method: before checking, certificates can be transformed as needed, and the whole process remains sound even if certificate transformations are not proved correct. Indeed, if certificates are transformed in an unsound way, the checker will not be able to reconstruct a proof.

This principle is widely applied in SMTCoq: upstream from the checker presented in this section, many preprocessors have been implemented (without the need to certify them), that in particular allocate clauses, but also clean certificates from unused steps, ...etc. The checker of Fig. 1 is thus used in the context of Fig. 2 (left), where the preprocessors need not be certified.

3.3 Modular Link with State-of-the-Art SMT Solvers

In addition to efficiency, the preprocessing technique allows the use of the Coq checker with any SMT solver without more certification, even if there is no standard

for SMT proof witnesses (Sect. 2): it is sufficient to encode proof witnesses into the SMTCoq certificate format, and this encoding does not need to be proved (see Fig. 2 (right)). Thus, handling a new solver is simply writing an uncertified encoder, and the SMTCoq format is generic enough to welcome state-of-the-art solver’s formats.

Encoders for the SAT solvers zChaff and Glucose, as well as the SMT solvers veriT and CVC4 are currently implemented, allowing an efficient check of the answers of all these solvers with the same certified checker.

4 Applications

4.1 Certified Validation

The direct application of this checker is to certify answers coming from SAT and SMT solvers: given a SAT or SMT problem and a proof witness provided by a supported solver, the checker can be used to check the unsatisfiability of the input problem, using the proof witness as a hint. This idea is detailed in Fig. 3 (top left). Note that in this use case, the parser of the SAT/SMT problem must be trusted. If it was to replace the input problem with a trivially unsatisfiable problem, then the checker could easily answer “yes” but it would have certified nothing! In SMTCoq, the parser has not been certified, but it is a very small piece of code that straightforwardly transforms a string into the corresponding SAT/SMT abstract syntax tree (contrary to the encoders and preprocessors that can perform arbitrary transformations). Hence, in this application, if the checker answers “yes”, we can be sure that the original problem is unsatisfiable: the checker is correct. Note that, however, if the checker answers “no”, we know nothing: the answer coming from the solver may be invalid or incomplete, or the checker may fail to check the proof since it is not shown to be complete. However, it has been tested against a very large benchmark of problems (coming from the SAT and SMT competitions) to make sure that it is complete in practice.

To handle this application, SMTCoq offers two possibilities: the checker can be called from Coq via a dedicated command, or extracted to the OCaml programming language to be used without the need to install Coq. Thanks to the use of native-coq, both methods are really efficient: applications to the benchmarks of the SAT and SMT competitions showed that proof search by the solvers is the bottleneck, but not proof checking by SMTCoq (Armand et al., 2019).

4.2 Theorem Import

More generally, the checker can be used to safely import new theorems into Coq (Fig. 3 (top right)). Given a problem and a proof witness provided by a supported

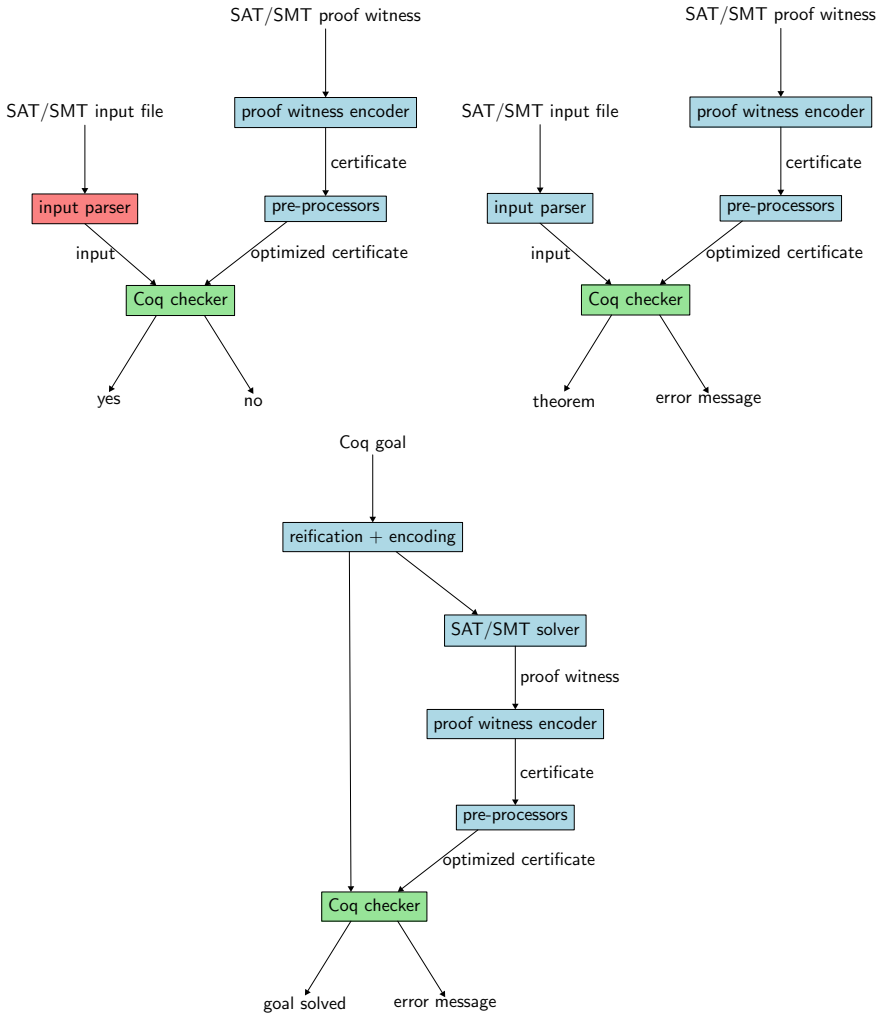


Fig. 3 Applications of the SMTCoq checker

solver, a Coq command generates a new theorem, proved by an application of the correctness of the checker (if the checker fails, then the theorem is not proved and not added to Coq). This theorem can then be used to deduce facts inside Coq. Contrary to the previous subsection, the input parser does not need to be trusted anymore; if it changes the statement of the problem, then a useless theorem may be imported in Coq, but it will not compromise soundness.

As illustrated in Sect. 2, this mechanism has been used to check mathematical proofs containing very large combinatorial results, such as the proof of the Erdős Discrepancy, by importing the combinatorial part from SAT and proving the remaining

as standard in Coq. An ongoing work applies this methodology to other combinatorial proofs such as the Boolean Pythagorean Triples problem (Heule et al., 2016). Moreover, SMTCoq is generic enough to import theorems based on SMT solving from many domains.

4.3 Automatic Tactics

Finally, as discussed in the introduction, SMTCoq can also be used to automatically solve Coq goals by discharging them to a SAT or a SMT solver and checking the answer (Fig. 3 (bottom)). The input problem given to the solver comes from a concrete goal that the user wants to prove: the goal is provable if and only if its negation is unsatisfiable. Then, the same process as before is used. Hopefully, the chosen solver returns a proof witness that can be verified by the SMTCoq checker, and if so, the correctness of this latter allows to conclude. Notice that, if the goal is not provable, then its negation is satisfiable and the SAT/SMT solver may return a model that can be used to give a counter-example to the user.

SMTCoq comes with such tactics for most supported provers, which are actually able to solve goals that belong to the combination of theories supported by the provers. Ongoing work (Ekici et al., 2017) consists of improving the expressivity of these tactics, in particular by encoding goals that are not directly supported by the logic of SMT.

5 Conclusion and Perspectives

We have presented SMTCoq, a plug-in for the Coq proof assistant to work in conjunction with external SAT/SMT solvers. SMTCoq has been designed to be modular at many levels (handling new theories and new provers), making it extensible at small cost, and already comes with support for state-of-the-art SAT and SMT solvers. It is distributed as a Coq plug-in that users can enjoy, and is still under active development and expansion. It can be used for various applications ranging from formal proofs of combinatorial problems to day-to-day automation in Coq.

Recently, mathematicians have more frequently used programs and automated provers to establish new results. In addition to the combinatorial problems already presented, two major successes are the proofs of the four-color theorem (Gonthier, 2007) and of the Kepler conjecture (Hales et al., 2015). SMTCoq is today a generic way to certify some of these proofs, and we argue that, as a Coq plugin, it may become a way to revisit and discover mathematical knowledge by structuring them in a new way (Gonthier et al., 2013).

The certificate approach has been designed first for proof exchange and proof checking. However, we believe that it is more general and will become a standard way of designing reliable proofs in mathematics and computer science.

As presented in the introduction, SMTCoq belongs to a long-term goal to take advantage of mechanized mathematical reasoning which is both automatic and extremely reliable. In this direction, having a single tool is unrealistic: many different proof assistants and automated provers have been designed in the last decades because they all have strong and weak points (regarding automation and reliability, but also expressivity, degree of expertise needed to master them, ...etc.). We rather advocate interoperability between different proof systems, and strongly argue that it relies on *universality* of proofs and mathematical libraries (Miller, 2013; Kohlhase & Rabe, 2016; Saillard, 2015). Proof systems should be able to output evidence of their reasoning, in such a way that it can be combined with proofs coming from other systems, while having the flexibility to have various granularity and underlying logic.

With respect to computer science, many successful tools (Filliâtre & Paskevich, 2013; Swamy et al., 2016) have been designed to prove the correctness of software based on the autarkic approach, and are used in critical industries such as avionics or cryptography (Jourdan, Laporte, Blazy, Leroy, & Pichardie, 2015; Delignat-Lavaud et al., 2017). To reach a larger audience, and even be applicable to most software, the skeptical approach offers a lighter technique that separates the software design from its verification: instead of certifying (possibly complex) algorithms, we certify checkers for their answers. Ongoing works are applying this method to other domains than proof checking, with an objective to generalize certificates rather than having to design a checker for each application. Recent cryptographic technologies can be applied in this direction (Parno, Howell, Gentry, & Raykova, 2013; Fournet, Keller, & Laporte, 2016).

Recent works show that the interoperability between systems could be designed in a correct-by-construction approach rather than relying on certificates (Bonacina, Graham-Lengrand, Shankar, 2018; Boulmé & Maréchal, 2017). A way of understanding it is that a certificate checker can be turned into the kernel of a certificate producer (Boulmé & Maréchal, 2017), in the same sense as a proof assistant. Ongoing work consists of transforming SMTCoq into a kernel for an SMT solver that could experiment with many proof search strategies without compromising soundness.⁹ Further work will lead to an understanding as to how this technique may apply to generic a posteriori certification of software discussed in the previous paragraph.

Acknowledgements This work would be impossible without the help from past, present and future contributors of SMTCoq. Past and present contributors are Mikäl Armand, Clark Barrett, Valentin Blot, Burak Ekici, Germain Faure, Benjamin Grégoire, Guy Katz, Tianyi Liang, Alain Mebsout, Andrew Reynolds, Laurent Théry, Cesare Tinelli and Benjamin Werner. Contributors to the certification of the Erdős Discrepancy Conjecture include Maxime Dénès and Pierre-Yves Strub.

The author thanks Véronique Benzaken and Évelyne Contejean for providing good feedback in the choice of the examples.

The author finally thanks the editors for their invitation to contribute to this book, and the reviewers and editors for their valuable feedback.

⁹This research is supported by Labex DigiCosme (project ANR11LABEX0045DIGICOSME) operated by ANR as part of the program «Investissement d’Avenir» Idex ParisSaclay (ANR11IDEX000302).

References

- Allen, S. F., Constable, R. L., Eaton, R., Kreitz, C., & Lorigo, L. (2000). The Nuprl open logical environment. In D. A. McAllester (Ed.), *Proceedings of Automated Deduction—CADE-17, 17th International Conference on Automated Deduction, Pittsburgh, PA, USA, 17–20 June 2000* (Vol. 1831, pp. 170–176). Lecture Notes in Computer Science. Springer.
- Armand, M., Faure, G., Grégoire, B., Keller, C., Théry, L., & Werner, B. (2011). A modular integration of SAT/SMT solvers to coq through proof witnesses. In: J.-P. Jouannaud & Z. Shao (pp. 135–150).
- Asperti, A., Ricciotti, W., Coen, C.S., & Tassi, E. (2011). The matita interactive theorem prover. In N. Bjørner & V. Sofronie-Stokkermans (Eds.), *Proceedings of Automated Deduction—CADE-23—23rd International Conference on Automated Deduction, Wrocław, Poland, July 31–August 5, 2011* (Vol. 6803, pp. 64–69). Lecture Notes in Computer Science. Springer.
- Barbosa, H. (2016). Efficient instantiation techniques in SMT (work in progress). In P. Fontaine, S. Schulz, & J. Urban (Eds.), *Proceedings of the 5th Workshop on Practical Aspects of Automated Reasoning co-located with International Joint Conference on Automated Reasoning (IJCAR 2016), Coimbra, Portugal, 2 July 2016, CEUR Workshop Proceedings* (Vol. 1635, pp. 1–10). CEUR-WS.org.
- Brummayer, B., & Biere, A. (2009). Fuzzing and delta-debugging SMT solvers. In *Proceedings of the 7th International Workshop on Satisfiability Modulo Theories* (pp. 1–5). ACM.
- Bancerek, G., Bylinski, C., Grabowski, A., Kornilowicz, A., Matuszewski, R., Naumowicz, et al. (2015). Mizar: state-of-the-art and beyond. In M. Kerber, J. Carette, C. Kaliszyk, F. Rabe & V. Sorge (Eds.), *Proceedings of Intelligent Computer Mathematics—International Conference, CICM 2015, Washington, DC, USA, 13–17 July 2015* (Vol. 9150, pp. 261–279). Lecture Notes in Computer Science. Springer.
- Barrett, C., Conway, C.L., Deters, M., Hadarean, L., Jovanovic, D., King, T., et al. (2011). CVC4. In G. Gopalakrishnan & S. Qadeer (Eds.), *Proceedings of Computer Aided Verification—23rd International Conference, CAV 2011, Snowbird, UT, USA, 14–20 July 2011* (Vol. 6806, pp. 171–177). Lecture Notes in Computer Science. Springer.
- Besson, F., Cornilleau, P.-E., & Pichardie, D. (2011). In J.-P. Jouannaud, & Z. Shao (Eds.), *Modular SMT proofs for fast reflexive checking inside coq* (pp. 151–166).
- Boespflug, M., Dénès, M., & Grégoire, B. (2011). In J.-P. Jouannaud, & Z. Shao (Eds.), *Full reduction at full throttle* (pp. 362–377).
- Barrett, C., de Moura, L. M., Ranise, S., Stump, A., & Tinelli, C. (2010). The SMT-LIB initiative and the rise of SMT—(HVC 2010 award talk). In S. Barner, I. G. Harris, D. Kroening, & O. Raz (Eds.), *Hardware and Software: Verification and Testing—6th International Haifa Verification Conference, HVC 2010, Haifa, Israel, 4–7 October 2010*. Revised Selected Papers (Vol. 6504, p. 3). Lecture Notes in Computer Science. Springer.
- Besson, F. (2006). Fast reflexive arithmetic tactics the linear case and beyond. In T. Altenkirch & C. McBride (Eds.), *TYPES* (Vol. 4502, pp. 48–62). Lecture Notes in Computer Science. Springer.
- Besson, F., Fontaine, P., & Théry, L. (2011). A flexible proof format for SMT: A proposal. In *PxTP 2011: First International Workshop on Proof eXchange for Theorem Proving August 1, 2011 Affiliated with CADE 2011, 31 July–5 August 2011 Wrocław, Poland* (pp. 15–26).
- Bachmair, L., & Ganzinger, H. (1998). Equational reasoning in saturation-based theorem proving. *Automated deduction—A basis for applications* (Vol. 1, pp. 353–397).
- Bonacina, M. P., Graham-Lengrand, S., & Shankar, N. (2018). Proofs in conflict-driven theory combination. In J. Andronick & A. P. Felty (Eds.), *Proceedings of the 7th ACM SIGPLAN International Conference on Certified Programs and Proofs, CPP 2018, Los Angeles, CA, USA, 8–9 January 2018* (pp. 186–200). ACM.
- Biere, A., Heule, M., van Maaren, H., & Walsh, T. (Eds.). (2009). *Handbook of satisfiability* (Vol. 185). Frontiers in Artificial Intelligence and Applications. IOS Press.
- Blanchette, J. C., Kaliszyk, C., Paulson, L. C., & Urban, J. (2016). Hammering towards QED. *Journal of Formalized Reasoning*, 9(1), 101–148.

- Boulmé, S., & Maréchal, A. (2017). Toward certification for free! Working paper or preprint, July 2017.
- Bouton, T., de Oliveira, D., Déharbe, D., & Fontaine, P. (2009). In R. A. Schmidt (Eds.), *veriT: An open, trustworthy and efficient SMT-solver* (pp. 151–156).
- Böhme, S. B., & Weber, T. (2010). Fast LCF-style proof reconstruction for Z3. In M. Kaufmann & L. C. Paulson (Eds.), *ITP*, (Vol. 6172, pp. 179–194). Lecture Notes in Computer Science. Springer.
- Cruz-Filipe, L., & Schneider-Kamp, P. (2017). Formally proving the Boolean pythagorean triples conjecture. In T. Eiter & D. Sands (Eds.), *LPAR-21, 21st International Conference on Logic for Programming, Artificial Intelligence and Reasoning, Maun, Botswana, 7–12 May 2017* (Vol. 46, pp. 509–522). EPiC Series in Computing. EasyChair.
- Delignat-Lavaud, A., Fournet, C., Kohlweiss, M., Protzenko, J., Rastogi, A., Swamy, N., et al. (2017). Implementing and proving the TLS 1.3 record layer. In *2017 IEEE Symposium on Security and Privacy, SP 2017, San Jose, CA, USA, 22–26 May 2017* (pp. 463–482). IEEE Computer Society.
- de Moura, L. M., & Bjørner, N. (2007). Efficient e-matching for SMT solvers. In F. Pfenning (Ed.), *Proceedings of Automated Deduction—CADE-21, 21st International Conference on Automated Deduction, Bremen, Germany, 17–20 July 2007* (Vol. 4603, pp. 183–198). Lecture Notes in Computer Science. Springer.
- de Moura, L. M., & Bjørner, N. (2008). Z3: An efficient SMT solver. In C. R. Ramakrishnan & J. Rehof (Eds.), *TACAS* (Vol. 4963, pp. 337–340). Lecture Notes in Computer Science. Springer.
- de Moura, L., Kong, S., Avigad, J., van Doorn, F., & von Raumer, J. (2015). The lean theorem prover (system description). In A. P. Felty & Aart Middeldorp (Eds.), *Proceedings of Automated Deduction—CADE-25—25th International Conference on Automated Deduction, Berlin, Germany, 1–7 August 2015* (Vol. 9195, pp. 378–388). Lecture Notes in Computer Science. Springer.
- Ekici, B., Katz, G., Keller, C., Mebsout, A., Reynolds, A. J., & Tinelli, C. (2016). Extending SMT-Coq, a certified checker for SMT (Extended Abstract). In J. Christian Blanchette & C. Kaliszyk (Eds.), *Proceedings First International Workshop on Hammers for Type Theories, HaTT@IJCAR 2016, Coimbra, Portugal, 1 July 2016* (Vol. 210, pp. 21–29). EPTCS.
- Ekici, B., Mebsout, A., Tinelli, C., Keller, C., Katz, G., Reynolds, A., et al. (2017). SMTCoq: A plugin for integrating SMT solvers into Coq. In R. Majumdar & V. Kuncak (Eds.), *Computer Aided Verification—29th International Conference, CAV 2017, Heidelberg, Germany, 24–28 July 2017, Proceedings, Part II* (Vol. 10427, pp. 126–133). Lecture Notes in Computer Science. Springer.
- Fournet, C., Keller, C., & Laporte, V. (2016). A certified compiler for verifiable computing. In *IEEE 29th Computer Security Foundations Symposium, CSF 2016, Lisbon, Portugal, June 27–July 1, 2016* (pp. 268–280). IEEE Computer Society.
- Fu, Z., Marhajan, Y., & Malik, S. (2007). zChaff. Research Web Page. Princeton University, USA, March 2007. <http://www.princeton.edu/~chaff/zchaff.html>.
- Filliâtre, J.-C., & Paskevich, A. (2013). Why3—Where programs meet provers. In M. Felleisen & P. Gardner (Eds.), *Proceedings of Programming Languages and Systems—22nd European Symposium on Programming, ESOP 2013, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2013, Rome, Italy, 16–24 March 2013* (Vol. 7792, pp. 125–128). Lecture Notes in Computer Science. Springer.
- Gonthier, G., Asperti, A., Avigad, J., Bertot, Y., Cohen, C., Garillot, F., et al. (2013). A machine-checked proof of the odd order theorem. In S. Blazy, C. Paulin-Mohring & D. Pichardie (Eds.), *Proceedings of Interactive Theorem Proving—4th International Conference, ITP 2013, Rennes, France, July 22–26, 2013* (Vol. 7998, pp. 163–179). Lecture Notes in Computer Science. Springer.
- Grégoire, B., & Mahboubi, A. (2005). Proving equalities in a commutative ring done right in Coq. In J. Hurd & T. F. Melham (Eds.), *TPHOLs* (Vol. 3603, pp. 98–113). Lecture Notes in Computer Science. Springer.
- Gonthier, G. (2007). The four colour theorem: Engineering of a formal proof. In D. Kapur (Ed.), *Computer Mathematics, 8th Asian Symposium, ASCM 2007, Singapore, 15–17 December 2007. Revised and Invited Papers* (Vol. 5081, p. 333). Lecture Notes in Computer Science. Springer.

- Gordon, M. (2000). From LCF to HOL: A short history. In G. D. Plotkin, C. Stirling & M. Tofte (Eds.), *Proof, language, and interaction, essays in honour of Robin Milner* (pp. 169–186). The MIT Press.
- Harrison, J., et al. (1996). *Formalized mathematics*. Citeseer.
- Hales, T. C., Adams, M., Bauer, G., Dang, D. T., Harrison, J., Le Hoang, T., et al. (2015). A formal proof of the Kepler conjecture. *CoRR*. [arXiv:abs/1501.02155](https://arxiv.org/abs/1501.02155).
- Huet, G. P., & Herbelin, H. (2014). 30 years of research and development around coq. In S. Jagannathan & P. Sewell (Eds.), *The 41st Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL '14, San Diego, CA, USA, 20-21 January 2014* (pp. 249–250). ACM.
- Heule, M. J. H., Kullmann, O., & Marek, V. W. (2016). Solving and verifying the Boolean pythagorean triples problem via cube-and-conquer. In N. Creignou & D. Le Berre (Eds.), *Proceedings of Theory and Applications of Satisfiability Testing—SAT 2016—19th International Conference, Bordeaux, France, 5–8 July 2016* (Vol. 9710, pp. 228–245). Lecture Notes in Computer Science. Springer.
- Hurd, J. (2005). System description: The Metis proof tactic. In *Empirically Successful Automated Reasoning in Higher-Order Logic (ESHO)* (pp. 103–104).
- Jourdan, J.-H., Laporte, V., Blazy, S., Leroy, X., & Pichardie, D. (2015). A formally-verified C static analyzer. In S. K. Rajamani & D. Walker (Eds.), *Proceedings of the 42nd Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL 2015, Mumbai, India, 15–17 January 2015* (pp. 247–259). ACM.
- Jouannaud, J.-P. & Shao, Z. (Eds.). (2011). *Proceedings of Certified Programs and Proofs—First International Conference, CPP 2011, Kenting, Taiwan, 7–9 December 2011* (Vol. 7086). Lecture Notes in Computer Science. Springer.
- Konev, B., & Lisitsa, A. (2015). Computer-aided proof of erdős discrepancy properties. *Artificial Intelligence*, 224, 103–118.
- Kaufmann, M., & Moore, J. S. (1996). ACL2: An industrial strength version of Nqthm. In *Proceedings of the Eleventh Annual Conference on Computer Assurance, 1996. COMPASS'96, Systems Integrity, Software Safety, Process Security* (pp. 23–34). IEEE.
- Kohlhase, M., & Rabe, F. (2016). QED reloaded: Towards a pluralistic formal library of mathematical knowledge. *Journal of Formalized Reasoning*, 9(1), 201–234.
- Lescuyer, S. (2011). *Formalizing and Implementing a Reflexive Tactic for Automated Deduction in Coq. (Formalisation et développement d'une tactique reflexive pour la demonstration automatique en coq)*. Ph.D. thesis, University of Paris-Sud, Orsay, France.
- MacKenzie, D. (1995). The automation of proof: A historical and sociological exploration. *IEEE Annals of the History of Computing*, 17(3), 7–29.
- Miller, D. (2013). Foundational proof certificates: Making proof universal and permanent. In A. Momigliano, B. Pientka & R. Pollack (Eds.), *Proceedings of the Eighth ACM SIGPLAN International Workshop on Logical Frameworks & Meta-languages: Theory & Practice, LFMTP 2013, Boston, Massachusetts, USA, 23 September 2013* (pp. 1–2). ACM.
- Norell, U. (2009). Dependently typed programming in Agda. In A. Kennedy & A. Ahmed (Eds.), *Proceedings of TLDI'09: 2009 ACM SIGPLAN International Workshop on Types in Languages Design and Implementation, Savannah, GA, USA, 24 January 2009* (pp. 1–2). ACM.
- Nieuwenhuis, R., Oliveras, A., & Tinelli, C. (2006). Solving SAT and SAT modulo theories: From an abstract Davis-Putnam-Logemann-Loveland procedure to DPPL(). *Journal of the ACM*, 53(6), 937–977.
- Owe, S., Rushby, J. M., & Shankar, N. (1992). PVS: A prototype verification system. In D. Kapur (Ed.), *Proceedings of Automated Deduction—CADE-11, 11th International Conference on Automated Deduction, Saratoga Springs, NY, USA, 15–18 June 1992* (Vol. 607, pp. 748–752). Lecture Notes in Computer Science. Springer.
- Paulson, L. C., & Blanchette, J. C. (2010). Three years of experience with sledgehammer, a practical link between automatic and interactive theorem provers. In G. Sutcliffe, S. Schulz & E. Ternovska

- (Eds.), *The 8th International Workshop on the Implementation of Logics, IWIL 2010, Yogyakarta, Indonesia, 9 October 2011* (Vol. 2, pp. 1–11). EPiC Series in Computing. EasyChair.
- Parno, B., Howell, J., Gentry, C., & Raykova, M. (2013). Pinocchio: Nearly practical verifiable computation. In *2013 IEEE Symposium on Security and Privacy, SP 2013, Berkeley, CA, USA, 19–22 May 2013* (pp. 238–252). IEEE Computer Society.
- Riazanov, A., & Voronkov, A. (2002). The design and implementation of VAMPIRE. *AI Communications*, 15(2–3), 91–110.
- Saillard, R. (2015). *Typechecking in the lambda-Pi-Calculus Modulo : Theory and Practice. (Vérification de typage pour le lambda-Pi-Calcul Modulo : théorie et pratique)*. Ph.D. thesis, Mines ParisTech, France.
- Schmidt, R. A. (Ed.). (2009). *Proceedings of Automated Deduction—CADE-22, 22nd International Conference on Automated Deduction, Montreal, Canada, 2–7 August 2009* (Vol. 5663). Lecture Notes in Computer Science. Springer.
- Schulz, S. (2013). System description: E 1.8. In K. McMillan, A. Middeldorp, & A. Voronkov (Eds.), *Proceedings of the 19th LPAR, Stellenbosch* (Vol. 8312). LNCS. Springer.
- Swamy, N., Hritcu, C., Keller, C., Rastogi, A., Delignat-Lavaud, A., Forest, S., et al.: Dependent types and multi-monadic effects in F. In R. Bodík, R. Majumdar (Eds.), *Proceedings of the 43rd Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL 2016, St. Petersburg, FL, USA, 20–22 January 2016* (pp. 256–270). ACM.
- Silva, J. P. M., Lynce, I., & Malik, S. (2009). Conflict-driven clause learning SAT solvers. In Biere et al. (Ed.), (pp. 131–153).
- Stump, A. (2009). Proof checking technology for satisfiability modulo theories. *Electronic Notes in Theoretical Computer Science*, 228, 121–133.
- Tseitin, G. (1970). On the complexity of proofs in propositional logics. *Seminars in Mathematics*, 8, 466–483.
- Weidenbach, C., Dimova, D., Fietzke, A., Kumar, R., Suda, M., & Wischniewski, P. (2009). SPASS version 3.5. In R. A. Schmidt (Ed.), (pp. 140–145).

Studying Algebraic Structures Using Prover9 and Mace4



Rob Arthan and Paulo Oliva

1 Introduction

Prover9 is an automated theorem prover for first order logic (McCune, 2005). The companion program Mace4 searches for finite models of first order theories. Tools such as these have been used to attack problems in algebra for many years. A headline result was McCune's use of a predecessor of Prover9 to find the first proof of the Robbins conjecture (McCune, 1997). It is noteworthy in connection with the present chapter that the first proof relied on reductions of the conjecture that were originally proved by Winker (1992) using a mixture of human reasoning and automated proof search. Several people worked on human readable accounts of the machine-generated proof (Dahn, 1998; Burris, 1997).

It is the goal of this chapter to illustrate an approach that we have found very useful in research on algebraic structures: we develop human readable proofs of new results by mixing automated proof search and human analysis of the proofs found by Prover9 and of examples found by Mace4. We propose that guided use of such tools on known examples could also be of benefit in teaching algebra.

In Sect. 1.1 we illustrate the use of the tools taking the theory of *semilattices* as a simple example. In Sect. 1.2 we give an introduction to the class of algebraic structures called *hoops* and apply Prover9 and Mace4 to an investigation of these algebraic structures at the level of a possible undergraduate project.

In Sect. 2 of this chapter, we present some results from our own research obtained with the assistance of Prover9 and Mace4. Surprisingly, Prover9's machine-generated

Electronic supplementary material The online version of this chapter (https://doi.org/10.1007/978-3-030-28483-1_5) contains supplementary material, which is available to authorized users.

R. Arthan · P. Oliva (✉)
School of Electronic Engineering and Computer Science, Queen Mary University
of London, Mile End Road, London E1 4NS, UK
e-mail: p.oliva@qmul.ac.uk

R. Arthan
e-mail: rda@lemma-one.com

proofs can be tractable and useful artefacts. Human insight allows us to extract new abstractions and further conjectures, leading to an iterative interactive process for developing the theory and producing a human-oriented account of the proofs.

The examples in this chapter are supported by a set of Mace4 and Prover9 scripts, available as ESM from https://doi.org/10.1007/978-3-030-28483-1_5. There the reader can also find information about obtaining Prover9 and Mace4 and instructions for running the scripts. The examples are organised into two separate folders, one for Sect. 1 and one for Sect. 2. In these sections, references to named files refer either to the scripts or the output files produced by running them in the corresponding folder.

As a final introductory remark, we would like to stress that we are concerned here with applications of tools. Applications invariably suggest desirable new features or new tools, but we are concerned here with the development of mathematical knowledge, either in a research or a teaching context, by exploiting potential synergy between human capabilities and the capabilities of tools that are available now. We believe that this kind of synergy will be required at every stage of technological advancement and that to learn this from practical experience is a worthwhile achievement in its own right. We confine our thoughts on possible future developments to the tools to our concluding remarks.

1.1 Using Prover9 and Mace4

In this section we will introduce Prover9 and Mace4 using the theory of semilattices as an example.

Recall that a semilattice can be defined as a set equipped with a least upper bound operation on pairs of elements: i.e., a structure (S, \cup) where the binary operation \cup is associative, commutative and idempotent, i.e., it satisfies:

$$\begin{aligned}x \cup (y \cup z) &= (x \cup y) \cup z \\x \cup y &= y \cup x \\x \cup x &= x.\end{aligned}$$

Prover9 and Mace4 use a common syntax for first-order logic. In this syntax, we can formalise the semilattice axioms as follows (see file `semilattice.ax`):

```
op(500, infix, "cup").
formulas(assumptions).
  (x cup y) cup z = x cup (y cup z).
  x cup y = y cup x.
  x cup x = x.
end_of_list.
```


Here the first line declares that we will use `cup` as an infix operator. This is followed by the axioms of our theory (referred to as “assumptions” by Prover9 and its documentation) using `cup` in place of \cup .¹ By convention, the letters x , y and z denote variables that are implicitly universally quantified.

We can now ask Prover9 to prove theorems based on these assumptions. As a very simple example we can formulate the following goal²:

```
formulas (goals) .
  x cup (x cup x) = x.
end_of_list.
```

If we now execute the following command:

```
Prover9 -f semilattice.ax sl-pr1.gl
```

then Prover9 will almost instantaneously respond with the pleasing message “THEOREM PROVED”. Its output also includes a detailed linear representation of the proof³:

```
1 x cup (x cup x) = x # label(non_clause) # label(goal). [goal].
2 x cup x = x. [assumption].
3 c1 cup (c1 cup c1) != c1. [deny(1)].
4 $F. [copy(5),rewrite([4(4),4(3)]),xx(a)].
```

At first glance this looks daunting, but with a little work it is possible to gain insight from it. The Prover9 proofs are always presented as proofs by contradiction: after stating the goal and the assumptions that will be used to prove it, Prover9 denies the goal by asserting the existing of a constant `c1` that does not satisfy the equation we are trying to prove (`!=` is the Prover9 syntax for \neq). In general, the line that denies the goal will be followed by a sequence of steps each comprising a formula that can be deduced by applying logical inference rules to formulas given in earlier steps. In this case, there is just one step in this sequence, in which some rewriting has arrived at the desired contradiction `$F` (Prover9 syntax for falsehood). In each inference step, the formula is followed by a justification giving a precise description of the inference rules and how they have been applied. There is a program `prooftrans` supplied with Prover9 that can be used to perform some useful transformations on proofs (we have already used it in generating the proof as shown above to renumber the steps in consecutive order). In this case, we can ask for the rewriting steps to be expanded giving:

```
4A c1 cup c1 != c1. [para(2(a,1),3(a,1,2))].
4B c1 != c1. [para(2(a,1),4A(a,1))].
4 $F. [copy(4B),xx(a)].
```

¹Prover9 input uses only ASCII characters.

²See goal script `sl-pr1.gl`.

³See output file `sl-pr1.txt`.

It then quickly finds that repeated rewriting with the idempotency law (the assumption, i.e., axiom stated in step 2) gives the false ($\$F$) conclusion that $c1$ is not equal to itself. The justifications in the Prover9 proofs look complicated because they encode a very detailed description of how each inference rule has been applied. In trying to understand the proof, one can generally ignore most of this detail. The rule names `copy`, `para`⁴ etc. are followed by a list of parameters in brackets identifying the steps that provided the inputs (antecedents) to the rule the list of letters and numbers in brackets that come immediately after the step number identify exactly how the rule was applied to the formula of that step, but in most cases that is obvious and this information can be ignored.

In a semilattice, one defines a relation \geq by

$$x \geq y \iff x \cup y = x.$$

We can formalise this definition in Prover9 syntax as follows (see file `sl-ge-def.ax`):

```
formulas(assumptions).
  x >= y <-> x cup y = x.
end_of_list.
```

We can now get Prover9 to prove some more interesting properties. E.g. the transitivity of \geq ⁵:

```
formulas(goals).
  x >= y & y >= z -> x >= z.
end_of_list.
```

If we execute the command:

```
Prover9 -f semilattice.ax sl-ge-def.ax sl-trans.gl
```

then Prover9 will again quickly respond with “THEOREM PROVED”. In this case the proof it finds has 20 steps.⁶ As always it is a proof by contradiction, but with a little analysis it is easy to extract the elements of a human-readable direct proof from it. As we will see later in this chapter, it is not always easy to extract human-readable proofs from the Prover9 output, but analysis of the machine-generated proofs often leads to useful insights.

The reader may well ask what happens if we ask Prover9 to prove a false conjecture. What if we were to conjecture that the ordering relation on a semilattice is a total order? We would express this goal in Prover9 as follows (see file `sl-total.gl`):

⁴`para` stands for “paramodulation”, an inference rule that performs a form of equational reasoning generalising the usual notion of using an equation to rewrite a term within formula.

⁵See goal script `sl-trans.gl`.

⁶See output file `sl-trans.txt`.

```

formulas(goals).
  x >= y | y >= x.
end_of_list.

```

Here “|” denotes disjunction, and, recalling that variables are implicitly universally quantified, in textbook logical notation the goal is $\forall x \forall y (x \geq y \vee y \geq x)$. If we run Prover9 on this goal it will necessarily fail to deduce a contradiction: it will either fail to terminate or terminate without finding a proof (either because no new formulas can be generated or because it has reached a user-specified bound on execution time or memory usage). In this case the companion program Mace4 does something much more useful. If we execute the command:

```
mace4 -p 1 -f semilattice.ax sl-ge-def.ax sl-total.gl
```

then the Mace4 program will search for a finite semilattice that provides a counter-example to our false conjecture. It quickly finds one and in its log of the search process it prints out the counter-example as follows:

```

cup :
  | 0 1 2
  ---+-----
  0 | 0 2 2
  1 | 2 1 2
  2 | 2 2 2

>= :
  | 0 1 2
  ---+-----
  0 | 1 0 0
  1 | 0 1 0
  2 | 1 1 1

```

Here we see that Mace4 uses 0, 1, 2 . . . to represent the elements of the model, which in this case has size 3. The model is the smallest example of a semilattice that is not totally ordered. It is made up of two incomparable atoms 0 and 1 together with their least upper bound $2 = 0 \cup 1$.

In conjunction with two companion programs `isofilter` and `interpformat`, Mace4 can enumerate all the models of a given size. The `Makefile` contains the commands to do this for all semilattices with at most 5 elements. This is particularly useful when investigating more complex algebraic structures as generating examples by hand is often error prone, particularly if associative operators are involved as associativity is time-consuming to check.

1.2 Investigating the Algebraic Structure of Hoops

In Sect. 1.1, we took semilattices as our running example for reasons of simplicity. However, semilattices are a little too simple to demonstrate the real power of tools such as Prover9 and Mace4. In this section we introduce the algebraic structures called hoops (Blok & Ferreirim, 2000; Bosbach, 1969; Büchi & Owens, 1974) that will provide the running example for the rest of the paper. Hoops are a generalisation of Heyting algebras (used in the study of intuitionistic logic⁷). They are of considerable interest, e.g., in connection with Łukasiewicz logic and fuzzy logic, and there are many difficult open problems concerning them.

These structures were originally investigated, first by Bosbach (1969), and independently by Büchi and Owens (1974). In this section, we present an elementary investigation of hoops using Prover9 and Mace4, and indicate how one might put these tools to use at the level of an undergraduate project assignment.

A hoop⁸ is a structure $(H, 0, 1, \oplus, \ominus)$ satisfying the following axioms:

$$x \oplus (y \oplus z) = (x \oplus y) \oplus z \quad (1)$$

$$x \oplus y = y \oplus x \quad (2)$$

$$x \oplus 0 = x \quad (3)$$

$$x \ominus x = 0 \quad (4)$$

$$(x \ominus y) \ominus z = x \ominus (y \oplus z) \quad (5)$$

$$x \oplus (y \ominus x) = y \oplus (x \ominus y) \quad (6)$$

$$0 \ominus x = 0 \quad (7)$$

$$x \ominus 1 = 0 \quad (8)$$

The example scripts in the supporting material for this section contain a Prover9 formalisation of the hoop axioms (file `hoop-eq-ax`) and various goals for Prover9 and Mace4.

Axioms (1), (2) and (3) are very familiar as the axioms for a commutative monoid with binary operation \oplus and identity element 0. Axioms (4) and (5) are reminiscent of properties of subtraction in a commutative group, but the remaining axioms are less familiar.

Axiom (6) says that the operation \cup defined by $x \cup y = x \oplus (y \ominus x)$ is commutative. Using axioms (3) and (4) we see that this operation is also idempotent. We might conjecture that \cup is also associative, so that it makes any hoop into a semilattice, and Prover9 will readily prove this for us.⁹ The proof is short but intricate. This

⁷In a Heyting algebra one normally uses $x \rightarrow y$ for $y \ominus x$, and $x \wedge y$ for $x \oplus y$.

⁸Strictly speaking this is a *bounded* hoop: an unbounded hoop omits the constant 1 and axiom (8). We are only concerned with bounded hoops in this book, so for brevity, we drop the word “bounded”.

⁹See output file `hp-semilattice.txt`.

Table 1 Hoops of order 2 and 3

Ordering	Operation Tables				Name
	\oplus	0 1	\ominus	0 1	
$0 < 1$	0	0 1	0	0 0	L_2
	1	1 1	1	1 0	
$0 < a < 1$	\oplus	0 a 1	\ominus	0 a 1	L_3
	0	0 a 1	0	0 0 0	
	a	a 1 1	a	a 0 0	
	1	1 1 1	1	1 a 0	
$0 < a < 1$	\oplus	0 a 1	\ominus	0 a 1	$L_2 \sim L_2$
	0	0 a 1	0	0 0 0	
	a	a a 1	a	a 0 0	
	1	1 1 1	1	1 1 0	

semilattice structure induces an ordering on a hoop which turns out to be equivalent to defining $x \geq y$ to hold when¹⁰ $y \ominus x = 0$.

Using Mace4, we can quickly generate examples of hoops. Tables 1 and 2 are based on the Mace4 output and show all hoops with between 2 and 4 elements. Inspection of these tables is very instructive, particularly if one looks at the interactions between the order structure and the algebraic operations.

In all cases, one notes that the elements in the rows and columns of the operation tables for \oplus are in increasing order; in the tables for \ominus the elements in the rows are in decreasing order while the elements in the columns are in increasing order. This suggests the conjecture that \oplus is monotonic in both its operands, while \ominus is monotonic in its right operand and anti-monotonic in its left operand. Prover9 quickly proves this for us.¹¹

Axiom (5) suggests (and inspection of the tables supports) the conjecture that for any x, y and $z, z \geq x \ominus y$ iff $z \oplus y \geq x$. This property, an analogue of one of the laws for manipulating inequalities in an ordered commutative group, is known as the *residuation* property and is quickly proved by Prover.¹²

A structure for the signature $(0, 1, \oplus, \ominus, \geq)$ such that $(0, \oplus, \geq)$ is an ordered commutative monoid with least element 0, greatest element 1 and satisfying the residuation axiom:

$$z \geq x \ominus y \Leftrightarrow z \oplus y \geq x$$

¹⁰See output file hp-ge-sl.txt.

¹¹See output file hp-plus-mono.txt, hp-sub-mono-left.txt and hp-sub-mono-right.txt.

¹²See output file hp-res.txt.

Table 2 Hoops of order 4

Ordering	Operation Tables		Name
	\oplus	\ominus	
$0 < a < b < 1$	0	0	$\mathbf{L}_2 \frown \mathbf{L}_3$
	a	a	
	b	b	
	1	1	
$0 < a < b < 1$	0	0	\mathbf{L}_4
	a	a	
	b	b	
	1	1	
$0 < a < b < 1$	0	0	$\mathbf{L}_3 \frown \mathbf{L}_2$
	a	a	
	b	b	
	1	1	
$0 < a < b < 1$	0	0	$\mathbf{L}_2 \frown \mathbf{L}_2 \frown \mathbf{L}_2$
	a	a	
	b	b	
	1	1	
$0 < a, b < 1$ $a \not\leq b$ $b \not\leq a$	0	0	$\mathbf{L}_2 \times \mathbf{L}_2$
	a	a	
	b	b	
	1	1	

is known as a (bounded) *pocrim*. One might conjecture that any pocrim is a hoop. However this conjecture is false, if we ask Mace4 to enumerate small pocrims,¹³ it finds 2 pocrims with 4 elements that are not hoops, as shown in Table 3. The residuation property is strictly weaker than axiom (6). Inspection of the operation tables reveals the weakness: in a hoop, if $x \geq y$ then $x = y \oplus (x \ominus y)$, but in a pocrim, even when $x \geq y$, we can have $x < y \oplus (x \ominus y)$: in the first example in Table 3, $x \oplus y = 1$ unless one of x and y is 0. However, the axiomatisation of hoops

¹³ See output file `pc-egs.txt`.

Table 3 Pocrims that are not hoops

Ordering	Operation Tables			
	\oplus	0 a b 1	\ominus	0 a b 1
$0 < a < b < 1$	0	0 1 1 1	0	0 0 0 0
	a	a 1 1 1	a	a 0 0 0
	b	b 1 1 1	b	b a 0 0
	1	1 1 1 1	1	1 a a 0
$0 < a < b < 1$	\oplus	0 a b 1	\ominus	0 a b 1
	0	0 a b 1	0	0 0 0 0
	a	a a 1 1	a	a 0 0 0
	b	b 1 1 1	b	b b 0 0
1	1 1 1 1	1	1 b a 0	

via the pocrim axioms together with axiom (6) is often more convenient and intuitive than the purely equational axiomatisation.

Inspection of Tables 1 and 2, shows that for each n , there is a hoop of order n that is linearly ordered and generated by its least nonzero element, in the sense that if a is the least nonzero element, the other nonzero elements are $a \oplus a, a \oplus a \oplus a$, etc. For any n , let us define¹⁴

$$\mathbf{L}_n = (\{0, \frac{1}{n-1}, \frac{2}{n-1}, \dots, 1\}, 0, 1, \oplus, \ominus)$$

where $x \oplus y = \min(x + y, 1)$ and $x \ominus y = \max(x - y, 0)$. Then \mathbf{L}_n is a linearly ordered hoop generated by its least non-zero element $\frac{1}{n-1}$. Copies of \mathbf{L}_n for various n often appear in other hoops. Inspection of the first hoop in Table 2 shows that, the subset $\{0, b, 1\}$ comprises a subhoop isomorphic to \mathbf{L}_3 , while the subset $\{0, a\}$ is isomorphic to \mathbf{L}_2 viewed as a structure for the signature $(0, \oplus, \ominus)$ (i.e., ignoring the fact that $a \neq 1$). This suggests a way of constructing new hoops from old: given hoops \mathbf{H}_1 and \mathbf{H}_2 with underlying sets H_1 and H_2 , we can form what is called the *ordinal sum*, $\mathbf{H}_1 \frown \mathbf{H}_2$ of the two hoops whose underlying set is the disjoint union $H_1 \sqcup (H_2 \setminus \{0\})$, with 0 (resp. 1) given by $0 \in H_1$ (resp. $1 \in H_2$) and with the operation tables defined to extend the operations of \mathbf{H}_1 and \mathbf{H}_2 so that for $h_1 \in H_1$ and $h_2 \in H_2$, $h_1 \oplus h_2 = h_2$, $h_1 \ominus h_2 = 0$ and $h_2 \ominus h_1 = h_2$. The column headed “Name” in Tables 1 and 2 gives an expression for each hoop as an ordinal sum or product of the hoops \mathbf{L}_n .

Linearly ordered hoops are of particular importance. We can see at a glance from Tables 1 and 2 that there is 1 linearly ordered hoop of order 2 and 2 of order 3 (and

¹⁴ We call these hoops \mathbf{L}_n in honour of Łukasiewicz and Tarski (1930) whose multi-valued logics have a natural semantics with values in these hoops.

these are the only hoops of these orders), while there are 4 linearly ordered hoops of order 4 (and just one other). Mace4 tells us¹⁵ that there are 8 linearly ordered hoops of order 5 (and two others¹⁶). This leads us to the following theorem. The proof of this theorem proceeds by induction and Prover9 cannot be expected to find a proof automatically, but if, informed by the examples provided by Mace4, we set up the framework for the inductive proof, then Prover9 will help us with the low-level details.

Theorem 1.1 *For each $n \geq 2$, there are 2^{n-2} isomorphism classes of linearly ordered hoops of order n .*

Proof We claim that any linearly ordered hoop of order n is isomorphic to an ordinal sum $\mathbf{L}_{m_1} \frown \dots \frown \mathbf{L}_{m_k}$ for some $k, m_1, \dots, m_k \geq 2$ such that $m_1 + \dots + m_k - k + 1 = n$. It is easy to see that two such ordinal sums $\mathbf{L}_{m_1} \frown \dots \frown \mathbf{L}_{m_k}$ and $\mathbf{L}_{n_1} \frown \dots \frown \mathbf{L}_{n_l}$ are isomorphic iff $k = l$ and $m_i = n_i$, $1 \leq i \leq k$. Moreover the sequences $\langle m_1, \dots, m_k \rangle$ indexing these ordinal sums are in one-to-one correspondence with the subsets of $\{1, \dots, n - 2\}$ via:

$$\langle m_1, \dots, m_k \rangle \mapsto \{m_1 - 1, m_1 + m_2 - 2, \dots, m_1 + \dots + m_{k-1} - k + 1\},$$

Hence there are, indeed, 2^{n-2} such hoops. We have only to prove the claim which follows immediately by induction from the observation that any finite linearly ordered hoop \mathbf{H} is isomorphic to $\mathbf{L}_m \frown \mathbf{K}$ for some m and some subhoop \mathbf{K} of \mathbf{H} . This observation may be proved by considering the subhoop generated by the least non-zero element of \mathbf{H} using the fact that (in any hoop), if $x + x = x$ and $y \geq x$, then $x + y = y$, which Prover9 will readily verify for us.¹⁷

This theorem and its proof are a nice example of synergy between automated methods and the traditional approach: Mace4 provides examples that suggest a general conjecture and indicate a possible line of proof by induction. While it is not able to automate the inductive proof,¹⁸ Prover9 can help us fill in tricky algebraic details.

We encourage readers interested in using tools such as Prover9 and Mace4 in undergraduate teaching to use the example scripts we have provided to inform the design of an undergraduate project involving a guided investigation of a more familiar class of algebraic structure, say Boolean algebras or Heyting algebras using these tools.

¹⁵See output file `hp-linear-egs.txt`.

¹⁶See output file `hp-egs.txt`.

¹⁷See output file `hp-sum-lemma.txt`.

¹⁸Prover9 is a theorem-prover for finitely axiomatisable first-order theories: it is not designed to work with something like the principle of induction that can only be expressed either as an infinite axiom schema or as a second-order property. The use of interactive proof assistants that can handle induction is of potential interest in mathematics education, but is not the focus of the present chapter. There has been research on fully automated proof in higher-order logic, but this is in its early days.

1. Start with a new conjecture ϕ
2. Use Mace4 to check ϕ does not have trivial (small) counterexamples
3. User Prover9 to search for a proof of ϕ
 - (a) Once proof found, mine the proof for new “concepts” and “properties”
 - (b) Rerun the proof search taking these new concepts and properties as given
 - (c) Use knowledge learned, formulate new conjecture, and go back to 1.

Fig. 1 Methodology

2 Analysing Larger Proofs

In the second part of this chapter we discuss, using the theory of hoops as a running example, how we have used Prover9 and Mace4 to explore new conjectures, and the methodology we used to analyse and “understand” Prover9’s machine generated proofs. The main challenge we want to focus on is in dealing with large Prover9 proofs, and how one should go about breaking these proofs into smaller, more intuitive and understandable steps. As a general methodology, we have adopted the process described in Fig. 1.

2.1 A Homomorphism Property for Hoops

As mentioned in Sect. 1.2, hoops generalise Heyting algebras. Defining the dual of an element as $x^\perp = 1 \ominus x$, we have that in Heyting algebras the double-dual operation $x \mapsto x^{\perp\perp}$ is a homomorphism. The conjecture ϕ we were working on, was whether this was also the case for hoops, i.e. do the following two homomorphism properties hold:

$$(x \ominus y)^{\perp\perp} = x^{\perp\perp} \ominus y^{\perp\perp} \tag{9}$$

and

$$(x \oplus y)^{\perp\perp} = x^{\perp\perp} \oplus y^{\perp\perp} \tag{10}$$

Using Mace4 we were able to check in just a few minutes that no small (size 20 or below) counter-examples existed.¹⁹ To our surprise, Prover9 found a proof of (9)

¹⁹See output files `conjectureNNSNNSNN.txt` and `conjecturePNNNNPNN.txt`.

is just over 100 min.²⁰ This proof, however, is not as short as the ones we have seen in the previous section, involving around 177 steps.

2.2 *Discovering Derived Operations and Their Basic Properties*

When faced with a long Prover9 derivation such as the one above, we tried to identify new concepts and intermediate steps in the proof that had intrinsic value, and could be understood in isolation. For instance, we noticed that the patterns $x^\perp \oplus (x \ominus y)$ and $x \ominus (x \ominus y)$ appeared multiple times in the derivation. This led us to introduce new operations so that multiple steps in the proof could be understood as properties of these new operations. In total we found, apart from $x \cup y$, three further new derived operations:

$$\begin{aligned} x \cup y &\equiv x \oplus (y \ominus x) \\ x \cap y &\equiv x \ominus (x \ominus y) \\ y \setminus x &\equiv (x \oplus y) \ominus x \\ x \downarrow y &\equiv x^\perp \oplus (x \ominus y) \end{aligned}$$

Our final choice of notation for these new operations came after we had studied their properties. The Prover9 symbols we used for these (in ASCII) are shown in Table 4. When identifying these operations we also used our knowledge of the correspondence between hoops and Heyting algebras. For instance, $x \cap y$ in logical terms corresponds to $(y \rightarrow x) \rightarrow x$, which generalises double negation and in theoretical computer science is known as the *continuation monad* (Moggi, 1989).

So, according to step 3. (a) and (b) of our methodology, we looked first for basic properties of these new operations, or of their relation with the primitive operations. We come up with six simple properties (listed in the following lemma) that we then added as axioms, and rerun the proof search.

Lemma 2.1 *The following hold in all hoops:*

- (i) $x \geq y \cap x$
- (ii) $x \geq x \setminus y$
- (iii) $(x \setminus y) \ominus x = 0$
- (iv) $x \oplus y = x \oplus (y \setminus x)$
- (v) $z \cap (y \ominus x) \geq (z \cap y) \ominus (z \cap x)$
- (vi) $x \ominus (x \cap y) = x \ominus y$

Adding these lemmas cut the proof search time to just over 10 min,²¹ and the number of steps to 132. This is still a reasonably large proof, which would be hard

²⁰See output file `theoremNNSNNSNN-eq-expanded.txt`.

²¹See output file `theoremNNSNNSNN-eq-basic-lemmas.txt`.

to “understand” as a whole. So we continued looking for more complex properties of these new defined operations. This time we focused on the following four steps in the proof script `theoremNNSNNSNN-eq-basic-lemmas.txt`, and noticed that these do have intrinsic value:

$$126 \quad (x \sim y) + (1 \sim x) = 1 \sim (x \sim (x \sim y)).$$

states a duality between $x \downarrow y$ and $x \cap y$, i.e. $x \downarrow y = (x \cap y)^\perp$.

$$132 \quad (x \sim y) + (1 \sim x) = (y \sim x) + (1 \sim y).$$

states the commutativity of $x \downarrow y$, i.e. $x \downarrow y = y \downarrow x$.

$$134 \quad 1 \sim (x \sim (x \sim y)) = 1 \sim (y \sim (y \sim x)).$$

states the commutativity of $x \cap y$ under $(\cdot)^\perp$, i.e. $(x \cap y)^\perp = (y \cap x)^\perp$.

$$153 \quad 1 \sim (x \sim (1 \sim (1 \sim x))) = 1.$$

immediately implies that although $x \ominus x^{\perp\perp}$ is not 0 in general, we do have that $(x \ominus x^{\perp\perp})^{\perp\perp} = 0$; and, as we will see, this is the crucial lemma in the proof of (9).

In order to emphasise how we were able to break this long proof into a small collection of simple lemmas (each with a reasonably short proof), we will explicitly give the proof of these lemmas, and the proof of the conjecture from these lemmas. Readers who are not planning to undertake this kind of work themselves are invited to skip the details. We believe, however, that the details will be helpful to those wanting to apply a similar methodology in other contexts.

Notation. But before we do that, let us set up a notation for naming hoop properties. We associate a letter to each of the operations as shown in Table 4 and name each property by reading all the operations on the statement of the property from left to right. For instance, the property $x \downarrow y = y \downarrow x$ is named AA. Although there is a risk that two different properties will end up with the same name, this is not the case for the properties we consider here.

2.3 Discovering Basic Properties

The first set of basic properties we discovered relate to commutativity. One of the hoop axioms states that $x \cup y$ is commutative. It is easy to construct a model, however, which shows that $x \cap y$ is not commutative in general. When analysing proofs generated by Prover9 we spotted two other interesting commutativity properties, alluded to above. The first (which we call AA as discussed above) is that $x \downarrow y$ also satisfies the commutativity property:

Table 4 Nomenclature

Operator	Prover9	Letter	Intuition
0	0	Z	(zero)
1	1	O	(one)
\oplus	+	P	(plus)
\ominus	\sim	S	(subtraction)
\cup	cup	J	(join)
\cap	cap	M	(meet)
\setminus	\setminus	D	(difference)
\downarrow	nand	A	(ampheck) ^a
$(\cdot)^\perp$	$(\cdot)'$	N	(negation)

^a“Ampheck” from a Greek word meaning “cutting both ways” was the name coined by C. S. Peirce for the logical NAND operation

Lemma 2.2 (AA) $x \downarrow y = y \downarrow x$

Proof By symmetry it is enough to prove $x \downarrow y \geq y \downarrow x$. Note that $((y \ominus x) \ominus x^\perp) = 0$, hence:

$$\begin{aligned}
 (x \ominus y) \oplus x^\perp &= (x \ominus y) \oplus x^\perp \oplus ((y \ominus x) \ominus x^\perp) && \text{Axiom (3)} \\
 &= (x \ominus y) \oplus (y \ominus x) \oplus (x^\perp \ominus (y \ominus x)) && \text{Axiom (6)} \\
 &= (x \ominus y) \oplus (y \ominus x) \oplus ((y \ominus x) \oplus x)^\perp && \text{Axiom (5)} \\
 &= (x \ominus y) \oplus (y \ominus x) \oplus (y \oplus (x \ominus y))^\perp && \text{Axiom (6)} \\
 &= (y \ominus x) \oplus (x \ominus y) \oplus (y^\perp \ominus (x \ominus y)) && \text{Axiom (5)} \\
 &= (y \ominus x) \oplus y^\perp \oplus ((x \ominus y) \ominus y^\perp) && \text{Axiom (6)} \\
 &\geq (y \ominus x) \oplus y^\perp. && \text{Monotonicity}
 \end{aligned}$$

We also identified this interesting duality between $x \cap y$ and $x \downarrow y$:

Lemma 2.3 (MNA) $(x \cap y)^\perp = x \downarrow y$

Proof By Lemma 2.1 (i) we have $y \geq x \cap y$; and by [EFQ] we have $1 \geq x$. Hence, $(x \cap y)^\perp \geq x \ominus y$ so that (*) $(x \ominus y) \ominus (x \cap y)^\perp = 0$. Clearly we also have that (†) $(x \ominus y) \ominus x = 0$. Therefore

$$\begin{aligned}
 (x \cap y)^\perp &= (x \cap y)^\perp \oplus ((x \ominus y) \ominus (x \cap y)^\perp) && (*) \\
 &= (x \ominus y) \oplus ((x \cap y)^\perp \ominus (x \ominus y)) && \text{Axiom (6)} \\
 &= (x \ominus y) \oplus ((x \ominus y) \oplus (x \cap y))^\perp && \text{Axiom (5)} \\
 &= (x \ominus y) \oplus (x \oplus ((x \ominus y) \ominus x))^\perp && \text{Axiom (6)} \\
 &= (x \ominus y) \oplus x^\perp. && (\dagger)
 \end{aligned}$$

This duality when combined with Lemma 2.2 immediately implies that $x \cap y$ is also commutative when in the following weaker form:

Lemma 2.4 (MNMN) $(x \cap y)^\perp = (y \cap x)^\perp$

Note that we make the point of giving the full proofs of all the lemmas in order to emphasise our goal of reducing the overall proof to a sequence of simple yet interesting lemmas, each of which should have reasonably short proofs (around 10 steps). The other steps in the proof `theoremNNSNNSNN-eq-basic-lemmas.txt` which we found of interest were 101, 138 and 144, which again all have short proofs as follows:

Lemma 2.5 (NPJSSO) $x^\perp \oplus ((y \cup x) \ominus (y \ominus x)) = 1$

Proof Note that (*) $x^\perp = (y \ominus x) \oplus (x^\perp \ominus (y \ominus x))$. We have

$$\begin{aligned}
 1 &\geq x^\perp \oplus ((y \cup x) \ominus (y \ominus x)) && \text{Axiom (8)} \\
 &= x^\perp \oplus ((y \oplus (x \ominus y)) \ominus (y \ominus x)) && \text{Def. } \cup \\
 &= (y \ominus x) \oplus (x^\perp \ominus (y \ominus x)) \oplus ((y \oplus (x \ominus y)) \ominus (y \ominus x)) && (*) \\
 &= y \oplus (x \ominus y) \oplus (x^\perp \ominus (y \ominus x)) \oplus ((y \ominus x) \ominus (y \oplus (x \ominus y))) && \text{Axiom (6)} \\
 &= x \oplus (y \ominus x) \oplus (x^\perp \ominus (y \ominus x)) \oplus ((y \ominus x) \ominus (y \oplus (x \ominus y))) && \text{Axiom (6)} \\
 &\geq x \oplus (y \ominus x) \oplus (x^\perp \ominus (y \ominus x)) && \text{Monotonicity} \\
 &= x \oplus x^\perp \oplus ((y \ominus x) \ominus x^\perp) && \text{Axiom (6)} \\
 &\geq x \oplus x^\perp && \text{Monotonicity} \\
 &\geq 1. && \text{Residuation}
 \end{aligned}$$

Lemma 2.6 (NSNSM) $x^\perp = (x \ominus y)^\perp \ominus (y \cap x)$

Proof Note that (*) $(x \ominus y) \ominus x = 0$. Hence

$$\begin{aligned}
 x^\perp &= (x \oplus ((x \ominus y) \ominus x))^\perp && (*) \\
 &= ((x \ominus y) \oplus (x \ominus (x \ominus y)))^\perp && \text{Axiom (6)} \\
 &= ((x \ominus y) \oplus (x \cap y))^\perp && \text{Def. } \cap \\
 &= (x \cap y)^\perp \ominus (x \ominus y) && \text{Axiom (5)} \\
 &= (y \cap x)^\perp \ominus (x \ominus y) && \text{Lemma 2.4} \\
 &= (x \ominus y)^\perp \ominus (y \cap x). && \text{Axioms (2) and (5)}
 \end{aligned}$$

Lemma 2.7 (NNSNNSNN) $x^\perp = x^\perp \ominus (x \ominus x^{\perp\perp})$

Proof It is easy to show that (*) $x \oplus (x^\perp \ominus (x \ominus x^{\perp\perp})) = 1$. Let us use the abbreviation $X = x \ominus x^{\perp\perp}$. It is also easy to see that (†) $((X^\perp \ominus x) \ominus ((x \oplus (x^\perp \ominus X)) \ominus x)) = 0$. Hence

$$\begin{aligned}
x^\perp &= 1 \ominus x && \text{Def. } (\cdot)^\perp \\
&= (x \oplus (x^\perp \ominus (x \ominus x^{\perp\perp}))) \ominus x && (*) \\
&= ((x \oplus (x^\perp \ominus (x \ominus x^{\perp\perp}))) \ominus x) \oplus ((X^\perp \ominus x) \ominus ((x \oplus (x^\perp \ominus X)) \ominus x)) && (\dagger) \\
&= ((x \ominus x^{\perp\perp})^\perp \ominus x) \oplus (((x \oplus (x^\perp \ominus X)) \ominus x) \ominus (X^\perp \ominus x)) && \text{Axiom (6)} \\
&= (x \ominus x^{\perp\perp})^\perp \ominus x && \text{Axiom (5)} \\
&= x^\perp \ominus (x \ominus x^{\perp\perp}). && \text{Axiom (5)}
\end{aligned}$$

The above three lemmas are interesting, in the sense that it describes properties of the duality operation x^\perp , either showing equivalent ways of writing x^\perp , or how it relates to other complex expressions.

Finally, the crucial lemma of the proof shows that, although $x \ominus x^{\perp\perp} \neq 0$ in general, we always have $1 \ominus (x \ominus x^{\perp\perp}) = 1$.

Lemma 2.8 (SNNNO) $(x \ominus x^{\perp\perp})^\perp = 1$

Proof Note that $(*) x^\perp \oplus ((x \ominus x^\perp) \ominus x^{\perp\perp}) = x^\perp$ since $(x \ominus x^\perp) \ominus x^{\perp\perp} = 0$. Hence,

$$\begin{aligned}
(x \ominus x^{\perp\perp})^\perp &= (x \ominus x^{\perp\perp})^\perp \oplus (x^\perp \ominus x^\perp) && \text{Easy} \\
&= (x \ominus x^{\perp\perp})^\perp \oplus ((x^\perp \oplus ((x \ominus x^\perp) \ominus x^{\perp\perp})) \ominus x^\perp) && (*) \\
&= (x \ominus x^{\perp\perp})^\perp \oplus ((x^\perp \oplus ((x \ominus x^{\perp\perp}) \ominus x^\perp)) \ominus x^\perp) && \text{Axioms (2) and (5)} \\
&= (x \ominus x^{\perp\perp})^\perp \oplus ((x^\perp \cup (x \ominus x^{\perp\perp})) \ominus x^\perp) && \text{Def. } \cup \\
&= (x \ominus x^{\perp\perp})^\perp \oplus ((x^\perp \cup (x \ominus x^{\perp\perp})) \ominus x^\perp) && \text{Def. } \cup \\
&= (x \ominus x^{\perp\perp})^\perp \oplus ((x^\perp \cup (x \ominus x^{\perp\perp})) \ominus (x^\perp \ominus (x \ominus x^{\perp\perp}))) && \text{Lemma 2.7} \\
&= 1. && \text{Lemma 2.5}
\end{aligned}$$

Lemma 2.8 immediately implies step 159 of theoremNNSNNSNN-eq-basic-lemmas.txt, namely:

Lemma 2.9 (SSNNSNO) $((x \ominus y) \ominus (x^{\perp\perp} \ominus y))^{\perp\perp} = 1$

Proof We have

$$\begin{aligned}
1 &= (x \ominus x^{\perp\perp})^\perp && \text{Lemma 2.8} \\
&\leq ((x \ominus x^{\perp\perp}) \ominus (y \ominus x^{\perp\perp}))^\perp && \text{Monotonicity} \\
&= (x \ominus (x^{\perp\perp} \oplus (y \ominus x^{\perp\perp})))^\perp && \text{Axiom (5)}
\end{aligned}$$

$$\begin{aligned}
&= (x \ominus (y \oplus (x^{\perp\perp} \ominus y)))^{\perp} && \text{Axiom (6)} \\
&= ((x \ominus y) \ominus (x^{\perp\perp} \ominus y))^{\perp} && \text{Axiom (5)} \\
&\leq 1
\end{aligned}$$

2.4 Producing a Human-Readable Proof of (9)

We are now in a position where we can derive a human-readable proof of the homomorphism property (9) using the lemmas of the previous section. Our proof is based on the one in the proof script `theoremNNSNNSNN-eq-expanded.txt`.

Theorem 2.10 (NNSNNSNN) $x^{\perp\perp} \ominus y^{\perp\perp} = (x \ominus y)^{\perp\perp}$

Proof Since $(x^{\perp\perp} \ominus y) \ominus (x \ominus y) = 0$ it follows that (*) $(x^{\perp\perp} \ominus y) \cap (x \ominus y) = x^{\perp\perp} \ominus y$. Hence

$$\begin{aligned}
x^{\perp\perp} \ominus y^{\perp\perp} &= (x^{\perp\perp} \ominus y)^{\perp\perp} && \text{Lemma 2.1 (vi)} \\
&= (1 \ominus (x^{\perp\perp} \ominus y))^{\perp} && \text{Def. } (\cdot)^{\perp} \\
&= (1 \ominus ((x^{\perp\perp} \ominus y) \cap (x \ominus y)))^{\perp} && (*) \\
&= (((x \ominus y) \ominus (x^{\perp\perp} \ominus y))^{\perp} \ominus ((x^{\perp\perp} \ominus y) \cap (x \ominus y)))^{\perp} && \text{Lemma 2.9} \\
&= (x \ominus y)^{\perp\perp}. && \text{Lemma 2.6}
\end{aligned}$$

2.5 Tackling the Harder Conjecture (10)

We have also been able to produce a human-readable proof of (10), although this seems to be a much harder result. Prover9 is also able to find a proof of (10) from the basic hoop axioms, but that takes almost 7 hours, and requires 624 steps.²² In a process of “mining” this proof for new lemmas, and then searching for a proof again using these lemmas, we were able to find a small set of lemmas from which Prover9 derives the proof of the conjecture in just a fraction of a second.²³ Again, in order to emphasise how we were able to produce a human-like mathematical presentation of this proof, we prove here these other lemmas in full, and give the (short) proof of (10) from these lemmas.

The first two lemmas enable us to rewrite x or x^{\perp} as a sum of two other elements. In a hoop we do not have idempotence ($x = x \oplus x$) in general, but we can obtain weaker forms of this as follows:

²²See file `theoremPNNNNPNN-eq.txt`.

²³See file `theoremPNNNNPNN-eq-lemmas.txt`.

Lemma 2.11 (MPS) $x = (x \cap y) \oplus (x \ominus y)$

Proof We have

$$\begin{aligned} x &= x \oplus ((x \ominus y) \ominus x) && \text{Axiom (3)} \\ &= (x \ominus (x \ominus y)) \oplus (x \ominus y) && \text{Axiom (6)} \\ &= (x \cap y) \oplus (x \ominus y). && \text{Def. } \cap \end{aligned}$$

Lemma 2.12 (NSPJN) $x^\perp = (y \ominus x) \oplus (x \cup y)^\perp$

Proof That $x^\perp \geq (y \ominus x) \oplus (x \cup y)^\perp$ follows directly since $(x \cup y)^\perp = (y \ominus x)^\perp \ominus x$. For the other direction we have:

$$\begin{aligned} x^\perp &= x^\perp \oplus (y \ominus 1) && \text{Axiom (8)} \\ &\geq x^\perp \oplus ((y \ominus x) \ominus x^\perp) && \text{Easy} \\ &= (y \ominus x) \oplus (x^\perp \ominus (y \ominus x)) && \text{Axiom (6)} \\ &= (y \ominus x) \oplus (x \oplus (y \ominus x))^\perp && \text{Axiom (5)} \\ &= (y \ominus x) \oplus (x \cup y)^\perp. && \text{Def. } \cup \end{aligned}$$

The following two lemmas can be seen as properties relating $x \oplus y$ with the new derived connectives $x \cup y$, $x \cap y$ and $x \setminus y$.

Lemma 2.13 (PPMD) $x \oplus y = x \oplus (y \cap (y \setminus x))$

Proof Note that (*) $(y \ominus (y \setminus x)) \ominus x = 0$. Hence

$$\begin{aligned} x \oplus y &= (y \setminus x) \oplus x && \text{Lemma 2.1 (iv)} \\ &= (y \setminus x) \oplus x \oplus ((y \ominus (y \setminus x)) \ominus x) && (*) \\ &= (y \setminus x) \oplus (y \ominus (y \setminus x)) \oplus (x \ominus (y \ominus (y \setminus x))) && \text{Axiom (6)} \\ &= y \oplus ((y \setminus x) \ominus y) \oplus (x \ominus (y \ominus (y \setminus x))) && \text{Axiom (6)} \\ &= y \oplus (x \ominus (y \ominus (y \setminus x))) && \text{Lemma 2.1 (iii)} \\ &= (y \cap (y \setminus x)) \oplus (y \ominus (y \setminus x)) \oplus (x \ominus (y \ominus (y \setminus x))) && \text{Lemma 2.11} \\ &= (y \cap (y \setminus x)) \oplus x \oplus ((y \ominus (y \setminus x)) \ominus x) && \text{Axiom (6)} \\ &= x \oplus (y \cap (y \setminus x)). && \text{Monotonicity} \end{aligned}$$

Lemma 2.14 (NPNPM) $x^\perp \oplus y = x^\perp \oplus (y \cap x)$

Proof That $x^\perp \oplus y \geq x^\perp \oplus (y \cap x)$ follows directly from $y \geq y \cap x$. For the other direction, note that $x \geq y \setminus x^\perp$. Hence, $y \cap x \geq y \cap (y \setminus x^\perp)$. Therefore, the result follows directly from Lemma 2.13.

The above lemmas give us another interesting and useful duality between the two defined operations $x \cap y$ and $x \setminus y$:

Lemma 2.15 (JNND) $(x \cup y)^\perp = y^\perp \setminus x$

Proof That $y^\perp \setminus x \geq (x \cup y)^\perp$ is easy to show. For the converse, observe that by Lemma 2.14 (with y and x interchanged) it follows that (*) $y^\perp \geq (x + y^\perp) \ominus (x \cap y)$. Hence

$$\begin{aligned}
 (x \cup y)^\perp &= (x \oplus (y \ominus x))^\perp && \text{Def. } \cup \\
 &= (y \oplus (x \ominus y))^\perp && \text{Axiom (6)} \\
 &= y^\perp \ominus (x \ominus y) && \text{Axiom (5)} \\
 &\geq ((x \oplus y^\perp) \ominus (x \cap y)) \ominus (x \ominus y) && (*) \\
 &= ((x \oplus y^\perp) \ominus ((x \ominus y) \ominus x)) \ominus x && \text{Axiom (6)} \\
 &= (x \oplus y^\perp) \ominus x && \text{Easy} \\
 &= y^\perp \setminus x. && \text{Def. } \setminus
 \end{aligned}$$

The above, together with Lemma 2.15, immediately gives us another interesting commutativity property.

Corollary 2.16 (NDND) $y^\perp \setminus x = x^\perp \setminus y$

The final main lemma in the proof of (10) is a surprising duality between \ominus and \oplus .

Lemma 2.17 (SNNPN) $(y \ominus x^\perp)^\perp = x^\perp \oplus y^\perp$

Proof That $x^\perp \oplus y^\perp \geq (y \ominus x^\perp)^\perp$ is easy to derive. For the converse, note that we have $x^{\perp\perp} \geq y \ominus x^\perp$, and hence

$$(*) (x^{\perp\perp} \oplus (y \ominus x^\perp)^\perp) \ominus x^{\perp\perp} \geq (x^{\perp\perp} \oplus x^{\perp\perp\perp}) \ominus x^{\perp\perp} = x^{\perp\perp\perp}.$$

Hence, taking $x' = y \ominus x^\perp$ and $y' = x^{\perp\perp}$ in Lemma 2.12, we have the first line of the following chain

$$\begin{aligned}
 (y \ominus x^\perp)^\perp &= (x^{\perp\perp} \ominus (y \ominus x^\perp)) \oplus ((y \ominus x^\perp) \cup x^{\perp\perp})^\perp && \text{Lemma 2.12} \\
 &= (x^{\perp\perp} \ominus (y \ominus x^\perp)) \oplus ((y \ominus x^\perp)^\perp \setminus x^{\perp\perp}) && \text{Theorem 2.15} \\
 &\geq (x^{\perp\perp} \ominus (y \ominus x^\perp)) \oplus x^{\perp\perp\perp} && (*) \\
 &= (x^\perp \oplus (y \ominus x^\perp))^\perp \oplus x^\perp && \text{Lemma 2.1 (vi)} \\
 &= (x^\perp \cup y)^\perp \oplus x^\perp && \text{Def } \cup \\
 &= (x^{\perp\perp} \setminus y) \oplus x^\perp && \text{Theorem 2.15} \\
 &= (y^\perp \setminus x^\perp) \oplus x^\perp && \text{Corollary 2.16 (i)} \\
 &\geq x^\perp \oplus y^\perp. && \text{Residuation}
 \end{aligned}$$

Lemma 2.17 above, immediately implies the homomorphism property for $x \oplus y$, since $(x \oplus y)^{\perp\perp} = (y^{\perp} \ominus x^{\perp\perp})^{\perp}$.

Theorem 2.18 (PNNNNPNN) $(x \oplus y)^{\perp\perp} = x^{\perp\perp} \oplus y^{\perp\perp}$

Remark 2.1 It is interesting to observe that in the proof `theoremPNNNNPNN-eq.txt` of property (10) one also finds some of the lemmas used in the proof of (9), for instance, Lemmas 2.3 (step 329) and 2.12 (step 486), but more interestingly, it also discovers Lemma 2.17 (step 633), and uses that to derive a more general duality between \ominus and \oplus (step 683), namely

$$(x \ominus y)^{\perp} = x^{\perp} \oplus y^{\perp\perp}$$

an interesting property, as in the absence of idempotence ($x = x \oplus x$), i.e., in a hoop that is not a Heyting algebra, it is usually hard to find non-trivial equivalences between non-sums and sums.

3 Concluding Remarks

In Sect. 1 we have attempted to introduce the tools and methods we have been using by examples at the level of an undergraduate project. We hope this is of interest to educators and advocate introduction of tools such as Prover9 and Mace4 into mathematical curricula.

At a more advanced level, we have discussed our own research using Prover9 and Mace4 to investigate algebraic structures. It is possible to demonstrate the provability of properties like duality, commutativity or homomorphism properties by model-theoretic methods but these methods are not constructive, whereas the methods discussed in Sect. 2 construct explicit equational proofs.

In De Villiers (1990), argues that proof has many purposes apart from verification, including explanation, systematization, intellectual challenge, discovery and communication. Tools such as Prover9 automate the process of discovering a proof, but at first glance, the proofs that are discovered seem inaccessible to a human reader. We take this as an intellectual challenge in its own right and claim that with human effort, judiciously applied, we can “mine” explanative and systematic human-oriented proofs from machine-generated ones, potentially leading to new insights into the problem domain.

We have deliberately avoiding discussing potential developments of the tools in the main body of this chapter. However, there are several obvious areas for future investigation. Some automated support for refactoring the machine-generated proofs could be very helpful. The refactoring steps of interest would include separating out lemmas and retrofitting derived notations. It is certainly of interest to speculate on possibilities for fully automating extraction of human-readable proofs from machine-generated proofs, but we view this as a hard challenge for Artificial Intelligence.

References

- Blok, W. J., & Ferreirim, I. M. A. (2000). On the structure of hoops. *Algebra Universalis*, 43(2–3), 233–257.
- Bosbach, B. (1969). Komplementäre Halbgruppen. *Axiomatik und Arithmetik. Fundamenta Mathematicae*, 64, 257–287.
- Büchi, J. R., & Owens, T. M. (1974). Complemented monoids and hoops. Unpublished manuscript.
- Burris, S. (1997). An Anthropomorphized Version of McCune’s machine proof that Robbins Algebras are Boolean algebras. Private communication.
- Dahn, B. I. (1998). Robbins algebras are Boolean: A revision of McCune’s computer-generated solution of Robbins problem. *Journal of Algebra*, 208(2), 526–532.
- De Villiers, M. (1990). The role and function of proof in mathematics. *Pythagoras*, 24, 17–24.
- Łukasiewicz, J., & Tarski, A. (1930). Untersuchungen über den Aussagenkalkül. *C. R. Soc. Sc. Varsovie*, 23(1930), 30–50.
- McCune, W. (2005–2010). Prover9 and Mace4. <http://www.cs.unm.edu/~mccune/prover9/>.
- McCune, W. (1997). Solution of the Robbins problem. *Journal of Automated Reasoning*, 19(3), 263–276.
- Moggi, E. (1989). Computational lambda-calculus and monads. In *Symposium of Logic in Computer Science*, California, June 1989. IEEE.
- Winker, S. (1992). Absorption and idempotency criteria for a problem in near-Boolean algebras. *Journal of Algebra*, 153(2), 414–423.

Theoretical Perspectives on Computer-Assisted Proving

Didactical Issues at the Interface of Mathematics and Computer Science



Viviane Durand-Guerrier, Antoine Meyer and Simon Modeste

1 Introduction

The work supporting this chapter takes place in the context of the ongoing research project *DEMaIn* (Didactics and Epistemology of interactions between Mathematics and Informatics), funded by the French ANR (National Agency for Research). This project addresses the epistemology and the didactics of the relations between mathematics and computer science. Its aim is to gain a better understanding of the relations between these two disciplines by studying the foundations, objects, methods, types of questions and modes of thinking which they may share, or which may be specific to one of them. It also proposes to consider the questions that each field asks the other, and the uses that they may find for each other (as a tool or as an object of study).

The DEMaIn project has two main axes. The first deals with the scientific foundations of mathematics and computer science, in particular regarding logic, algorithms, language and proof. Indeed, thinking of the relationships between mathematics and computer science from an educational perspective leads to taking into consideration, among other questions, issues regarding proofs (seen as scientific texts) and proving (the activity of producing such texts) in both domains, and to identifying the role of logic as a possible lens through which to examine and hopefully better understand their interactions.

This chapter is structured as follows. In Section 2, we provide some additional context and motivation. In Section 3, we highlight a few key aspects of the logical

V. Durand-Guerrier (✉) · S. Modeste
IMAG, Univ Montpellier, CNRS, Montpellier, France
e-mail: viviane.durand-guerrier@umontpellier.fr

S. Modeste
e-mail: simon.modeste@umontpellier.fr

A. Meyer
LIGM (UMR 8049), UPEM, CNRS, ESIEE, ENPC, Université Paris-Est,
Marne-la-Vallée, France
e-mail: antoine.meyer@u-pem.fr

issues in mathematics and computer sciences. In Section 4, we analyze several ways in which algorithms and mathematical proof might interact in an educational context.

2 Motivation and Context

2.1 *The Necessity of Epistemological Insights for Didactical Work*

According to Howson and Kahane (1986), the relationship between mathematics and computer science—especially the influence of computer science in mathematics and the role of mathematics in computer science—is an epistemological and didactical issue that transcends school systems and national contexts. The use of computer tools in the teaching of mathematics and informatics, raises questions about the nature of these tools. This can be connected to the particular role played by mathematics in computer science, the proximity of some aspects of both disciplines and the common nature of some of their questions. For example, in a didactical perspective, is it reasonable to use a chart plotter without questioning the accuracy of calculations or that of the display on the screen? Can we use dynamic geometry software without asking how exactly an intersection or a symmetry are built? Can we simulate random experiments without questioning how a machine can produce, or at least imitate, randomness? Can we implement a numerical or formal calculation without asking how a computer can interpret it or, on the contrary, why it rejects it? Can we design a long program without asking how we can make sure it does not contain errors?

The relationships between Mathematics and Computer science are deep and complex. According to Chabert (1999), they share objects, foundations and a part of their history. Indeed, computer science finds much of its theoretical and practical underpinnings in mathematics and has partly built itself as a branch of applied mathematics and logic before emancipating. In this respect, logic plays an important role in the interaction between mathematics and computer science. According to Sinaceur (1991b), logic (in line with Tarski's development) can be considered as an “effective epistemology” providing means for analysing mathematical practices and hence for understanding mathematical activity (op. cit. pp. 341–342). She also stressed that logic became, through computer science, an applied science, which echoes Aristotle's view of logic as an *Organon*. In Sect. 2, we will present the main logical issues in mathematics and computer science that we identify as relevant for our work.

Several authors consider that computer science raises new questions in mathematics, opens up new areas of research and enriches some traditional fields of mathematics (Colton, 2007; Kahane, 2002). Main aspects concern the modes of validation in mathematics through proofs such as those of the four-color theorem or Kepler's conjecture (e.g., Borwein, 2012), the value of the results by questioning the place of constructive proofs and algorithms (e.g., Basu, 2006), and their methods, in particular concerning the experimental dimension of mathematics (e.g., Perrin,

2007, 2012, 2012). New fields of mathematics such as discrete mathematics and theoretical computer science are developing at the interface between mathematics and computer science. This questions mathematicians and didacticians about how these fields should be passed on to teaching (see, Grenier and Payan, 1998; Hart, 1998; Lovasz, 2007; Ouvrier-Bufferet, 2014).

Following Modeste (2016) who studied the introduction of algorithmic in high school in France, we formulate the hypothesis that an introduction of numerical tools or computer science elements in curricula without significant consideration of the epistemology of computer science, mathematics and their links, neither allows nor participates in an in-depth renewal of mathematics education, nor answers the problems of mathematics and computer science mentioned above. The increasing introduction of computer science elements in the teaching of mathematics in the curricula of various countries and in mathematics themselves, supports the importance and urgency of an epistemological and didactic study of interactions between mathematics and computer science.

2.2 Institutional Context in France

We present here some specifics of the teaching of computer science, algorithms and programming in French public schools. This section summarizes elements developed in Guedet et al. (2017).

In the 1980s, in line with an international dynamic (Howson and Kahane, 1986), an optional teaching of computer science centered on algorithms and programming was introduced in upper secondary school in France. However there was at the time no social consensus in the country on the purpose and importance of this teaching (Baron and Bruillard, 2011), and computer science disappeared as a school discipline in the 1990s. It was replaced in curricula by a somewhat informal initiation to what is nowadays often referred to as *digital literacy*, namely the set of abilities allowing one to use computers and technology as *tools* for various purposes. These contents were referred to in France as *transversal* to underline the fact that they were not perceived as forming a standalone topic, but their teaching was rather spread amongst several disciplines (and assumed usually by non-specialised teachers).

In the 2000s, the CREM¹ (Kahane, 2002) advocated for the introduction of elements of computer science in mathematics school curricula and teachers' education, and defended the importance of interactions between mathematics and computer science, relying on the following arguments:

¹Commission de Réflexion sur l'Enseignement des Mathématiques, National Commission for Reflection on the Teaching of Mathematics.

- Algorithmic thinking, implicit in the teaching of mathematics, could be developed and enlightened with the instruments of Algorithmic;
- Programming promotes formalized reasoning;
- Questions about effectiveness of algorithms involve mathematics;
- Data processing and digital computations are common in other disciplines;
- Computer Science transforms Mathematics, bringing new points of view on objects, bringing new questions, creating new fields in mathematics that are expanding rapidly, and changing the mathematician's activity with new tools.

Just after this report was published, algorithmic content was introduced in mathematics in grades 11 and 12, in literature series, and in optional mathematics courses in the last year of the economy and sciences series.

Later, between 2009 and 2012 in new official programs, algorithms were introduced as part of the mandatory mathematical content to be taught in all series of the general curriculum (literature, economy, sciences) from grades 10–12. Finally, in the 2010s, computer science reappeared as an autonomous discipline in upper secondary school, together with algorithms as part of the contents in mathematics. Since 2016, computer science is also taught in *cycle 4* (grades 7–9), but divided between two disciplines (mathematics and technology).

This renewal of the teaching of computer science in French curricula in mathematics raises the need for reworking and developing research in didactics of mathematics and informatics and of their interactions, which was the motivation for project DEMaIn. As a first step of the research, we led an epistemological study on these interactions in a didactic perspective, with a main focus on proof and proving. This is developed in Sect. 3.

3 Logical Issues in Mathematics and Computer Science

Following Durand-Guerrier and Arsac (2005), we consider that the classical first-order logic, namely the predicate calculus in the semantic perspective opened by Frege, Wittgenstein or Tarski, is a relevant epistemological reference for analysing proof and proving in mathematics education. Following authors such as Gribomont et al. (2000), we hypothesise that it is also the case for computer science. In this section,² we give a brief overview of this topic.

It should be noted that, even though more specialized logics and techniques exist in the research literature on programming language semantics and program verification, we do not focus here on the theories underlying automated or computer-assisted proof systems, even though they might be of interest as teaching tools. Since we are concerned here with the practice of proof in secondary or undergraduate education, we hypothesize that classical first-order logic is relevant for most of our goals.

²This was presented in an unpublished regular lecture given at ICME 11 (<http://www.icme11.org/>).

3.1 *Semantic Perspectives in Logico-Mathematical Disciplines*

In this text, semantics is considered in a logical perspective consistent with the definitions given by Morris (1938): *semantics* concerns “the relation of signs to the objects which they may or do denote” (op. cit. p. 21); *syntax* concerns the “relations of signs to one another in abstraction from the relations of signs to objects and interpreters” (op. cit. p. 13), and *pragmatics* refers to “the relation of signs to their users” (op. cit. p. 29). Morris claims that “Syntactics, Semantics and Pragmatics are components of the single science of semiotic but mutually irreducible components” (op. cit. p. 54). We illustrate the relevance of this approach below.

For example, when considering the addition of natural numbers, the semantic point of view refers to the definition of the sum as the cardinal of the union of two relevant discrete collections; the result is independent of the nature of the involved objects (provided that mixing these objects preserves their integrity). The syntactic point of view arises when addition is defined as the iteration of the successor operation; it does not require any reference to quantities; this provides algorithmic rules in a given system of numeration. Finally the pragmatic aspect concerns the articulation between syntax and semantics that is built by subjects in a back-and-forth between calculation (syntax) and effective counting (semantics). According to Da Costa (1997, p. 42), it is necessary to take in account all three of these aspects in order to gain a proper understanding of logico-mathematical fields.

Regarding computer science, one may consider that syntax is at the very core of the discipline, but there is evidence that semantic and pragmatic aspects are also involved [see for instance Gribomont et al. (2000)].

The semantic perspective in logic appears in Aristotle, and was developed in the late nineteenth and early twentieth centuries, mainly by Frege (1882), Wittgenstein (1921), Tarski (1933, 1943) and Quine (1950). In particular, Tarski (1933, 1943) provides a semantic definition of truth which he describes as *formally correct and materially adequate*, through the crucial notion of satisfaction of an open sentence by an object, and developed a model-theoretic point of view, of which semantics is at the very core.

3.1.1 **The Semantic Conception of Truth**

The main concern of Tarski is to give a definition of truth materially adequate and formally correct (Tarski, 1943). He claims his only intent in this work is to grasp the intuitions formulated by the so-called “classical” theory of truth, i.e. the conception that “truly” has the same meaning as “in agreement with reality” (contrary to a conception that “true” means “useful in such or such regard” (Tarski, 1933).

In order to be formally correct, such a definition ought to be recursive, but recursivity is usually difficult to grasp directly. Tarski’s idea was to introduce the notion of satisfaction of a propositional function (in modern terms, a predicate) of a given

formal language in a “domain of reality” (a piece of discourse, a mathematical theory etc.). In the field of algebra, this definition coincides exactly with that of solution of an equation. Tarski argues that this definition of satisfaction is the key for a recursive definition of the truth of a complex sentence.

First, there is an extension of logical connectors between propositions, as defined by Wittgenstein, to connectors between propositional functions (predicates). For example, given an interpretation, and P and Q two monadic predicates (with exactly one free variable), and a an element of the discourse universe, a satisfies $P(x) \Rightarrow Q(x)$ if and only if a satisfies $P(x)$ and $Q(x)$, or a does not satisfy $P(x)$.

Second, the two quantifiers “for all” and “there exists at least one” are defined in agreement with common sense. Then, once the logical structure of a sentence is identified (atomic formulae, scope of connectors and quantifiers), it is possible to establish the truth of the whole sentence as soon as one knows the truth-value of the interpretation of each atomic formula.

3.1.2 A Model-Theoretic Point of View

The model-theoretic point of view emerged in Tarski (1954, 1955), but the main ideas were already present in previous papers. It relies on a simple and very fruitful idea. At first, Tarski (1936, 1983) considers the notion of model of a formula. Given a formalized language L , a syntax providing recursively well-formed statements (formulae): $F, G, H\dots$, an interpretative structure (a domain of reality, a piece of discourse, a mathematical theory, a computation model) is a model of a formula F of L if and only if the interpretation of F in this structure is a true statement.

Some formulae are true for every interpretation of their letters in every non-empty domain. They are said to be universally valid (Quine, 1950). This is a generalisation of the notion of tautology in propositional calculus. A classical example is the logical equivalence $\forall x (P(x) \Rightarrow Q(x)) \Leftrightarrow \forall x (\neg Q(x) \Rightarrow \neg P(x))$ which describes the equivalence between a universal implication and its contrapositive and gives a logical basis to *proofs by contraposition*.

From the concept of model of a formula, Tarski defines the key concept of logical consequence in a semantic perspective: “The sentence X follows logically from the sentences of the class K if and only if every model of the class K is also a model of the sentence X ” (Tarski, 1983, p. 417). As was the case for propositional logic in Wittgenstein (1921), logical consequences support classical modes of reasoning. For example $Q(y)$ is a logical consequence of $P(y) \wedge \forall x (P(x) \Rightarrow Q(x))$. It corresponds to the extension of the propositional inference rule named *modus ponens* to predicate calculus.

3.1.3 The Methodology of Deductive Sciences

In his famous book *Introduction to logic* (Tarski, 1941; 1995), Tarski introduced in chapter VIII the methodology of deductive sciences. To a given miniature deductive

theory (he gave the example of the congruence of line segments), in which there are primitive terms, defined terms, axioms and theorems, one may associate an *axiom system*, with no reference to objects, which takes the form of a language and a set of formulae that can be reinterpreted in the given miniature theory. He then defines a *model* of the axiom system as any interpretation in which the formulae corresponding to the axioms of the given theory are interpreted as true. Of course the initial theory is a model of the obtained axiomatic formal system, but there may also be other models.

This leads to an important result and a powerful method for proving. Tarski proves, along with other logicians, the deduction theorem (in the meaning of Tarski), namely:

Every theorem of a given deductive theory is satisfied by any model of the axiom system of this theory; and moreover, to every theorem there corresponds a general statement which can be formulated and proved within the framework of logic and which establishes the fact that the theorem in question is satisfied by any such model. We have here a general law from the domain of methodology of deductive sciences, which, when formulated in a slightly more precise way, is known as the law of deduction (or the deduction theorem). (Tarski, 1995, p. 127)

As a consequence, “All theorems proved on the basis of a given axiom system remain valid for any interpretation of the system” (op. cit. p. 128).

This observation leads to the idea of *proof by interpretation*: one way to prove that a given statement is not a logical consequence of the axioms of a certain theory is to provide a model of the theory that is not a model of the formula associated with the statement in question. This can be seen as analogous to the use of a counterexample to the possibility of a proof or to the validity of a proof of a true statement.


Following Sinaceur (1991a), we consider that the model-theoretic point of view offers powerful tools enabling us to take into account both form and content and to distinguish between truth and validity, both crucial issues of the teaching and learning of mathematics. In a didactic perspective, Durand-Guerrier (2008) has shown that this point of view offers fruitful paths to enrich *a priori* analyses and to analyse students’ activity in mathematics. We will now attempt to provide evidence that this is also the case in computer science education.

3.1.4 Example: Tiling by Dominos

This example is based on our experience with *research situations for the classroom* (Gravier, 2008; Godot and Grenier, 2004). The problem is the following: given a rectangular grid (with integral dimensions), is it possible to tile it with dominos (1×2 rectangles)?

Theorem 1 *A rectangular grid can be tiled by dominos iff its area is even.*

A frequent (incorrect) proof of the above theorem given by students is the following. A grid can be tiled by dominos if and only if its area is $2k$ where k is the number of dominos, which means that the area of the grid is even.

The stated theorem is correct but the proof is not. It is sometimes difficult to invalidate an incorrect proof of a true statement. In order to do so, one can notice that the fact that the grid is rectangular was not used in the proof. So, this proof can be used for any shape consisting of an even number of squares. It is easy to see that the shape  can not be tiled by dominos but has an even area.

In other words, the set of grids of arbitrary shapes is a model of the theory used in the proof above. But in this model, the theorem becomes false. Hence, the initial proof is invalid (because otherwise it could be transported into the new model).

3.2 *Logic and Proof in Computer Science*

In Hopcroft et al. (2007, p. 5), the authors give the following remark:

In the USA of the 1990s [sic] it became popular to teach proof as a matter of personal feelings about the statement. While it is good to feel the truth of a statement you need to use, important techniques of proof are no longer mastered in high school. Yet proof is something that every computer scientist needs to understand.

In this section, we focus on the privileged role that logic and proof play in computer science, in particular to reason about programs and algorithms. We start by giving a few ideas on the interplay between syntax and semantics in the context of programming, then on the issues underlying the translation of an ideal algorithm into the rigid syntax of a programming language. We present a few classical types of proofs required in the study of algorithms, provide an example of such a proof using an ad-hoc deduction system, and close this section by a very brief presentation of the links between logic and well-known computation models called finite automata.

3.2.1 **Syntax and Semantics of Programming**

The distinction between syntax and semantics is somewhat more obvious in computer science (in particular in programming) than in mathematics, due to the way computers interpret programs. Indeed, for an algorithm to be executable by a machine, one first has to express it as a *text* amenable to automated treatment, from the lowest possible description level (elementary machine instructions) to the highest (modern programming languages). In this context, “syntax” refers to the rules of composition of a valid text in the chosen language, and “semantics” to the expected effect on the actual machine (or a model thereof) of each construct in that language and of their combinations.

Important pieces of software called *compilers* and *interpreters*, which rely on theoretical advances from the last decades of the 20th century, enable automatic translations of higher-level programs into machine-level lists of instructions (see for instance, Aho et al. 1986). These tools proceed in several phases, the first of which (lexical and syntactic analysis) aim at ensuring that the text of a program respects

the formal syntax of the chosen language. Further steps are mostly of a semantic nature: checking for type errors, modifying parts of the program, translating it into another, possibly lower-level language while preserving its meaning, or even running (*interpreting*) the program directly.

Contrary to low-level program descriptions such as assembly languages, modern languages are designed to be executable on several (possibly any) computer architectures. This “abstraction” from material constraints is an essential aspect of modern programming, in that it allows one to work at a level closer to general algorithmic ideas rather than being distracted by technical issues. Enforcing the semantic consistency of programs through each of these transformations regardless of the final target architecture is thus an essential responsibility of programming language designers and compiler implementers.

The study of programming language semantics is a wide and very active field of research, with numerous links to deep mathematical theories. It is essential for the design and understanding of whole paradigms of programming.

3.2.2 From Algorithms to Programs

Describing an algorithm as a machine-executable program is somehow similar to translating an informal mathematical statement in some formal (for instance logical or axiomatic) language, which can then be interpreted in the appropriate mathematical model. Indeed, algorithms are often described informally, using either natural language, mathematical notations, *pseudo-programs* expressed in some semi-formal language inspired by actual programming languages, or a mix of all three.

When one wishes to actually produce an executable program realizing the tasks described by such an informal algorithm (its *implementation*), one therefore has to remove any possible ambiguity, and ensure that this translation process faithfully renders the ideas and principles which allow the algorithm to solve the problem at hand. In some sense, like a mathematician’s, a programmer’s activity therefore has to do with *pragmatics*: it proceeds as a constant back-and-forth between syntax and semantics, with the additional parameter of technical constraints. This pragmatic work is very similar to the work of producing the formal proof of a mathematical result using a proof assistant. In some sense, formalizing an algorithm into a program is an activity of the same nature as producing a formal proof from a standard mathematical proof.

However, as is the case in mathematics, ensuring that some formal statement is syntactically correct is not enough to guarantee that it is “true”, or in this case that it actually performs the task it was meant to perform. Therefore, in order to ascertain the actual *correctness* of a program (or even of the algorithm it is supposed to implement), one usually has to resort to external arguments which are of a logical or mathematical nature.

3.2.3 Reasoning About Programs or Algorithms

One of the most obvious questions one may ask about an algorithm or a piece of program is “Does it work?”. Trying to state this question more precisely leads to a formal definition of *computation problems*, which one may summarize as: “mathematical relations between a set of *instances* and a set of *results* (or *answers*, or *solutions*)”. One further distinguishes *decision* problems, where possible answers are simply truth values. In this case, a problem might equivalently be described as its *set* of positive instances, instances which are mapped to the value *true*. Such problems play an important role in the more theoretical aspects of computer science, in particular in formal languages, automata and computation theories [see for instance Hopcroft (2007)].

Correctness Let P denote an algorithmic problem. Seeing P as a map between instances and outcomes, let us write $P(x)$ the outcome associated with some admissible instance x . To say that an algorithm or program A solves problem P means that given any admissible instance x of P , A is able to provide (indeed *compute*), after a finite sequence of *elementary operations*, a description of $P(x)$. In view of this, the question of knowing whether some algorithm A “works” (i.e. “proving” A) comes down to establishing the following two properties, whose conjunction might be seen as expressing the (full) correctness of A :

Termination: on any instance of P , A performs at most a finite number of computation steps.

Partial correctness: on any instance x of P , the value computed by A is $P(x)$.

Complexity The above questions are sometimes complemented by questions regarding A 's efficiency, in terms of the number of computation steps it performs on instances of a certain size (assuming some appropriate notion of *size* on instances). One typical property of interest is:

Worst-case upper bound: Function f is a worst-case upper bound for the complexity of A if there exists a positive constant c such that, on any instance of size n of P , the number of computation steps performed by A is at most $c \cdot f(n)$ for n large enough.

Similar questions can be asked of the amount of memory required by an algorithm (*space complexity*). Such concerns form the well-established fields of *complexity theory* and *algorithm analysis*, which strongly rely on tools and techniques from algebra and combinatorics. In the above statement, one is concerned with worst-case guarantees on the number of performed computation steps. Other natural questions concern the behaviour of A on *typical* cases, bringing into play the question of probabilistic distributions on instances, and possibly involving powerful techniques from probability theory and analysis (see for instance Arora and Barak, Arora and Barak (2009) for an introduction to the field, or Sedgewick and Flajolet (2013) for more in-depth material on average-case analysis).

Lower bounds and optimal algorithms Finally, interesting questions lie *beyond* the analysis of a single algorithm solving a problem P , and study the intrinsic complexity of P itself. For instance, in the so-called *comparison tree model*, in which all executions of an algorithm are decided through a series of elementary, binary comparisons between numbers, it can be shown that the well-known problem of sorting a list of numbers *cannot* be solved using less than $n \log n$ comparisons, up to a constant factor [for more details, see (Cormen et al. 2009)].

This impossibility result comes at the price of a rather involved argument, with several “layers” of quantification: one has to consider the longest computation, on any instance of some size n , of the (hypothetical) most efficient algorithm solving P . This supports the claim that proficiency with logic and reasoning are a prerequisite for a reasonably complete understanding of algorithmic concepts.

Modeste (2012, 2013) showed that this theoretical view of algorithms, which leads to adopting a definition of *problem* as a set of instances and a question about any of the instances, can be used as a relevant didactic tool, in particular to help develop an epistemological model for didactical purposes, to analyse curricula and to design didactical situations (see Sect. 4). Meyer and Modeste (2018) give an example of the didactical analysis of an algorithmic question (about the binary search algorithm and the bisection method).

3.2.4 An Example: Partial Correctness Using a Deduction System

In Gribomont et al. (2000), several key examples of particularly fruitful uses of logic in a computer science setting are given. The first such example is that of Hoare logic (Hoare, 1969), which may be used to show the partial correctness of a sequential program.

The main building block of Hoare logic takes the form of triples $\{P\} C \{Q\}$, called *Hoare triples*, where P and Q are assertions (usually written in classical predicate logic) and C is a program statement. Such a triple expresses the fact that, whenever P holds in some state of the machine (or model thereof) over which statement C is executed, it must be the case that Q holds in the state which is reached after C is performed. Hoare triples are manipulated using deduction rules, which are very reminiscent of classical proof systems. One of the simplest rules describes the semantics of sequential composition:

$$\frac{\{P\}C\{Q\} \quad \{Q\}D\{R\}}{\{P\}C; D\{R\}}$$

This rule expresses the fact that if $\{P\}C\{Q\}$ and $\{Q\}D\{R\}$ are both valid Hoare triples, then by performing statements C and D from a state verifying assertion P , one may guarantee that assertion R holds. Combining several rules of this kind and additional mathematical knowledge about manipulated values (for instance arithmetic) allows one to formally prove that, if and when a program terminates, some assertion holds at the end of its execution. See Gribomont et al. (2000) for a more detailed description and example, or Reynolds (1998) for a textbook covering this topic among others.

The issue of termination is of a different nature and cannot be established using this technique. It has to be proven separately, often relying on some kind of infinite-descent argument. Other examples of how logico-mathematical formalisms may be used in order to reason about other kinds of programs are given in Gribomont et al. (2000). One may cite in particular the cases of functional programs (where recursion and more particularly structural induction play a central role), concurrent or parallel programs, etc.

A word on structural induction To conclude this section, let us remark that the correctness of the final assertions obtained by applying the above technique actually relies on a *structural induction* argument: indeed, the syntactic structure of a program's text can be described by its so-called *abstract syntax tree*, whose nodes are program constructs and whose leaves are essentially identifiers and values. Successively applying deduction rules such as the one described above for sequential composition actually comes down to inductively labelling each node of this tree, from the leaves to the root, with sets of assertions. Finally, assertions carried by the root of the tree represent true facts about the whole program.

Other (simpler) examples of the usefulness of structural induction are provided in Gribomont et al. (2000), in the context of correctness proofs for functional programs written in the language LISP or SCHEME.

3.2.5 Modelling Program Behaviour Through Logic and Automata

Another bridge between logic and computer science illustrated by Gribomont et al. (2000) concerns the study of a class of computation models called *automata*, which stem from a long line of research originating in the 1960s and have known many interesting developments. These results are collectively referred to as *automata theory* (see for instance Hopcroft et al. (2007) for a classical textbook, Straubing and Weil (2012) or Thomas (1997) for a more logic-oriented exposition).

Finite-state automata A finite-state automaton is characterized by a finite directed edge-labelled graph, whose vertices and edges are respectively called *states* and *transitions*. Some states are marked as *initial*, others as *terminal* or *accepting*. The labels of edges are called *letters*, they belong to a finite set called *alphabet*. A sequence

of letters, or *word*, is said to be *accepted* by an automaton if its letters label the successive transitions along a path from some initial state to some final state (or in the case of infinite words, to some kind of accepting “repetition”). Each automaton therefore accepts a *language*, which is the set of all words it accepts.

Finite automata have very good and well-understood algorithmic properties. For instance, one can write algorithms to decide whether a given automaton accepts the empty language or the set of all possible words, build an automaton whose language is the union or intersection of the languages of two other automata or the complement of the language of a given automaton. A particularly strong connection between automata and logic, discovered in the 1960s and much developed since, is that the languages of some classes of automata coincide with the languages defined by some classes of logics (see, Thomas, 1997). Furthermore, in several cases there exist algorithms able to transform a logical formula into an equivalent automaton and vice-versa.

Modelling and verifying programs A typical application of automata theory has to do with automated program verification or *model-checking*. In this framework, a program is modeled as a (potentially very large) finite-state automaton, say A , in such a way that each execution of the real program corresponds to a (possibly infinite) path in A , but A may exhibit additional behaviours. A property φ to be verified on the program might then be expressed in a well chosen logical framework. This formula is then negated, and translated into another finite automaton $A_{\neg\varphi}$.

Determining whether the abstract program satisfies the property stated by formula φ then amounts to checking whether the languages of automata A and $A_{\neg\varphi}$ are disjoint, which can be done algorithmically. By construction of A , an erroneous answer may occur only in the case where some behaviour of A which violates φ is detected, but this behaviour does not exist in the original program (this is called a *false positive*). Otherwise, if no such execution is found, it is guaranteed that all executions of the actual program respect the property φ .

This verification procedure has shown great success despite the fact that it deals with finite-state computation models. Extending it to more realistic models while conserving good algorithmic properties is one of the challenges undertaken by the research field of program verification [see for instance Bérard (2001)].

3.3 First Conclusion

Adopting a semantic point of view, and being situated at a meta-mathematical level, a model-theoretic point of view provides on the one hand a frame to analyse *a priori* the situations under both mathematical and didactical aspects, and on the other hand to analyse students' activity, in particular by providing the researcher with a methodology to identify and study the elaboration, the evolution and the eventual overtaking of the local axiomatic all along the resolution and/or the proving process.

It seems clear that Tarski's meta-mathematical project goes beyond mathematics and echoes key questions in computer science education, such as the relationship between deductive systems and models, including the issue of limits of validity of these models, the relationships between proofs and programs, the notion of proof of an algorithm.

It should also be noted that the pervasive use of logic and other mathematical tools in computer science has provided and will most likely continue to provide new ideas, questions and perspectives to the fields of logic and mathematics themselves.

4 Modes of Interaction Between Mathematics and Computer Science

The content of this section mainly originates from Modeste (2012). In a didactical perspective and on the basis of an epistemological analysis, it proposes to distinguish three main modes of interaction between mathematical proof and algorithms, and two kinds of problems in which algorithms appear. This allows one to better understand and analyse the interactions between both fields, and give meaning to several possible relationships with the concept of *algorithm*, that is, *conceptions* of algorithm.

This work draws upon the model of conceptions as developed by Vergnaud and enriched by Balacheff. The $cK\phi$ model (conception, knowing, concept) was developed by Balacheff (2013) to build a bridge between mathematics education and research in educational technology. It proposes a model of learners' conceptions inspired by the theory of didactical situations (Brousseau, 1997) and the theory of conceptual fields (Vergnaud, 2009).

In this model, conceptions are defined as quadruples (P, R, L, Σ) in which P is a set of problems, R a set of operators, L a representation system and Σ a control structure. P and L directly correspond with situations and representations in Vergnaud's model, R and Σ distinguish between operational invariants, those which allow one to act on problems and those which allow one to control the actions. For this reason, the $cK\phi$ model is very relevant for emphasizing the dimension of proof.

4.1 Proof Paradigms

We first distinguish three frameworks in which our conceptions will be described, which we call *paradigms*. They are the *algorithmic proof* (AP), the *mathematical algorithm* (MA) and the *computer algorithm* (CA). These paradigms essentially correspond to three possible *habitats*³ of the notion of algorithm in mathematical and algorithmic activity.

³The term *habitat* was coined by Artaud (1998) in the context of the so-called *ecological* approach to didactics.

- The **AP paradigm** corresponds to an activity of the form Problem–Theorem–Proof, where the proof is of a finite, constructive nature and can thus be seen as algorithmic in nature. Induction proofs in particular fall into this category. In this paradigm, algorithms and proofs are not dissociable: algorithms are not written directly but are implicit in the theorem’s proof. They may be made explicit outside or after the proof, as a corollary or consequence thereof, in a fashion similar to the second paradigm (MA).
- The **MA paradigm** corresponds to an activity of the form Problem–Algorithm–Proof. For a given problem, one describes an algorithm solving all admissible instances, then provides a proof that this algorithm is indeed (totally) correct, in other words a termination proof (the algorithm provides an answer in a finite amount of steps on any instance) together with a partial correctness proof (whenever an answer is provided on an instance, it is the correct one). In this paradigm, algorithm and proof are clearly dissociated.
- The **CA paradigm** corresponds to an activity of the form Problem–Program–Validation. The algorithm solving the problem is expressed as a program which is expected to be executed on a machine. The term “validation” should be understood here in its ordinary meaning. It may or may not be of a mathematical nature, and may include all relevant tools and practices such as syntactic analysis and type-checking performed by the compiler, manual, semi-automatic, or automatic testing procedures, verification techniques (for instance model-checking tools as described in Sect. 3.2.5), etc. Note that mathematical validation of the program or the underlying algorithm via proof (as in paradigm MA) may also provide a form of validation in CA.

A single problem can be addressed in different paradigms. Its study may even draw on several of them simultaneously or successively. What distinguishes the three paradigms is therefore not the kind of problems which are addressed but rather the way in which they are treated, the kind of solution obtained, and the kind of validation which is provided. They also differ in the concrete form in which algorithms are expressed. This encourages us in our choice to make use of the model of *conceptions* in order to formalize these differences in terms of operators, representation systems and control structures.

Representation systems for algorithms are an important criterion for distinguishing the three paradigms, but are not the only one. We must however acknowledge their impact on the way algorithms are expressed.

It remains to ask which kind of problems are the most appropriate for giving meaning to the concept of algorithm in the classroom.

4.2 *The Tool–Object Dialectic*

We propose to adapt here a definition which comes from the theory of algorithmic complexity (see Sect. 3.2.3). According to this definition, a problem (e.g. finding the gcd) is given by a pair (I, Q) , with:

- I a set of instances (e.g. \mathbb{N}^2 , the set of all pairs of natural numbers) ;
- Q a question about these instances (e.g. what is the gcd of the 2 provided numbers?).

This definition of problems allows to formalize what an algorithm is. An algorithm is a systematic method which must give an answer for all instances of the problem, after a finite number of steps (e.g. Euclid's algorithm solves the problem of gcd for any pair of natural numbers).

Additionally, we say a problem is *instantiated* when one chooses a particular instance i and tries to answer the question $Q(i)$ for this particular case (e.g. what is the gcd of 3654 and 76?). To grasp the concept of algorithm in its full generality, it is important not to address only instantiated questions but to study a problem in all of its instances.

It is also important to distinguish two kinds of problems giving sense to the concept of algorithm:

- The set \mathcal{P}_a of problems that may be solved using an algorithm (e.g. the problem of finding the gcd);
- The set \mathcal{P}_A of problems that concern algorithms (which includes the problem of determining if a given problem is in \mathcal{P}_a , the problem of determining the complexity of an algorithm, etc.).

In the first case, the algorithm is seen as a tool, and in a teaching context it is important that at least some of the chosen exercises and problems are general, and not instantiated (in some high school textbooks, several exercises in the same chapter turn out to be instantiated versions of the same problem). In the second case, the algorithm is seen as an object, and a problem can be instantiated (on a specific algorithm for instance).

We argue that the tool–object dialectic (Douady, 1986) can be useful to think about the interaction between mathematics and computer science, in particular to deal with proof issues.

Computer Science can be seen as a tool for mathematics (simulating experiences, or testing small cases) or an object (probabilities for analysing the complexity of an algorithm). Conversely, mathematics can be seen as an object or a tool for computer science, according to whether one is studying the mathematical or computer science aspects of a situation (as presented above).

4.3 Six Conceptions to Analyse Algorithmic Activity

We now revisit the three paradigms defined above in the light of the tool–object dialectic. We saw how the distinction between problems in \mathcal{P}_a , where the algorithm is a tool, and \mathcal{P}_A , where it is the object of study, on one hand, and between instantiated and non-instantiated problems on the other hand, provide insights on the tool–object dialectic.

Table 1 Conceptions in paradigm AP

<i>(a) The AP-tool conception</i>	
<i>P:</i>	Problems in \mathcal{P}_a
<i>R:</i>	Operators are those of proof restricted to “constructive” modes of reasoning (recurrence, induction, infinite descent, existence of lower or upper bounds of finite sets...), excluding in particular proofs by contradiction or using the law of excluded middle
<i>L:</i>	The representation system is that of <i>mathematical language</i> . The involved objects are mathematical object. In this conception, one manipulates information. At all times, all information provided by the instance, and all deduced information is usable. The only variables which are used are <i>mathematical variables</i>
Σ :	The control structure is that of <i>mathematical logic</i> , together with known properties of occurring objects
<i>(b) The AP-object conception</i>	
<i>P:</i>	Problems in \mathcal{P}_A
<i>R:</i>	Operators are those of mathematical proof and operations on theoretical computation models (algorithmic reductions, simulations, etc.)
<i>L:</i>	The representation system is that of mathematical language together with theoretical computation models (automata, Turing machines, recursive functions, decision trees...)
Σ :	The control structure is that of <i>mathematical logic</i> , together with properties of occurring objects

We therefore further refine the three paradigms presented above, by distinguishing in each case a tool-conception and an object-conception. This yields a total of six conceptions which we will not describe in detail.

Tables 1, 2 and 3 describe the different components (*P*: problems, *R*: operators, *L*: representation structures and Σ : control structures) of each of these conceptions. Each emphasized term in the description of a conception is explained below.

- **The AP-tool conception** (Table 1a) concerns inherent, implicit algorithms in the description of certain constructive mathematical proofs, whose object is *not* an algorithm or algorithmic fact itself. One may relate this to the notion of *constructive* mathematical proof.

In the table, by *mathematical language* we mean the language commonly used in mathematical writings. By *mathematical variables* we refer to the different kinds of variables used in mathematics. By *mathematical logic*, we mean the set of implicit reasoning rules used in mathematical activity (in contrast with formal logic).

Example: A proof of the characterization of Eulerian graphs (graphs which possess a cycle traversing each edge exactly once) as graphs whose vertices all have even degree, written in usual mathematical terms, where each step is constructive and the structure of reasoning attests to this constructiveness, might fall into this conception.

Table 2 Conceptions in paradigm MA

<i>The MA-tool conception</i>	
<i>P:</i>	Problems in \mathcal{P}_a
<i>R:</i>	Operators are explicit algorithmic constructs (conditions, iterations, recursion) and effective, constructive operations on numbers, sets or other combinatorial or mathematical objects
<i>L:</i>	The representation system might be some type of pseudo programming language, mixing mathematical language and vocabulary inspired by programming practices. Manipulated objects are mathematical objects, sometimes belonging to the culture of computer science (for instance <i>abstract data types</i>), each admitting a known set of effective operations. Variables can be mathematical variables or <i>computer variables</i>
Σ :	The control structure is that of algorithm proof (correctness and termination) using all appropriate concepts and formalisms (logical proof systems, inductive proofs, infinite descent, invariants...).
<i>The MA-object conception</i>	
<i>P:</i>	Problems in \mathcal{P}_A
<i>R:</i>	Operators are those of mathematical proof: algorithm proof, properties of algorithms, invariants, computational complexity...
<i>L:</i>	The representation system is that of mathematical language
Σ :	The control structure is that of mathematical logic, reasoning rules and properties of occurring objects

Table 3 Conceptions in paradigm CA

<i>The CA-tool conception</i>	
<i>P:</i>	Problems in \mathcal{P}_a
<i>R:</i>	Operators are those provided by a given programming language, possibly including instructions, conditional structures, loops, functions, and various predefined operations on data
<i>L:</i>	The representation system is a programming language. Manipulated objects are data values which encode (or model) objects from the original problem using data structures which allow certain operations
Σ :	The control structures are provided by various computer programming tools and practices, either manual or automatic, formal or informal
<i>The CA-object conception</i>	
<i>P:</i>	Problems in \mathcal{P}_A
<i>R:</i>	Operators are those of formal program verification
<i>L:</i>	The representation system is that of mathematical language, together with the vocabulary and notations of appropriate analysis techniques and tools, possibly including <i>ad-hoc</i> proof systems or formal logic frameworks
Σ :	The control structure is provided by various relevant theories, including programming language theory, language semantics, computation models, and formal logic

- **The AP-object conception** (Table 1b) deals with mathematical proofs about algorithmic objects or facts. Here, an algorithm or algorithmic problem may be provided as part of the question.

Examples: A proof on the intrinsic complexity of some algorithmic problem might fall in this category: for instance proofs of the fact that *any* comparison-based sorting algorithm must perform at least $O(n \log n)$ comparisons in the worst case, or that the knapsack problem is NP-complete.

- **The MA-tool conception** (Table 2a) has to do with proofs explicitly providing an algorithm solving the mathematical problem (whose object is not itself explicitly algorithmic), and possibly providing some justification that the proposed algorithm is correct.

In the table, the notion of *abstract data type* refers to the description of a specific data domain (for instance finite lists and maps, stacks or queues, graphs) through a set of allowed operations, without any *a priori* knowledge on implementation details. These operations are assumed to be constructive, or effective (in the sense that they can be performed algorithmically). Describing such algorithmic data types is an important part of the field of algorithm design.

We call *computer variables*, for lack of a better term, entities that play the role of a temporary assignment between a name and a value, which is subject to change over time (for instance across multiple iterations of a loop). When a new assignment to a certain name is performed, the value previously assigned to it, if any, is lost. This is a simplified model of the memory of an actual computer during the execution of a program, designed to hide irrelevant technological details.

Examples: A proof of the existence of the greatest common divisor of two integers, presented first by writing down Euclid's algorithm, then by proving its correctness, falls in this category. Other examples might include the description of list-sorting or list-searching algorithms.

- **The MA-object conception** (Table 2b) is similar to the AP-object conception in that it concerns problems about algorithms and their proofs. One possible difference in this case is that actual algorithms are manipulated explicitly, instead of potential or hypothetical algorithms.

In this conception the description of the algorithm itself is considered part of the problem at hand; algorithmic description operators and representation systems are therefore not included in the table.

Example: The analysis of the computational complexity (in terms of time or space) of a particular algorithm might fall into this category. The study of other properties may appear as well, for instance the stability of a given sorting algorithm.

- **The CA-tool conception** (Table 3a) refers to the activity of directly providing a computer program expected to solve a given problem, either numerically or symbolically, perhaps even in an approximate manner.

Particular care might be taken in the validation and control of the proposed solution, which is considered good programming practice. The least required effort usually consists only in clear code documentation and sufficient testing, but other compelling arguments may be provided by other tools and procedures, such as syntactic analysis, type checking, certifiable code annotation, program verification, automated testing, etc.

Example: Computing the gcd of two integers using an implementation of Euclid’s algorithm written in C or some other programming language falls into this conception.

- **The CA-object conception** (Table 3b) concerns questions asked about the properties of a given, explicit computer program.

Examples: Writing manual or computer-assisted proofs about the termination or safety of programs relates to this conception.

Another interesting and rather extreme example is the development of automated code analysis, interpretation or transformation tools themselves, for instance compilers, interpreters, profilers or debuggers for a given programming language, or automatic or semi-automatic verification software. In this case, the adopted representation system must be able to handle code reification: namely, the representation and manipulation of programs themselves.

4.4 *Relationships Between Conceptions*

4.4.1 **The Tool–Object Continuum**

Let us first review the relationship between the tool-conception and object-conception of the same paradigm. We wish to make it clear that the boundary between these conceptions is not as clear-cut as our taxonomy might seem to indicate. Indeed, there is no wide gap between the two conceptions associated with a given paradigm, but rather a continuity according to the tool–object dialectic, or rather a transition from tool to object. This transition is accompanied by a move from specific, instantiated problems towards generic ones. However, there may in practice exist several intermediate problems which occur as one progressively extends and widens the set of instances of the problem at hand. One should also note that the control structure of the tool-conception plays a particular role in this transition from tool to object: the more it is present in a given activity, the closer one gets to the corresponding object-conception. The question of determining whether a given problem admits an algorithmic solution also plays a central role in this articulation.

Moreover, it should be remarked that when we mention the existence of this shift from tool to object, we do not mean to imply that algorithmic activity is necessarily linear or one-way. In fact, there are clearly numerous alternations, just like in mathematics, between tool and object. But it appears to us that this shift is globally directed from tool to object, in the sense that the use of any tool may naturally bring questions that make it a potential object of study. One can also see this as a movement which, in terms of the conceptions model, tends to change the status of control structures into operators in the context of new problems.

4.4.2 A Continuum Between Mathematics and Computer Science

Similarly, we have to point out that the distinction between paradigms AP, MA and CA is not absolute, but rather witnesses a fine gradation of conceptions and concerns between mathematics and computer science. This arbitrary split into three paradigms seems to offer a reasonable granularity for use as an analysis tool, to allow a sufficiently accurate representation of the various conceptions found in scientific literature and to support reflections about the production of this knowledge. Therefore, in addition to the tool–object continuity, there exists another continuity along the AP–CA axis, and it might be the case that certain activities found either in literature or in teaching may fall between two of our conceptions.

Still, let us remark that one may find some kind of “chronological” hierarchy (which one should absolutely not see in terms of value) between AP, MA and CA. For a given problem in \mathcal{P}_a , scientific study rather follows a chronological shift from AP to CA, via MA. One may describe this as a transition from constructivity concerns (AP) to effectivity concerns (MA) and finally to implementation concerns (CA), accompanied by an increasing level of detail in the specification of algorithms.

This process obviously has exceptions, for instance in the case where no exact proof of a certain phenomenon exists but empirical observation through programming may still offer insights as to its validity. One should also be aware that many (indeed most) questions about algorithms and programs are of course *undecidable* in general. It is therefore useless to expect a single computer program to check *exactly*, given the text of any other program, whether its executions always terminate. Nevertheless, in some cases automated tools may be able to provide exact information about elementary properties of programs, or approximate answers to more difficult questions.

The continuity along the AP–CA axis is also accompanied by an increasingly clear separation of the aspects related to proof, syntax and semantics. Indeed, in the AP paradigm, all three concerns are intermixed. In MA, the solution is given as a “construction” or algorithm. It carries in some sense both syntax and semantics and is separated from its validation. Finally in CA, a strong accent is put on syntax (indeed, a computer executing a program only performs purely symbolic manipulations), while concerns of semantics and proof are left to the designer of the program (possibly with the help of computer-assisted tools). This interplay between validation, meaning and representation of algorithms, which varies between paradigms, is, for us, a central point to understand the role and place of proof in mathematics and computer science. We make the hypothesis that it has a strong potential for studying didactical issues about proof in mathematics, computer science and their interactions.

5 Perspectives

In this chapter, we have tried to provide evidence of the necessity and the relevance of studying the interactions between mathematics and computer science, with a particular focus on proof and taking into account the interplay between syntax, semantics

and pragmatics. This opens up new avenues of research by helping identify issues in logic, mathematics and computer science that may be overlooked or remain implicit in the classroom. An example we are currently exploring is the representation and manipulation of polynomials. This topic illustrates several of the issues we discussed in this chapter:

- representation issues—polynomials represented as lists of coefficients or as arbitrary expressions via their syntax trees;
- algorithmic issues—computations in both representations (naive evaluation or using Horner’s scheme, arithmetic operations on polynomials, formal derivation or integration, canonical forms, equivalence...);
- complexity analysis—comparison between representations;
- interplay between syntax and semantics—work on the structure of algebraic expressions;
- possible context for the introduction of inductive constructions and reasoning;
- classical classroom mathematical content—operator priorities and other algebraic rules, decomposition of algebraic formulae into calculation programs.

Due to the role of polynomials in calculus and analysis, we consider that developing didactical situations aiming to deal with these aspects will improve the knowledge of polynomials as objects, and as a consequence will foster students’ skills in recognizing and using them as tools in relevant contexts.

Acknowledgements Research funded by the french *Agence Nationale pour la Recherche*, project number <ANR-16-CE38-0006-01>.

References

- Aho, A. V., Sethi, R., & Ullman, J. D. (1986). *Compilers, principles, techniques*. Addison Wesley.
- Arora, S., & Barak, B. (2009). *Computational complexity: A modern approach*. Cambridge University Press.
- Artaud, M. (1998). Introduction à l’approche écologique du didactique - L’écologie des organisations mathématiques et didactiques. *Actes de la IXème école d’été de didactique des mathématiques* (pp. 101–139). Caen: ARDM & IUFM.
- Arzarello, F., Bussi, M. G. B., Leung, A. Y. L., Mariotti, M. A., & Stevenson, I. (2012). Experimental approaches to theoretical thinking: Artefacts and proofs. In *Proof and proving in mathematics education* (pp. 97–143). Springer.
- Balacheff, N. (2013). $cK\Phi$, a model to reason on learners’ conceptions. In *PME-NA 2013-psychology of mathematics education* (pp. 2–15). North American Chapter.
- Baron, G. L., & Bruillard, É. (2011). L’informatique et son enseignement dans l’enseignement scolaire général français: enjeux de pouvoir et de savoirs. In: *Recherches et expertises pour l’enseignement scientifique* (Vol. 1, pp. 79–90). De Boeck Supérieur.
- Basu, S., Pollack, R., & Roy, M. F. (2006). Algorithms in real algebraic geometry (2nd ed.). In *Algorithms and computation in mathematics* (Vol. 10). Berlin, New York: Springer.
- Bérard, B. (2001). *Systems and software verification: Model checking techniques and tools*. Springer.
- Borwein, J. M. (2012). Exploratory experimentation: Digitally-assisted discovery and proof. In *Proof and proving in mathematics education* (pp. 69–96). Springer.

- Brousseau, G. (1997). *Theory of didactical situations in mathematics*. Kluwer Academic Publishers.
- Chabert, J. L. (1999). *A history of algorithms from the pebble to the microchip*. Springer.
- Colton, S. (2007). Computational discovery in pure mathematics. In *Computational discovery of scientific knowledge* (pp. 175–201). Springer.
- Cormen, T. H., Leiserson, C. E., Rivest, R. L., & Stein, C. (2009). *Introduction to algorithms* (3rd ed.). The MIT Press.
- Da Costa, N. C. A. (1997). *Logiques classiques et non classiques: essai sur les fondements de la logique*. Paris: Masson.
- Douady, R. (1986). Jeux de cadres et dialectique outil-objet. *Recherches en didactique des mathématiques*, 7(2), 5–31.
- Durand-Guerrier, V. (2008). Truth versus validity in mathematical proof. *ZDM*, 40(3), 373–384.
- Durand-Guerrier, V., & Arzac, G. (2005). An epistemological and didactic study of a specific calculus reasoning rule. *Educational Studies in Mathematics*, 60(2), 149–172.
- Frege, G. (1882). Über die wissenschaftliche Berechtigung einer Begriffsschrift. *Zeitschrift für Philosophie und philosophische Kritik*, 81, 48–56. (English translation in *Conceptual notation and related articles*. Clarendon Press (1972)).
- Godot, K., & Grenier, D. (2004). Research situations for teaching: A modelization proposal and examples. In *Proceedings of ICME 10*, IMFUFA, Roskilde University.
- Gravier, S., Payan, C., & Colliard, M. N. (2008). Maths à modeler: Pavages par des dominos. *Grand N*, 82, 53–68.
- Grenier, D., & Payan, C. (1998). Spécificité de la preuve et de la modélisation en mathématiques discrètes. *Recherches en didactique des mathématiques*, 18(2), 59–100.
- Gribomont, P., Ribbens, D., & Wolper, P. (2000). Logique, automates, informatique. In F. Beets & E. Gillet (Eds.), *Logique en perspective: Mélanges offerts à Paul Gochet*, Ousia (pp. 545–577).
- Guedet, G., Bueno-Ravel, L., Modeste, S., & Trouche, L. (2017). Curriculum in France. A national frame in transition. In *International perspectives on mathematics curriculum, research issues in mathematics education series*. IAP.
- Hart, E. W. (1998). Algorithmic problem solving in discrete mathematics. In L. J. Morrow & M. J. Kenney (Eds.), *The teaching and learning of algorithm in school mathematics, 1998 NCTM Yearbook* (pp. 251–267). Reston, VA: National Council of Teachers of Mathematics.
- Hoare, C. A. R. (1969). An axiomatic basis for computer programming. *Communications of the ACM*, 12(10), 576–580.
- Hopcroft, J., Motwani, R., & Ullman, J. (2007). *Introduction to automata theory, languages, and computation* (3rd ed.). Addison-Wesley.
- Howson, A. G., & Kahane, J. P. (Eds.). (1986). *The influence of computers and informatics on mathematics and its teaching, international commission on mathematical instruction edn*. ICMI Study Series. Cambridge University Press.
- Kahane, J. P. (2002). *Enseignement des sciences mathématiques : Commission de réflexion sur l'enseignement des mathématiques : Rapport au ministre de l'éducation nationale* (cndp ed.). Paris: Odile Jacob.
- Lovász L. (2007) Trends in mathematics: How they could change education? In *Conférence européenne "The future of mathematics education in Europe"*, Lisbonne.
- Meyer, A., & Modeste, S. (2018). Recherche binaire et méthode de dichotomie, comparaison et enjeux didactiques à l'interface mathématiques - informatique. In *Proceedings of EMF*, Paris, France (to appear).
- Modeste, S. (2012). Enseigner l'algorithme pour quoi ? Quelles nouvelles questions pour les mathématiques ? Quels apports pour l'apprentissage de la preuve ? Ph.D. thesis, Université de Grenoble.
- Modeste, S. (2013). Modelling algorithmic thinking: The fundamental notion of problem. In *Proceedings of CERME 8*, Antalya (Turkey).
- Modeste, S. (2016). Impact of informatics on mathematics and its teaching. In F. Gadducci & M. Tavosanis (Eds.), *History and philosophy of computing* (pp. 243–255). Cham: Springer International Publishing.

- Morris, C. W. (1938). Foundations of the theory of signs. In *International encyclopedia of unified science* (pp. 1–59), Chicago University Press.
- Ouvrier-Buffet, C. (2014). Discrete mathematics teaching and learning. In *Encyclopedia of mathematics education* (pp. 181–186).
- Perrin, D. (2007). L'expérimentation en mathématiques. *Petit x*, 73, 6–34.
- Quine, W. V. (1950). *Methods of logic*. Harvard University Press.
- Reynolds, J. C. (1998). *Theories of programming languages*. Cambridge University Press.
- Sedgewick, R., & Flajolet, P. (2013). *An introduction to the analysis of algorithms*. Pearson Education.
- Sinaceur, H. (1991a). Corps Et Modèles: Essai Sur l'Histoire de l'Algèbre Réelle. Vrin.
- Sinaceur, H. (1991b). Logique: mathématique ordinaire ou épistémologie effective? In: Hommage à Jean-Toussaint Desanti, Trans-Europ-Repress.
- Straubing, H., & Weil, P. (2012). An introduction to finite automata and their connection to logic. In P. S. Deepak D'Souza (Ed.), *Modern applications of automata theory* (pp. 3–43). IISc Research Monographs: World Scientific.
- Tarski, A. (1933). The concept of truth in the languages of the deductive sciences. *Prace Towarzystwa Naukowego Warszawskiego, Wydział III Nauk Matematyczno-Fizycznych* 34(13), 172–198 (English translation in Tarski (1983)).
- Tarski, A. (1936). On the concept of logical consequence. *Przegląd Filozoficzny*, 39, 58–68 (English translation in Tarski (1983), pp. 409–420)
- Tarski, A. (1941). *Introduction to logic and to the methodology of the deductive sciences*. Oxford University Press (reedited in Tarski (1995))
- Tarski, A. (1943). The semantic conception of truth and the foundations of semantics. *Philosophy and Phenomenological Research*, 4(3), 341–376.
- Tarski, A. (1954). Contributions to the theory of models, I–II. *Indagationes Mathematicae*, 16, 572–588.
- Tarski, A. (1955). Contributions to the theory of models, III. *Indagationes Mathematicae*, 17, 56–64.
- Tarski, A. (1983). *Logic, semantics, metamathematics: Papers from 1923 to 1938*. Hackett (J. H. Woodger, trans. Introduction: J. Corcoran).
- Tarski, A. (1995). *Introduction to logic and to the methodology of deductive sciences*. New York: Dover Publications, INC. (unabridged Dover republication of the edition published by Oxford University Press, New York, 1946)
- Thomas, W. (1997). Languages, automata, and logic. In *Handbook of formal languages* (pp. 389–455). Springer.
- Vergnaud, G. (2009). The theory of conceptual fields. *Human Development*, 52(2), 83–94.
- Wittgenstein, L. (1921). Logisch-philosophische abhandlung. *Annalen der Naturphilosophie*, 14 (English translations in C. K. Ogden trans. Routledge & Kegan Paul (1922) and D. F. Pears and B. F. McGuinness, trans. Routledge (1961)).

Issues and Challenges in Instrumental Proof



Philippe R. Richard, Fabienne Venant and Michel Gagnon

1 Introduction

Optics appears to be a mathematical art based on instrumental proof.

Sven Dupré (2006)

The notion of instrumental proof is relatively new; but if the term is as yet of little use in didactic literature, its association with technologies, old and new, seems self-evident.

On the epistemological side, the discovery of Archimedes' palimpsest recently gave us a better understanding of how the weighing method was considered a kind of mechanical proof, and this suggests that the association between proof and artifacts/tools is rather old. Similarly, in computer proofs such as the proof of the four-colour theorem—first shown in 1976 by Kenneth Appel and Wolfgang Haken, then formally addressed in 2005 using Coq software by Georges Gonthier and Benjamin Werner—the decision or the verification of all cases rely on programs, and thus reflect an unavoidable reality of contemporary mathematical work. Whether the tools are physical or logical, their use in a validation certainly renews the common ideas we have about the concepts of proof, modelling and representation of knowledge.

On the didactic side, it seems there is a constant struggle with paradoxes. Nowadays, when a student is asked to prove propositions, she has an automated reasoning

P. R. Richard (✉)
Université de Montréal, Montréal, Canada
e-mail: philippe.r.richard@umontreal.ca

F. Venant
Université du Québec à Montréal, Montréal, Canada
e-mail: venant.fabienne@uqam.ca

M. Gagnon
École Polytechnique de Montréal, Montréal, Canada
e-mail: michel.gagnon@polymtl.ca

© Springer Nature Switzerland AG 2019
G. Hanna et al. (eds.), *Proof Technology in Mathematics Research and Teaching*, Mathematics Education in the Digital Era 14,
https://doi.org/10.1007/978-3-030-28483-1_7

informatics device at her disposal; but at the same time she is required to work with meaningful knowledge of her own, and to transform this knowledge by working more and more at the interface of computer tools that deal with some part of the representation and of the treatment, sometimes experimenting on mathematical objects (e.g. dynamic figures) as a physicist does with objects of his own domain. And all of this happens while the teacher cannot refer to mathematics that could be described as “technological”, since he was educated in a deductive science traditionally developed in writing.

It is then by developing ideas already expounded in our previous work, including the recent paper *The Concept of Proof in the Light of Mathematical Work* (Richard, Oller, & Meavilla, 2016), and adopting the conclusions of our current research projects on the design of the tutorial system QED-Tutrix in high school geometry (Font, Richard, & Gagnon, 2018; Richard, Gagnon, & Fortuny, 2018), and the use of automated reasoning tools in teacher training (Kovács, Richard, Recio, & Vélez, 2017), that we address the question of instrumental proofs, keeping in mind that the interaction between subject and milieu¹ is a unit of epistemic necessity. With this consideration, the subject can be either a reader, considering traditional proofs, or the user of software or of a mathematical machine. The notions of reasoning in action or through algorithms, and of reasoning that unfolds by other means than discourse, will be addressed, as will the Theory of Mathematical Working Spaces, where the question of the coordination of discursive, semiotic and instrumental geneses arises between epistemological and cognitive planes (Kuzniak, Richard, & Michael-Chrysanthou, 2018).

2 Towards an Instrumental Proof

This assumed inference carries with it its own demonstration.

Alexis Claude Clairaut (1741)

2.1 Reasoning, Proof and Demonstration

The notions of proof and reasoning have always been closely linked; but if we can hardly imagine proving a proposition without some use of reasoning, there is no causal link or anteriority of one to the other. The classical definitions of these notions speak rather of operations in a generic sense, which allow the logical sequence of ideas or propositions (for the reasoning), or by which the accuracy of a result is controlled (for the proof). One could believe that the discourse constitutes the essential unifying foundation of these operations. Duval (1995) considers that inference

¹From Brousseau’s theory of didactical situations in mathematics (1998).

is a particular form of discursive expansion, like calculations for processing, and that reasoning is a form of discursive expansion like demonstration, when certain particular conditions of discursive organization are met. But beyond any mechanism of discursive expansion, it must be remembered that proofs or reasoning can also be expressed in action or with tools, and that they may invoke any of several registers of semiotic representation (natural language, symbolic languages, graphs, geometrical figures, etc.). Moreover, this deployment raises questions about the mobilization of registers, their articulation and their coordination. But whether we prove or we reason, we must remember that in addition to natural language and its means of expression there is a whole range of instruments for performing these operations.

We must remember that mathematical proof has its particular conditions of discursive organization. If the demonstration is essentially a text (Barbin, Duval, Giorgiutti, Houdebine, & Laborde, 2001), it allows the introduction of several registers of semiotic representation. In node theory, as an example, Sullivan (2000) shows reasoning steps in which justification relies on figures or graphs. Despite the formal style of the text, the deductive nature of the lemma-theorem-corollary organization and the high epistemic level of the journal, the author adds several graphs to his demonstrations—even going so far as to say, in the demonstration of a lemma (p. 309): “The proof of the next lemma is given in Figure 22”, and to formulate all his reasoning visually in the style of a comic strip. So it is not just a diagrammatic reasoning about the understanding of concepts and ideas, visualized with the use of imagery instead of by linguistic or symbolic means; it is authentic proof, based on a well-constituted semiotic representation register, and it fulfils the functions of a proof (de Villiers, 1993), as understood in mathematics. This type of demonstrative initiative is in line with Proof Without Words (Alsina & Nelsen, 2006) which succeeds in proving mathematical properties by all sorts of semiotic means, scrupulously avoiding the use of natural language. But in these proofs there is always some sort of treatment and control between what we know and what derives from that. Our conclusion is that in mathematical activity, the inferential capacity of the means for expressing the reasoning leads us to a mode of validation, even when conveyed by a tool or a machine.

When, at the beginning of the section, we quote Clairaut, it was in order to go in the same direction. That is, if “this assumed inference carries with it its own demonstration”,² it was not the type of rationality that supported the shift from “presumed” to “established”, but the inferential nature of operations that consists in acknowledging a result by virtue of its connection with other results already acknowledged. It is therefore not the type of rationality that carries itself from inductive to deductive reasoning, nor the same registers that are at stake. We can even add that in its inductive argument, Clairaut tries to make dynamic a figure by reasoning, playing on the fact that the text will be read and that the model reader will be able to visualize

²This translation was provided in 1881 by J. Kaines from the edition of the *Éléments de géométrie* published in Paris in 1830. In the original in French (“cette induction présumée porte avec elle sa démonstration”), Clairaut (1741) attempts to convince the reader of the relevance of a conjecture through inductive reasoning, before embarking on a deductive demonstration.

the animation while, in his demonstration, a classical figure like those in Euclid's *Elements* is proposed. In current terms, we can easily express this idea of inferential connection with the following functional notation:

$$f(\text{antecedents}) = \text{consequent},$$

where the “antecedents” are the previously acknowledged results and the “consequent” the newly acknowledged result, according to the connection of epistemic necessity f . We thus approach didactic definitions of reasoning, such as the one given by Balacheff (1987), where the reasoning designates “the intellectual activity, mostly non-explicit, of manipulating information to, from data, produce new information” (p. 148), while specifying that the type of “inferential” connection is also “of epistemic necessity”. If the quality of being necessary is shared, at a given moment, within a community, it is indeed a proof, extending the meaning given by Balacheff (1987) where “signification is the requirement to determine a validation system common to the interlocutors” (p. 148). Thus, by seeking his reader's conviction, Clairaut wants him to first join the community of those for whom his conjecture is necessary (inductive proof), to the point of showing him why it is mathematically necessary (deductive proof).

From an instrumental-proof perspective, we will examine the nature of f and the issues and the challenges that it poses; this is the main purpose of our paper. But before understanding more specifically what we mean by “instrument”, we must clarify what is mathematical work.

2.2 *Mathematical Work and Mathematical Thought: A Temporal Invariance?*

If writing is so important for the expression of reasoning, it is because mathematical models began with the first written documents. We can even suppose that it is at that time that mathematical science begins. If our distant ancestors had a form of mathematical thought, which we can describe as protomathematics, they had to interact in one way or another with the objects they could represent, beyond the act of visualization itself or the implicit treatment then imposed. It is known that, long before writing, Homo sapiens and their Neanderthal cousins could express a symbolic thought with geometric shapes that seem to be evidence of premeditated creations (Hoffmann et al., 2018). But in the absence of direct testimony, we do not know whether they reasoned with or about these forms, or whether these forms had any instrumental function. Conversely, the oldest potentially mathematical artefacts, such as Ishango's bone, seem to tell us more. Although the problem of prehistoric archaeological (strictly speaking), ethnographic and didactic sources calls for avoiding any over-interpretation (Keller, 2004), we can have in mind that if these artefacts had any mathematical function, it was the production of new information, like a

calculation or some reasoning. In other words, even before the invention of writing, a form of mathematical thought had to exist to preside over the interaction between reality and what would become models, the expression of this interaction implying both signs and tools. In short, mathematics has been an instrumented activity since its beginnings.

Even today, and to limit ourselves here to the world of education, it is not easy to distinguish whether it is mathematics (as a science) or mathematical thought (that does not necessarily proceed by writing) that is at stake (in a given situation/task/activity). In Canada, for example, a non-profit organization devoted to promoting chess in schools proudly displays its benefits:

When learning the movement of the knight, the bishop and the rest of the pieces, did you know that you were doing geometry? Yes, I assure you.³

If one considers mathematics a science, this assertion seems absurd: chess shares neither rationality nor means of expression with geometry. Of course, one can do mathematics when modelling some part of a game in graph theory, possibly with the help of a computer to support the discovery of a winning strategy. Such activity does not reflect the specifics of the game itself—but if the sentence above is replaced by: “when learning the movement of the knight, the bishop and the rest of the pieces, did you know that you were developing your geometrical thought?” one seems more inclined to answer “yes”, just as one needs a sense of numbers to do arithmetic or a sense of structure to do algebra.

We can push the question a little further: does the pupil do mathematics when programming with Scratch?⁴ We can easily propose in the classroom tasks defined in mathematics (or projects, as in designer jargon like Boutin, 2017), knowing that the means of expression of the graphic signs manipulated to accomplish the task (to realize a project) remain rather close to mathematical writing. At the same time, the combination of these elements is instrumented by the gesture at the interface of the computing device, which somewhat distances us from the writing. The choice and ordering of the elements evoke the development of some deductive reasoning or proof—such as a construction protocol in geometry too. But we can change the order of elements (or some parameters) dynamically in noting the effect of its algorithm on the interface, even when it is not completely executed. In this situation, the student develops his mathematical thinking with Scratch in terms of expression, reasoning and proof (when checking and executing an algorithm), and is thus “doing mathematics” as an instrumented activity, similarly to what was done at the time of Ishango’s bone, 20,000 years ago.

³This is the association *Échecs et Maths*, a pun in French that also means “checkmate” (retrieved April 17, 2018 from <https://echecs.org/les-bienfaits-des-echecs>).

⁴Coding and algorithmic learning platform using a visual and dynamic programming language in which programs are designed by assembling graphic elements (accessible April 17, 2018 at <https://scratch.mit.edu/>).

2.3 The Mathematical Working Space

In this mathematical science and activity, both sides of the same coin, which is studied and practised, we consider that mathematical work is the visible part of mathematical thought. For over ten years, the concept of mathematical work in mathematics didactics has been the object of collaborative research among various researchers, mainly from French and Spanish speaking countries (Kuzniak, Tanguay, & Elia, 2016). The Mathematical Working Space theory (MWS) aims to provide a tool for the specific study of mathematical work engaged during mathematical sessions. Mathematical work is progressively constructed, as a process of bridging the epistemological and the cognitive aspects in accordance with three yet intertwined genetic developments, identified in the theory as the *semiotic*, *instrumental* and *discursive* geneses (Kuzniak & Richard, 2014). MWS appears as a theoretical and methodological model that allows one to report on mathematical activity, potential or real, during problem solving or mathematical tasks. In particular it allows the description of dominant interactions, whether finalized or not, depending on the nature or issue of significant moments (e.g. didactic interactions during the devolution of a task, didactic interactions while solving a proof problem, etc.). In the next section, we interpret the types of proofs in the light of the MWS.

The MWS model is presented in a basic form as a skeleton to which different frameworks or theories ‘add flesh’, depending on the questions, problems or difficulties involved in a research study. Thus, in Fig. 1, the vertical planes are related to different phases of the mathematical work, as discovery, reasoning and commu-

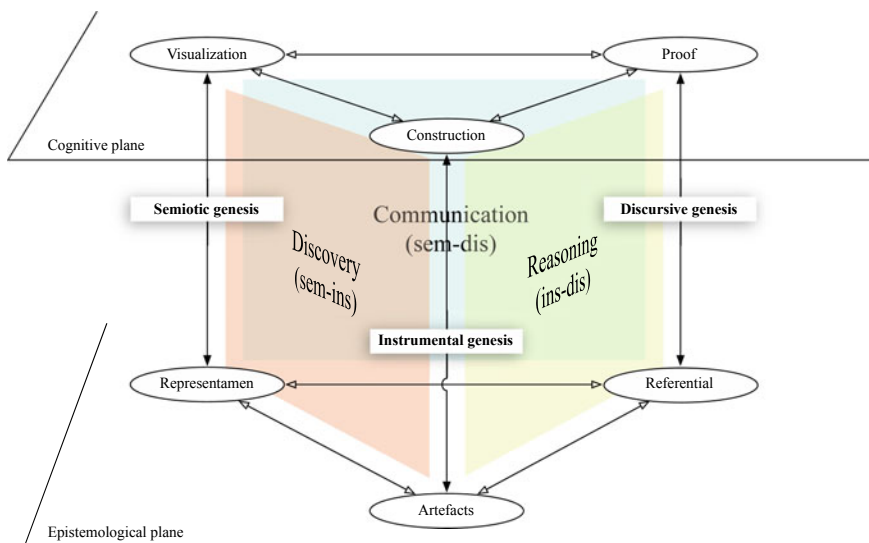


Fig. 1 The components of the MWS from Kuzniak and Richard (2014) in which the concept of proof is traditionally related to the epistemological plane by the discursive genesis

nication in a broad sense of mathematical competencies. The effective realization of these phases can define, for example, some cognitive mathematical competencies based on the coordination of the geneses, in order to think the integration of the phases of mathematical work. However, in the base form of the diagram, these planes are presented only with the generic labels *sem-ins*, *ins-dis* and *sem-dis* so that the coordination of the geneses can be adapted to the task at hand. In Sect. 3, we will take advantage of the adaptability of the model to describe our instrumental proofs.

The concept of fibration has been suggested to label moves, transitions and specific activities between the different elements of the MWS (Recio, Richard, & Vivier, 2015; Tanguay, Kuzniak, & Gagatsis, 2015). In Fig. 2, we see the internal fibrations that can intervene in the process of conceptualization, during both the formation of a mathematical conception and its implementation. In addition to internal fibration (between planes, between poles, between registers of representation, etc.), the model considers external fibrations in the same logic between some MWSs from various mathematical domains such as: during intra-mathematical modelling activity between analysis and statistics (Derouet, 2016), extra-mathematical modelling with Physics (Moutet, 2016) or transversal modelling with algorithms (Laval, 2018). The MWS model can then be articulated in a wide range of situations involving a mathematical task, at one time or another.

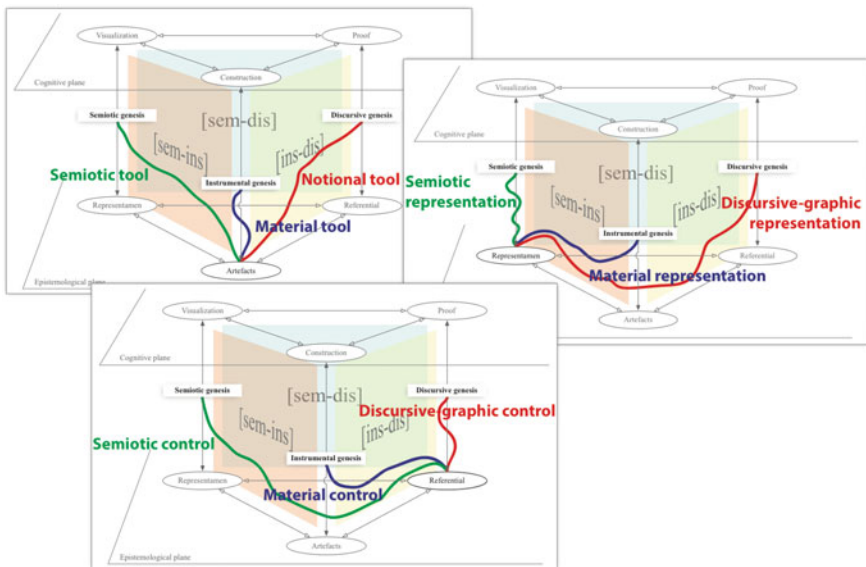
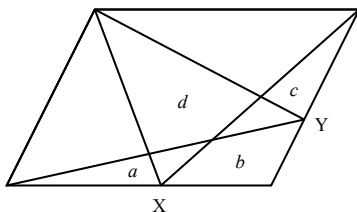


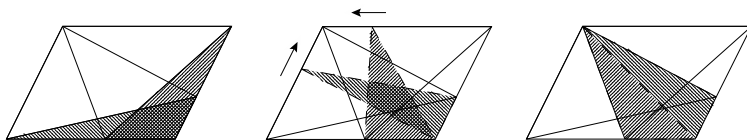
Fig. 2 Internal fibrations in an MWS from Lagrange, Recio, Richard, and Vivier (2017) that shows the roles of the tool (operational means), of representation and control that have been retained within the model

2.4 Instrumented Reasoning, Instrumental Proof

So far, we have used the notion of instrument to designate either a tool or a system of signs (diagram, figure or other registers of semiotic representation) used by someone in order to get something done. With such a broad definition, we could consider the many figural inferences we can find in Alsina and Nelsen (2006) as some kind of instrumented inferences. Thus, to prove the equality of areas $a + b + c = d$ in the partition of the following parallelogram (Richard, 2003):



the following sequence of figures is used as justification:



It is then a question of a structure inference: $f(a + b + c, \square, d) = (a + b + c = d)$, where f is the inference that emerges from the sequence of figures and \square , the signifying figural propositions from the top figure. Can we really say that this is instrumentation? In terms of the MWS model, the visualization of the properties represented by f is typical of semiotic genesis. One can also grant a role of semiotic tool to the sequence of figures (fibration), because the material support and the sequence itself, as a means of action animated by the reader, are not specific to the figural register, and allow him not only to view a movement but also to compare areas going back and forth from one figure to another. If, instead of a sequence of figures, the situation was set at the interface of a dynamic geometry software (see Appendix 1), it would be easy to interpret the situation as one of instrumentation in the user-milieu interaction (IT milieu). However, we will see in Sect. 3.1 that the activity of reading a figure and its interactive manipulation generate different connections of epistemic necessity. In our example, we consider that linking the rationale to the antecedents and to the consequent involves the discursive genesis: one can recognize here a step of discursive-graphical reasoning (in the sense of Richard, 2004a, 2004b), highlighting the coordination of these two geneses (sem-dis plane).

In the French didactic tradition, the notion of instruments is inspired by the work of Rabardel (1995). In his cognitive approach to contemporary instruments, it is no longer “the artefact that is explicitly or implicitly considered as the instrument” (p. 4), but a new entity that is both a subject and an artifact. According to Trouche (2005): “The word ‘instrument’ will designate a mixed entity consisting of ‘the technical object and its modes of use’ constructed by a user” (p. 93). Thus, since the

instruments are not immediately given to the user, he must develop their use through his mathematical activity of instrumental genesis. In this perspective, we consider that a figural inference⁵ as instrumented during the recognition of figural invariants (inductive) or instrumented construction (deductive), whether by the intervention of the ruler and the compass, a software of dynamic geometry or “mathematical machines” (see especially Bartolini-Bussi & Maschietto, 2005, but also Bryant & Sangwin, 2008). In his work, Rabardel does not address the question of proof, and he speaks little of reasoning. He mentions, with regard to Gérard Vergnaud’s theory of conceptual fields, the inferences (reasonings) which allow the treatment and the anticipation from the schemas of a subject, while specifying that “there is always a lot of implicitness in a schema, and therefore difficulties in making it explicit for subjects.” (Rabardel, 1995, p. 88). Rather, this limit encourages us to exploit the subject’s interaction with a milieu to speak about reasoning.

Besides, the very idea of instrumental proof is not at all contemporary. Recently, Cormack (2017) stresses the importance of so-called practical mathematics in early modern Europe, in order to show how the mathematization (and modelling) of natural philosophy became an investigation of the interplay between useful mathematics and its practitioners, and natural philosophers. For example, the book explains that cartographer Edward Wright first explained Mercator’s cartographic projection, providing an elegant Euclidean proof of the geometry involved. It is a typical modelling approach between, on the one hand, a cartographic representation system (the situation model, in the meaning of Blum & Leiß, 2007; see also Fig. 3 for details) and its mathematization in geometry on the other, the proof remaining attached to a discursive genesis activity within the mathematical model. But a diametrically opposed approach is also evoked in the collective book, which we readily assimilate to a cycle of “antimodelling” that starts from the mathematical model to be interpreted first in the reality of a situation model. We particularly consider as such Archimedes’ mechanical approaches, like the weighing method to discover the area of a parabola

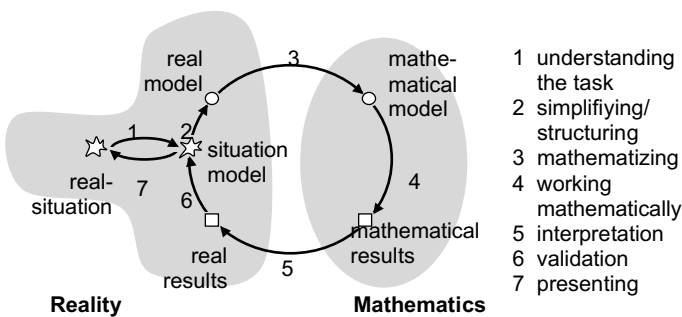


Fig. 3 Modelling cycle from Blum and Leiß (2007). According to Borromeo-Ferri’s (2006) cognitive point of view, the situation model is a mental representation of reality

⁵See Richard (2004a, 2004b) for a definition of a *figural inference*, and Coutat, Laborde, and Richard (2016), for the *instrumented figural inference*.

segment or the volume of a ball (Netz & Noel, 2008). In this approach the validity is assumed by the physical coherence under the constraints of a proper use of the method, before going back to the mathematical model to infer the areas or the volumes. Unlike Wright's problem, Archimedes's task is set in mathematics, so in order to solve a mathematical problem Archimedes needs his method to be anchored in physical reality.

Although the subject goes well beyond the purpose of our article, we can briefly raise the question of the validity of this mechanical proof. If it is at least a heuristic means of great pedagogical value—in this sense, Archimedes and Clairaut participate in the same movement—it can be restricted to an empirical method which would need mathematical discourse to give it a high epistemic value, understanding that Archimedes might have kept hidden in his drawers a more formal proof equivalent. Moreover, as Keller (2017) says:

For Plutarch, his biographer who lived long after Archimedes, the great mathematician could not really have found intellectual satisfaction in his machines; he could only have meant them to serve to impress the vulgar (...) who could not appreciate more abstract ideas. Supposedly for Archimedes and Plato (Plutarch would have assumed), mathematical theorems and proofs dealt with ideal situations and one should not think of them as applicable to real life, which is by necessity so untidy. (p. 116).

Nevertheless, Vitrac (1992) shows that some scientific historians have long considered that the mechanical method is mathematically admissible. That is, if he does not deny that Archimedes “never assigns the status of proof to the mechanical method” (p. 76), it testifies to an Archimedes “axiomatizing mechanics [which] includes at least a part of it in geometry” (pp. 75–76). From this we deduce that physical coherence is induced not only by matter itself, but also by a scientific method that enables the interpretation of what is happening. So that if this type of proof appeared as justification for an inference, to the epistemic necessity connection f , it must be associated with its instrumental validity, as much in terms of use (of the method, of the artifact) as of the machine (its constitution, its domain of validity).

From the mental stress from which theory and practice derive, Dupré (2017) gives the example of the way in which Ettore Ausonio, mathematician and instruments creator,⁶ appropriated the reading of Witelo's optical treaty *Perspectiva*:

His teaching was based above all on his reading of Witelo's *Perspectiva*, but his lecture notes reveal highly selective reading practices. These notes listed only Witelo's descriptions of the instruments to measure reflection and refraction and those propositions in which the Polish perspectivist claimed the use of these instruments as proof of the proposition. Ausonio left out all of Witelo's propositions not established with the instruments, and in the selected propositions, he discarded the geometrical demonstrations and the geometrical diagrams. In sum, Ausonio appropriated Witelo's optics in such a way that optics appeared to be a mathematical art based on instrumental proof. (p. 140)

⁶We also write “Ettor Eusobio”. The Thesaurus of the Consortium of European Research Libraries (CERL) considers him as an instrument maker (from German “instrumentenbauer”, see <https://thesaurus.cerl.org/record/cnp02134922>), and in the 1678 edition of Leonardo Fioravanti's *Dello specchio di scientia universale*, it is said of him: “the great philosopher and mathematician, Mr. Ettor Eusobio da Venetia; inventor of the most beautiful mathematical material ever seen” (p. 94).

Compared to Archimedes's method, Ausonio's approach seems quite radical. Because not only does he avoid the geometrical demonstrations, he rejects geometrical diagrams. His mathematical activity would be limited mainly to the instrumental genesis, or if we prefer to reuse the spirit of the "mathematical art based on instrumental proof" by Dupré, it reflects a competence and a practice of validation requiring planning and intelligence, as an art. Unlike Archimedes, who deals "with ideal situations" in mathematics, the initial questioning of Ausonio, and the purpose of his work, is in op. And unlike Wright, who seeks to validate mathematically, Ausonio wants to perform a validation that remains in the universe of his instruments (artifact sense). From these considerations, we draw a first type of instrumental proof in mathematical work, the *mechanical proof*, which proceeds essentially by coordination of semiotic and instrumental genesis. In choosing a term relating to mechanics, it is not so much to highlight the fact that the proof depends on the operation of a machine or a mechanism, but that the justification is based on some sort of laws of motion or balance that objects exercise in relation to each other. This definition allows us, first, to account for certain proofs that already exist in mathematical education, such as the mathematics of, in and for the reality from Emma Castelnuovo or those that frequently emerge from pedagogical initiatives (Fig. 4), and it is consistent, second, with the use of mathematical machines, and especially with dynamic geometry software. For the "laws of motion" and the "balance of forces" relate to the operation of the software and the logic of the construction of the figure, with the particularity that by acting on the figure or on its elements, the user also acts on the register of semiotic representation, which is not seen in Archimedes's approach. From a didactic perspective, our point of view is the idea of the physicist geometer described by Tanguay and Geeraerts (2014).



Fig. 4 Examples of mechanical proofs of Pythagoras's theorem in mathematical education. The first on the left is justified by comparison of weights (Castelnuovo & Barra, 1976) and the second, on the right, by the transfer of liquid volumes (YouTube, 2009) (Extract from the original video entitled *Pythagorean theorem water demo* (retrieved August 3, 2018 from <https://youtu.be/CAkMUdeB06o>))

2.5 Algorithmic and Proof

Various algorithms were already known in antiquity, in arithmetic or in, geometry, including, among the most familiar:

- rules for calculating the length of arcs and the area of surfaces, in Egypt and in Greece;
- several methods for solving integer equations, following the work of Diophantus of Alexandria in the 4th century AD;
- the Euclid algorithm (c. 300 BC) that calculates the greatest common divisor of two natural numbers;
- the calculation schema of the number π due to Archimedes.

If there were to be a tool-object dichotomy at play, like the theory-practice dichotomy in Sect. 2.4, it would only be purely functional. Because the determination of geometric measurements or the approximation of irrational numbers like π refers to the essential nature of real numbers and suggests, as Gray and Tall (1994) show in a learning context with the notion of *procept*, that mathematical objects are formed by encapsulating processes. An algorithm therefore solves not a single problem but a whole class of problems differing only by the data and the specific course-of-values, but controlled by the same requirements—that is, it must operate with certainty regardless of the given problem.

Let's imagine Archimedes today trying to solve a problem of counterfeit coins with his weighing method and a Roberval scale. The problem goes like this:

In a set of coins, indistinguishable by sight or touch, there are false coins. Real coins all have the same weight, and so do counterfeits, but their weight is different from that of real coins. With the help of a scale and without being able to have a reference weight, how can Archimedes find the counterfeit coins? Which is the method that would allow him to find them with as little weighting as possible?

Such is the problem posed by Modeste, Gravier, and Ouvrier-Bufferet (2010), but without Archimedes (!), to pre-service teachers, playing on the constraints of weight and the number of counterfeit coins. This problem does indeed concern an algorithmic problematic, because it is a matter of elaborating an effective method to search for counterfeit coins, proving it and studying its optimality, e.g. by means of successive weighing operations or trees. In this type of open problem, students need to think about both the definition of the situation and the search for valid solutions, the design factors associated with the problem, the plausible creative processes in some kind of iterative problematic and an innovative line of reasoning behind the constraints, factors and the design choices made. This activity can be seen as a problematic-modelling dynamic between reality and the mathematical world (Fig. 3) which allows, under certain conditions, to get some results and enrich the understanding of reality (Fig. 5). According to the authors: “this problem and most of its variants, are not yet completely solved as far as mathematical research is concerned” (p. 61), which presupposes from the outset that it will be necessary to be inventive. The algorithmic approach would first be used for programming a mathematical work by

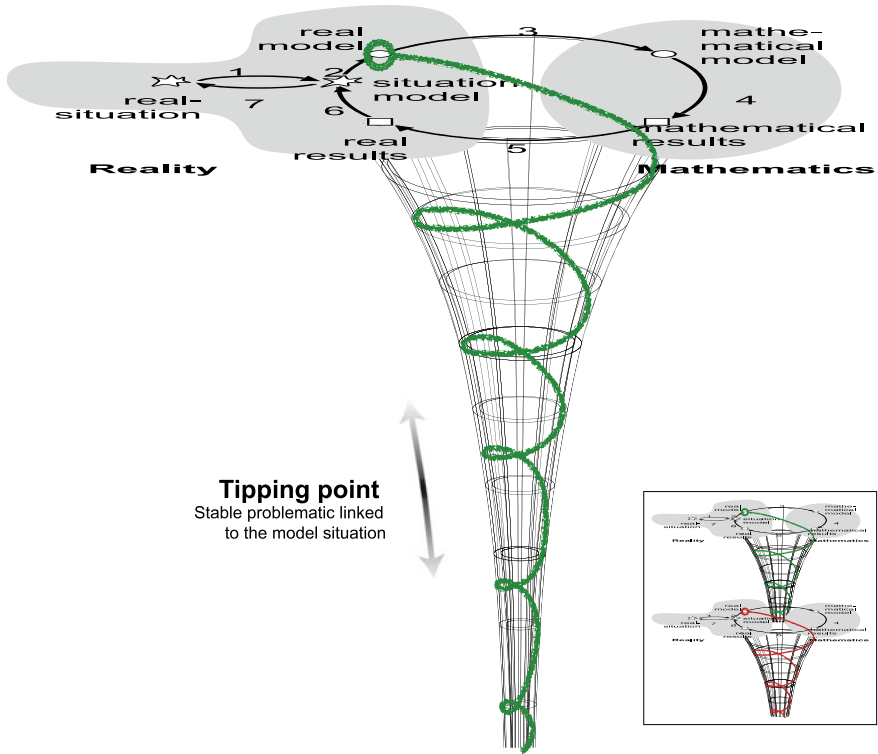


Fig. 5 When we attack a complex real problem, the modelling cycle (Fig. 3) is characterized by some back and forth processes that we wish to converge (infundibuliform path), at least until we obtain a stable problematic. At this tipping point, the cycle can continue for results (cycle tightening), or the real situation can be rethought by considering new constraints (cycle widening). The situation model is modified and the cycle starts again (vignette)

gradually approaching a reference situation through successive problem solving or judiciously considered cases.

Traditionally, algorithmic is the study and production of rules and techniques that are involved in the definition and design of algorithms, which are structured steps that transform well-defined inputs into desirable outputs. We can then talk about disconnected algorithmic. However, the algorithms can be encoded in a machine to reproduce these steps very efficiently, especially when it comes to solving complex problems that require a large computing capacity. Moreover, to function effectively, i.e. to ensure that it is executed quickly and that it converges towards desirable outputs, the algorithm must be stable and not modify itself. Otherwise it would be autonomous, even intelligent (in the sense of adapting to a new situation), and we could no longer predict or optimize how it would work towards the desired results. This algorithmic determinism may seem the complete opposite of human intelligence, which must constantly adapt to the unexpected. But it is this capacity for human adaptation

that makes it possible to govern the solution of a complex problem, in the sense that it is the human being who breaks down the problem, chooses the constraints, builds the necessary algorithms, possibly delegates the execution of the algorithm to a machine, and carries out the necessary verification (validity of the algorithm, encoding, response, calculability in a reasonable time, etc.). And it is also s/he who, if necessary, remodels the model situation to enrich the understanding of reality, the dimension of the initial problem or the validity domain of the resolution process that s/he has thus systematized. This human attitude to solve a problem in such a dynamic and which invites to “think about the tasks to be accomplished in the form of a series of steps” (cf. Venant, 2018, p. 58) is what we call the algorithmic spirit.

To remain in the mathematical sphere, the idea of a line of action or a series of operations proposed to achieve a result is highlighted by the algorithmic approach in solving the areas problem developed by Trahan (2016)⁷:

Let any two polygons have the same area. It is possible to cut the first polygon into a finite number of pieces and then reform into the second polygon.

This is the classic statement of Wallace-Bolyai-Gerwien’s theorem. Exactly as in the previous problem, the statement is simple and it is by playing on constraints that we succeed in demonstrating this result, in verifying certain effects and wondering about possible implications or generalizations, whether it be to curvilinear figures or higher dimensions. Without entering into a particular formalization, we can show the structure of its approach in four steps (Fig. 6). Although this does not appear in our paper, the author carefully demonstrates each of his propositions: his mathematical work is essentially activated through a discursive genesis, with several steps of discursive-graphic reasoning when he considers particular examples. But his reasoning is broader than that. Although it is easy to describe a posteriori his approach as systematic and to see only a series of traditional demonstrations, possibly presentable in the lemma-theorem-corollary mode, his program is an inventive mathematical work that shows vividly the search for a validation of the original problem. Despite this connection between the heuristics of mathematical discovery and validation by this type of proof approach, algorithmic is much more than a mathematical way of working. Moreover, this algorithmic spirit, already implicitly present in teaching, could be worked out and brought to light thanks to algorithmic instruments.

Trahan’s attitude is close to the way that mathematicians treat long proofs. By the 1980s, according to Krieger (2004), a variety of rigorous proofs were provided of various fundamental facts about our world, many of them lengthy and complex and involving much calculation (Krieger, 2004):

Actually, many of the preliminary theorems motivate the proof and indicate what is needed if a proof is to go through. And the lemmas might be seen as lemmas hanging from a tree of theorems or troops lined up to do particular work. As in many such calculations, the result almost miraculously appears at the end. (...) The achievement (of lengthy and complex

⁷The author explained that part of his resolution program is inspired by the website *Choux romanesco, Vache qui rit et intégrales curvilignes* accessible from <http://eljjdx.canalblog.com/>.

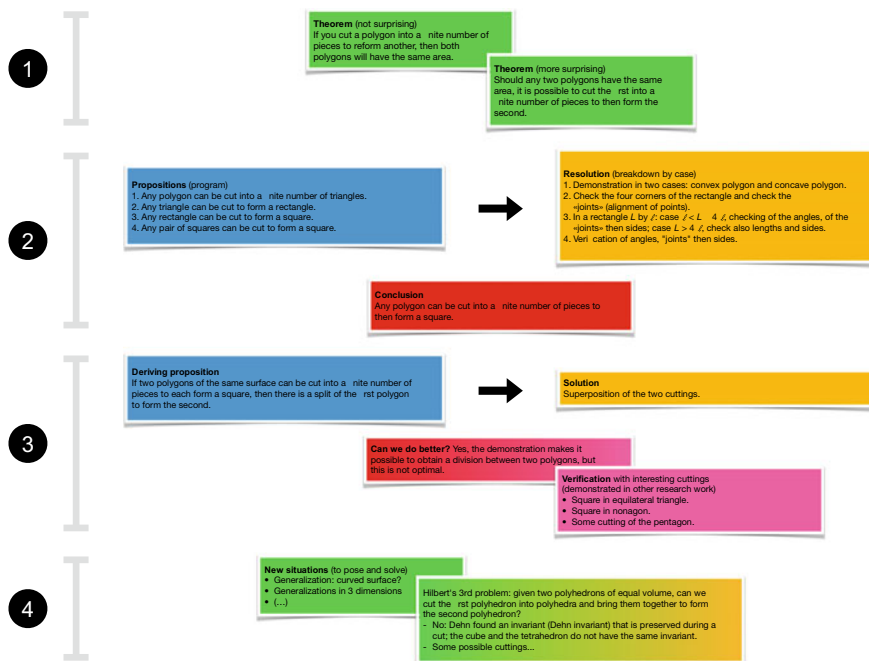


Fig. 6 Structure of the process of solving of the Area Puzzle Problem from the Trahan’s algorithmic approach (2016): setting the situation (part 1), first resolution program (part 2), second resolution program with effect verification (step 3) and opening towards new problematizations with results anticipation (step 4). The derived proposition in step 3 is an illustration of a modified situation model as suggested in the vignette in Fig. 5. For the convenience of the reader, a transcript of the contents of the boxes is presented in the Appendix 2

proofs) is again the ability to divide up the problem into tractable parts, to orchestrate the parts so that they work together, and to be able to tell a story of the proof. (p. 1227–1228)

Some of these “facts about our world” begin with questions initially asked in physics, but the proofs in question do indeed stem from mathematical work. We consider that this idea of orchestration of parts, which works with the same goal of proof, is typical of the algorithmic spirit. Of course, a general algorithm makes it possible to produce a proof and can possibly constitute other algorithms to verify parts of it, as it is also done with traditional proofs of lemmas.

Besides, in the research of Modeste et al. (2010), the natural link between the algorithmic approach and the learning of proof is particularly emphasized:

The algorithmic approach uses arguments that are common to mathematics, but also to a specific way of thinking. It seems essential that the study of the algorithm as an object allows for a challenge to this way of thinking. A central concern of mathematics is to provide demonstrations of its results, so it is inevitable, if one looks at the algorithm as an object of mathematics, to query the algorithm-proof link (p. 55).

Even if it is not a surprise for the authors, the tone is set: the algorithm is not only a mathematical way of working, but also an object that must be implemented in school. At the same time, since the introduction of algorithms as objects, new questions emerge, such as the efficiency of the algorithm in terms of the number of operations or the memory necessary for execution, all of which may divert thematic work. Anyway, computer science has changed mathematics, in particular by allowing objects to be studied from new perspectives, by bringing new questions, by creating emerging mathematical fields that are now booming, and by transforming the mathematician's activity thanks to new tools (*ibidem*, p. 57). In addition, computer programming allows some formalization of reasoning to support the student in mathematical tasks like derivation, discovery, and proving geometry statements (Kovács et al., 2017) or in learning mathematical proof (Tessier-Bailargeon, Richard, Leduc, & Gagnon, 2017).

2.6 *Another Instrumental Proof*

At an epistemological level, many situations ask for rethinking mathematical work: too many cases to study, intrinsic inability to demonstrate by mathematical induction, research by repeated tests or the conjecture invalidation iterations are not convergent, etc. The proof of the four-colour theorem, which we cited in our *Introduction*, is an example of a mathematical work that we cannot yet achieve without a computer. In a critical commentary that goes back some forty years, Appel and Haken (1977) already said:

Our proof of the four-colour theorem suggests that there are limits to what can be achieved in mathematics by theoretical methods alone. It also implies that in the past the need for computational methods in mathematical proofs has been underestimated. It is of great practical value to mathematicians to determine the powers and limitations of their methods. We hope that our work will facilitate progress in this direction and that this expansion of acceptable proof techniques justifies the immense effort devoted over the past century to proving the four-colour theorem. (p. 121).

The fundamental problem for the acceptability of this type of proof is that, despite its great interest, algorithmic thinking does not convey a mode of validation in itself.⁸ On the contrary, the theorem-demonstration problem moves towards a problem of validation of the algorithm and its execution by the machine. In other words, you must be sure that the algorithm is achieving the proper result, whatever the pending problem might be. In a very pragmatic way, it often happens that computer experts validate an algorithm by using verification methods. But in computability and complexity, one usually tries to prove mathematically the effectiveness of the algorithm, even if it

⁸Although we know that solving algorithmically a problem of proof and using a program to verify a set of particular cases is not the same thing, we can state as Clairaut said that an algorithm often carries with it its own demonstration. This wink to our quotation of Clairaut in an algorithmic context comes from Simon Modeste.

is necessary to reformulate its domain of validity. The algorithmic way of thinking, as a characteristic of mathematical work, as well as the execution of an algorithm by a machine, suggest the existence of a type of instrumental proof different from mechanical proof. We call this type of proof *algorithmic proof*: it usually proceeds by coordination of the discursive and instrumental geneses.

In some cases, the learning of the proof is instrumented by a computing device and it can go through an algorithmic process, constituting rightly an algorithmic proof according to the definition given above. Thus, in secondary school, we can mention the resolution of a problem of proof at the interface of the QED-Tutrix tutorial system (Leduc et al., 2017; Leduc, 2016; Tessier-Baillargeon, 2015). In this kind of tutorial system, we tend to support the student in elaborating a deductive proof, mostly by forcing him to construct his proof by forward chaining (from the hypotheses of the problem to the conclusion) or backward chaining (the other way round). However, the QED-Tutrix system accepts that the student can enter the propositions of his demonstration in whatever order he wants, whether they are assumptions, justifications or results that he deduced or he supposes are valid (Fig. 7). It even happens that the student tests the system itself by watching its reaction to one of his propositions (instrumental genesis). But the system is programmed to support the student in the logic of the problem, and to lead him gradually to write a deductive proof (discursive genesis).

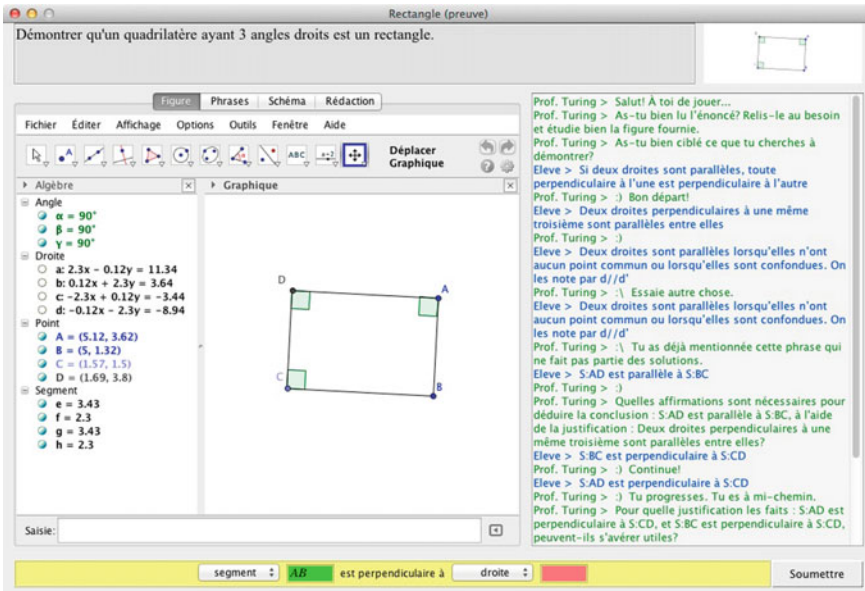


Fig. 7 QED-Tutrix interface, an intelligent tutorial system that supports solving problems of proof in geometry. From the problem statement (top), the student writes propositions (bottom), acts on the figure of the Geogebra dynamic geometry module (left) and discuss with an artificial pedagogical agent (right)

The instrumental proof would thus not materialize only by the engagement of artefacts, tools or methods to a process of mathematical proof. It would also open up to physical or algorithmic modes of thinking, renewing several questions about the epistemic necessity, the control of knowledge or the decision-making that arise from different types of reasoning. The example of proof learning and its quest using QED-Tutrix testifies that deductive reasoning is not necessarily an instruction tool for learning demonstration, contrary to what is sometimes heard among supporters of the old school.⁹ This is also what Brousseau (2004) reminds us with his paradoxes about the study of the teaching of reasoning and proof:

The “orthodox” presentation of mathematical texts gives the impression that formal logic (modus ponens, with perhaps a few other tools of logic) is the fundamental and necessary instrument of mathematics, and that the aim of mathematics is to demonstrate that its author has not produced any contradictions (with himself or with known mathematics). Many teachers tend to deduce from this that since mathematical reasoning is the sole means of establishing publicly that a mathematical statement is true, this reasoning must also necessarily describe (or serve as a model for describing) the thinking that correctly constructs mathematics, hence that describes the thinking of mathematicians and of students. As a result, they try to teach directly how to think, and then how to reason as one does in making a proof. They thus mix up the activity and mathematical reasoning of the students with their cultural product: the standard method of communication.

If, on the other hand, one assumes that the natural functioning of thought produces exact knowledge by some processes (rhetorical, heuristic, psychological...) which cannot be reduced to the presentations and notations that are most convenient (for mathematicians and their research), then what are those processes, and how can they be realized? or how can someone be led to achieve them?

Certainly, one might fear that by its versatility, instrumental proof inherently creates a problem of validity for mathematical theory, or for mathematics that is constructed as a science. Nevertheless, mathematical activity is much more than the slow establishment of a non-contradictory reference system, and this is even truer when it comes to mathematical work at school. But it is necessary from the outset to distinguish the mathematical work of the student from that of the teacher: the latter must develop not one but two kinds of geneses: one of them personal, for his or her mathematical work, and the other one professional, for teaching (see Haspekian, 2011, about a double instrumental genesis; Derouet, 2016, on the student-teacher contrast or Leduc et al., 2017, for the genetic distinction between mathematical work and didactic work). As a result, in addition to appearing legitimate from a didactic and epistemological point of view, the instrumental proofs become particularly useful in answering Brousseau’s questions about “those processes”.

⁹It is still common to hear at conferences that automated reasoning, however used, is not appropriate for learning mathematical proof in school. Even very recently, a colleague showed us an anonymous assessor’s comment for the evaluation of an important project: “automatic proofs are not necessarily suitable for educational purposes, therefore most parts of action are somewhat out of interest in this context”. This point of view seems to us similar to the one where, in the 1970s, it was feared that students would no longer learn to calculate, after the introduction of calculators in schools. Paradoxically, this artefact is today practically tossed in the dustbin of history in favor of tools once unthinkable.

3 Three Types of Proof in the Mathematical Work

I would also claim that, in a very specific sense, mathematical work is a form of philosophical analysis.

The mathematicians and mathematical physicists find out through their rigorous proofs just which features of the world are necessary if we are to have the kind of world we do have.

Martin H. Krieger (2004)

3.1 *In the Model of the Mathematical Working Space*

By introducing the mathematical working space theory, we were able to deal with the notion of proof both in school and in the work of the mathematician. Even if, since Chevallard's anthropological theory of the didactic (1992), we have a better understanding of why school mathematics is not a reduction of scholarly mathematics, Kuzniak et al. (2016) remind us that the MWS "involved in mathematics education, are related to different kinds of vigilance—epistemological, didactic, and cognitive" (p. 729). Moreover, among students who are destined to become mathematics teachers, the knowledge conveyed often moves from one mathematical paradigm to another, to the point that even the meanings of the mathematical concepts at stake seem to be agglutinated (Arzarello, 2006; Tanguay & Venant, 2016). Facing the inescapable complexity of mathematical work at school and in order to set our types of proof according to MWS, we have to make assumptions about the mathematical task, the subject-milieu interaction, the proof/reasoning nesting and the valence of the proof activity in mathematics, as follows.

1. The proof results from a mathematical task and not from a problem set in physics or in algorithmics. For example, some situations in special relativity at school involve Minkowski diagrams at the interface of a dynamic geometry software, playing on rationality frameworks in both physical sciences and mathematics (Moutet, 2016). In such a case, we must consider two epistemological planes, modelling links between these planes and external fibrations respectively between elements of the semiotic and instrumental geneses of each plane. If a mechanical proof had to be recognized in the resolution of the initial situation, then it would be necessary to start from a well-identified mathematical task during the resolution process and to deal with external coordination issues between geneses. Similarly, some algorithmic situations are already close to mathematics, such as introducing the dichotomy algorithm to find a square root bounding of prime numbers, or determining the zeros of a function by the sweeping-out method (Laval, 2018). In each case, one can start working with some software like Algobox¹⁰ and ask the software to test the algorithm that has been composed, evoking the algorithmic proof. Nevertheless, the transition between the worlds of Algorithmics

¹⁰Freeware by Pascal Brachet (2018) available at <http://www.xmlmath.net/algobox/>.

(A) and Mathematics (M) is not symmetrical: although the $A \rightarrow M$ direction is essentially based on a mathematization, the $M \rightarrow A$ direction is closer to a conversion, or even to intramathematical modelling. In the problems of bounding or determination of zeros, the mathematical work is expressed conceptually in a dynamics of *K(epsilon)-game* (Bartle & Sherbert, 2011) because of the very nature of real numbers (see Sect. 2.5). The activation of an MWS and its proof types would start at the same time as the mathematical task.

2. The proof is defined by the interaction between a subject, who can be a reader and/or a user, and a milieu. This idea of interaction is inspired by the didactic situations described by Brousseau (1998) and from the model to reason on learners' conceptions of Balacheff and Margolinas (2005). We have previously shown that the epistemic necessity is characteristic of this interaction for the instrumented learning of properties in geometry (Coutat, Laborde, & Richard, 2016) and that the type of proof of the same property may vary depending on the nature of this interaction (Richard et al., 2016). It follows that even traditional proofs are not the prerogative of individuals, but also of the milieu that supports them (paper-pencil, computers, etc.). When a proof involves an automated reasoning tool, the interest for the process is not in the calculation by the machine, as a tooled necessity, but in the questioning that triggers the calculation and in the interpretation of the results. If a proof encapsulates a procedure by semiotic, instrumental or discursive means, one must be able either to explain how the procedure works, despite the black box effect that could result, or to vary it with an additional, compensatory or confirmatory proof. The MWS can then be considered a system of activities that evidences the types of proof.
3. A proof may consist of several steps of reasoning, and reasoning may consist of several proofs. In addition to our considerations from Sect. 2.1, reasoning can consist of a sequence of inferences f_1, f_2, \dots, f_n , where the f_i s are the inferential justifications, both in traditional or instrumental reasoning. But justifications may depend on the very structure of the propositions at stake, such as in a syllogism, in a semantic inference, according to a third-party statement (from the French "énoncé-tiers", by Duval, 1995), or in a deduction or in a discursive inference. Besides the discursive register, these justifications can be expressed through a discursive-graphic reasoning, the use of an artifact or within an encapsulated procedure. In the same way, a proof can be structured by a sequence of connections of epistemic necessity, by adopting deductive, inductive or abductive validation modes. What must be emphasized here is not the structuring aspect of the inferences, but rather the functional purpose of the connection of epistemic necessity in the interaction between a subject and a milieu.¹¹ This hypothesis is that of a possible structuring for reasoning and a functional validation for the one who advances a proof in his mathematical work.

¹¹For more information, see the analysis of student texts and of the editorial organization from Duval (1995), and the analyses of the strategic contexture of proofs in secondary school by Richard (2004a).

4. The ability to prove that results from subject-milieu interactions is affected if the semiotic, instrumental and discursive means of mathematical work are opened or constrained. In Richard et al. (2016), we define the set of the potential subject-milieu interactions related to a type of epistemic necessity as the space of epistemic necessity. This allows us to consider the ability to prove, when interacting with different milieus, in terms of tolerance to possible variations in the coordination of geneses. Like the concept of valence of mathematical work, there is a valence of proof activity in mathematics in a given space of epistemic necessity. Originally, the concept of tolerance is based on the engineering tolerance which focuses on the permissible limit(s) of the potential interactions in a MWS. Thus, the tolerance analysis of a proof is the study of the operating domain of these interactions related to a given space of epistemic necessity. Because a specific milieu conveys mathematical knowledge and processes that are revealed in the use of a tool or of a semiotic production, a variation that involves the geneses into a MWS allows one to test the ability to produce a proof.

The first type of proof we retain is **the discursive-graphic proof** which is represented in the back vertical plane in Fig. 8 (the sem-dis plane in the base form of the diagram, see Fig. 1). This is the most common proof in the mathematical work, operating essentially by the coordination of semiotic and discursive geneses. Traditional demonstrations or those involving well-defined registers of semiotic representation in mathematics are examples of this type of proof. In other words, proofs without words would be the proofs least dependent on the discursive genesis in the mathe-

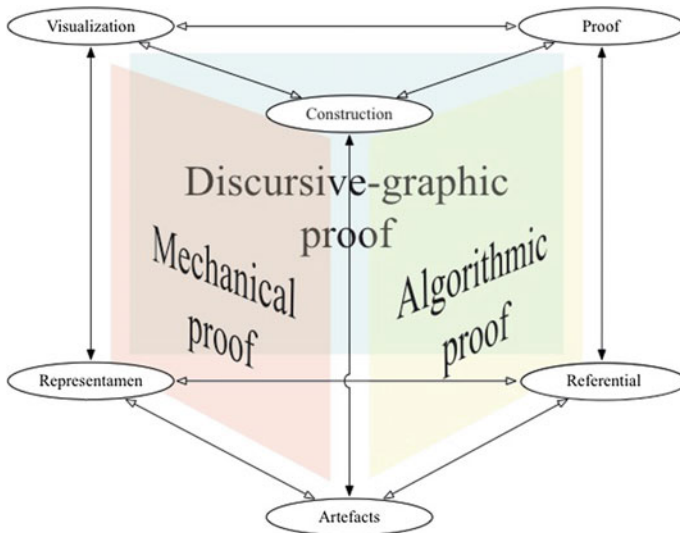


Fig. 8 In the model of MWS, the instrumental proofs appear on the salient vertical planes and the discursive-graphic proof, in the background plane, in order to highlight the coordination of the dominant geneses in mathematical work

mathematical activity, and formal proofs, those least related to the semiotic genesis. As we already mentioned Sect. 2, the question of medium for representation is important. In the absence of dynamism due to the material device (i.e., any form of physical or electronic data carrier), the subject has to animate and control the properties represented. To understand the articulation of connections of epistemic necessity, especially for someone who is not the author of “what to see”, it may be very helpful to give some wording to the reasoning, such as a discursive description or the use of plastic means (colours, textures, grain, lighting, frame, arrows, etc.) to focus on significant elements.

The second type of proof is the **mechanical proof**, which we have already introduced in Sect. 2.4. These proofs, represented in Fig. 8 in the left vertical plane (the sem-ins plane), proceed above all by the coordination of the semiotic and instrumental geneses. In the same way that discursive-graphic proofs can be described as effective, thanks to non-verbal semiotic representations that do not necessarily have to bear the weight of the discursive processing, the mechanical proofs are only functional, and if they “carry with it their own demonstrations”, it’s not according to a purely mathematical rationality. But this advantage has its drawback, because it raises the question of the operative transparency. The joint use of an artifact raises what Rabardel (1995) calls the phenomenal material causality. It concerns the structure of the artifact, its functioning, even its conduct (e.g. systems producing reasoning as the Automated Reasoning Tool (ART) with dynamic geometry software), or at least what is relevant for the subject’s action. To this causality is added the subject’s instrumented action oriented towards the finality of the task. Thus, during a mechanical proof, knowing that it is not the artifact, the method or the implemented reference model that is at stake, but the way they are used, is an example of causality of the instrumented action in proving. Some mechanical proofs can then introduce some ART with a certain independence from what is done with these tools, without compromising the very fact that it is proof.

Some mathematical properties are clearly revealed in an instrumented perspective, while engaging reasonings are activated on different planes within the MWS. Thus, to show that the tangent to a circle is perpendicular to the radius, we can construct a figure-situation at the interface of a dynamic geometry software where the centre of the circle and the end of the radius are defined on grid nodes, the tangent line being able to pivot around the point of tangency (Fig. 9, top left). To see the property, it is necessary to oscillate between satisfactory and unsatisfactory configurations: it is the invariance during these back-and-forth draggings that allow the property to be induced. This is a mechanical proof. Now, if we leave the figure in a position where the perpendicularity is visible, we can certainly see the property according to the visual appearance, but also by counting on the grid nodes (verification of the orthogonality criterion) to notice that we are definitely in a situation of two perpendicular lines. This time it is a discursive-graphic proof that could very well have been done without the computer tool, assuming that constraints are considered when defining the object. The hypothesis of nesting between proof and reasoning is clarified here: a connection of epistemic necessity that concludes similarly (being perpendicular to the radius)

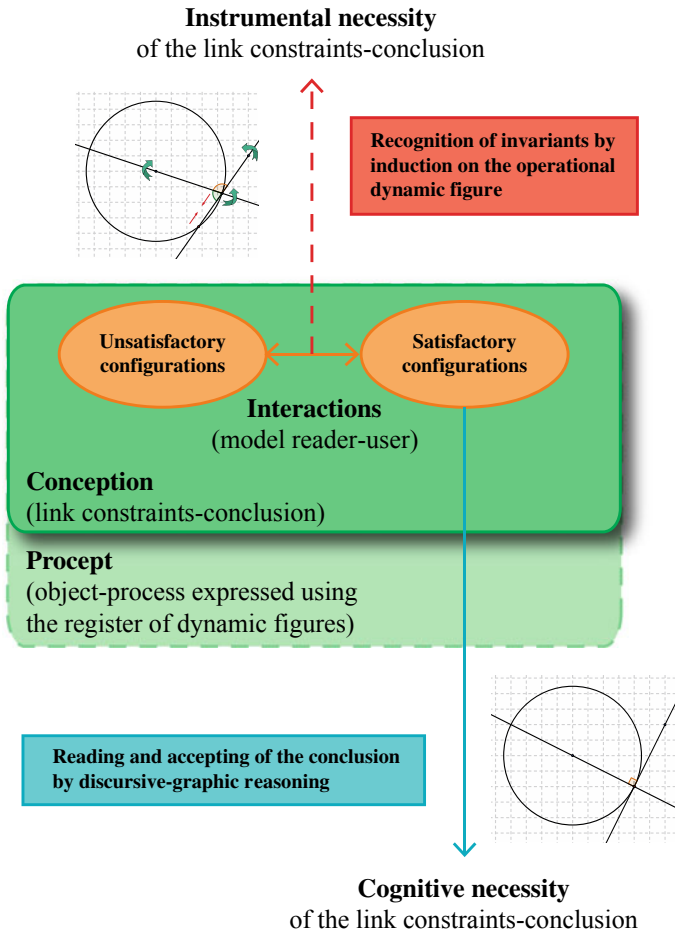


Fig. 9 Two types of epistemic necessity in the subject-milieu interaction (Richard et al., 2016)

can be both instrumental and cognitive, depending on the type of interaction involved such as an action at the interface or a reading.

The third type of proof is the **algorithmic proof**. In the same way that it is easy to consider the instrumental genesis when the execution of an algorithm by a machine is governed by the user, the setting up or the development of an algorithm is naturally associated with the discursive genesis. In an algorithmic proof process, in the meaning we introduced in Sect. 2.6, it is not so much a matter of validating an algorithm in a sort of disconnected computer activity, but rather of considering the algorithmic connected to a mathematical task whose purpose is to prove. Indeed, if we present the algorithmic proofs in the right vertical plane in Fig. 8 (the ins-dis plane), it is in order to highlight the interplay of geneses during the design-in-use of proving, that is the adaptive design of an algorithm while using a machine. However, the algorithm

thinking mode in mathematical work can appear independently of the execution of an algorithm by a machine, as in Trahan's example (Fig. 6), or during an instrumented reasoning that responds to the student's discourse, as does the QED-Turix system (Fig. 7). Like the mechanical proofs, these two examples of algorithmic proofs show that it is possible from time to time to engage in reasonings on separate planes in the MWS, which is the case with the joint support of discursive-graphic steps (proofs or reasonings).

3.2 *Didactic Implications*

If we were not to consider the types of proofs we have just defined, we would still have the essentially discursive proofs, such as traditional demonstrations, the semiotic proofs, such as proofs without words, and the machine-computable proofs, such as proofs tooled by automated reasoning; all these types of proofs would constitute a very good core. But to take into account the complexity of the proving activity, the "other proofs" become necessary. According to the model of mathematical working spaces, there is no executable representamen, hence the need to separate the semiotics from the instrumental and the discursive. For instance, when a geometric figure is represented at the interface of a dynamic geometry software, one who manipulates the figure is also manipulating a semiotic representation system managed partly by a tool, and this evokes the mechanical proof. Also, every time we use a semiotic method to draw conclusions, such as a Venn diagram to determine the probability of a composed event or with the spatial organization of the subtraction of equations to identify the general term of a geometric series, we are exactly in a process of discursive-graphic proofs. In the past, without any software, it was quite difficult to act on a dynamic representamen—there are very few material signs that react by themselves to an action—and the conjecture had to be suggested before the proof because the geneses involved in each process were fundamentally distinct. Such a work could be laborious, even going so far as to blatantly mask the process of discovery. Just remember how we could determine a locus of points using ruler and compass, starting with the production of several drawings. As soon as we had a good enough idea of the locus, we hid with shame our research activity to show discursively the necessity for it, quickly closing our sketchbook, and struggling to prove it properly. However, the significant heuristic moments could have been encapsulated in an instrumental proof worthy of interest, a prelude to the elaboration of reasoning considered as a routine for a class of problems. Whether one needs a discourse or a computer tool to animate the representamen, the opening of the perspective of instrumented proofs brings the heuristics of validation closer in the same crucible that the mathematical experience. In what follows, we focus on the issues concerning automated reasoning tools and intelligent tutorial systems.

Adopting this open attitude is particularly important when it comes to using ART in schools or teacher training. Until now, we have explained that a proof can consist of a series of connections of epistemic necessity, that these connections can be in

deductive, inductive or abductive reasoning modes, but that the questioning must be at the initiative of the student so that the computer-milieu remains a partner. Thus, questioning an oracle at the interface of a dynamic geometry software may seem worrisome if the objective of the task is limited to an immediate production of some deductive reasoning. But this same questioning appears to be very powerful when it comes to support the derivation, discovery and proof of geometry statements (Kovács et al., 2017). With ART tools, we can easily imagine the student's mathematical activity realized in a situation (context, problem or task) where she proactively questions the milieu, in the specific logic of the situation and in the more general logic of the didactic contract that links her to the knowledge at stake. The student then seeks answers adapted to the context, to solve the problem or to accomplish the task, without having to bear all the weight of the logical artillery of the "orthodox presentation of mathematical texts" (see Sect. 2.6). With a dynamic geometry software, for example, we can refer to the many ways that promote investigating geometrical properties of a figure or generalizing some observed/conjectured geometric properties (cf. the nine tools of Kovács, Recio & Vélez, 2018a), and to the combine use of LEGOs and the software to link proving, computation and experimental views in modelling tasks (Kovács, 2018).

In terms of reasoning, ART helps the student in producing valid abductions, as in Pierce's meaning, which brings his experience to the property to be discovered, fostering rigorous creativity during the solving of surprising problems. In fact, researches in didactics of mathematics do not sufficiently address the modelling of physical phenomena using geometrical tools. The same applies to problem-solving: most of the time, we work on problems of proof that rarely go beyond the simple discovery of well-defined properties already known by the teacher or even by the student himself. Undoubtedly, the student's adherence to geometric science requires the development of competence in modelling form, shape and space; but, unfortunately, modelling activity is generally not widely practised in compulsory education—the tasks can be solved using standard representations and definitions, routine procedures, pre-determined heuristics or well-defined methods. While few mathematics teachers in compulsory education or training initiatives seem to be concerned with solving open problems, we believe that the functionalities of ART are particularly useful for designing situations that engage mathematical work on the basis of well-founded judgments, relying on discovery and proof during the quest for problematization-modelling as instrumental proofs do.

The computer-assisted mathematical proof and the interactive proof assistants in schooling are two very different concepts. The QED-Tutrix system is based on this difference. The computer-assisted mathematical proof in an automated proof perspective allows users to check statements and to discover new ones. According to Font et al. (2018), one of the main goals of the research community in automated theorem—proving is to operate efficiently (here meaning fast and focused). Since synthetic approaches are typically slower, most solvers rely on algebraic resolutions. The problem with these systems of automated proving is that the algebraic model makes it possible to say if a geometric statement is true or not, or maybe true (or false) in parts (cf. Kovács, Recio, & Vélez, 2018b), but it does not provide any proof of

that, let alone a proof that a student can master. With the interactive proof assistants, the major problem is often the rigidity of the system that forces the student to work in forward or backward chaining, and this brings us back to Brousseau's paradoxes (2004). It is indeed the modelling of the learning conditions of mathematics in an instrumented perspective by the IT tool that must appear in the design of these tutorial systems. In other words, it is not towards the acceptance of systems to be used as they are, but towards those designed to integrate students, and this very early in the design process. The didactic advantage of an approach like QED-Tutrix is that it allows the student to prove jointly, with verbal and figural statements, as in discursive-graphic proofs, and that it accepts statements in the order suggested by the user. This is an indispensable condition for producing any algorithmic proofs. In fact, if it is no longer the user who manages the structure of the proof, he or she would be condemned to proceed as in a deductive calculation (e.g. by forward or backward chaining). Furthermore, he or she could not integrate the recognition of invariants when dragging, which would unnecessarily hinder the realization of instrumental proofs. If automated reasoning tools can be integrated into the interactive proof assistants, it is mainly to operate with readable proofs.

If we were to pursue research beyond this paper, consideration could be given to the development of a catalogue of instrumental proofs, both proofs that already exist in educational practice and those that should be encouraged in the classroom. If we consider that teachers, trainer and researchers may take different avenues in their proofs, it is obvious that the notion of mathematical work will still be playing a unifying role. We knew that our initial idea about merging thought and activity may be considered as surprising but bright, because we know that they are not similar. But it appears to be useful because it is difficult to distinguish, in algorithmical thinking, when an algorithm "carries with it its own demonstration", independently of any execution by a machine, just as it is difficult to distinguish in interaction with physical reality, which is related to semiotics or to instrumented representation.

4 Conclusion

Two proofs are better than one. "It is safe riding at two anchors".

George Pólya (1973)

The notion of proof has long been a subject of study in mathematics education. Thematic working groups, such as the International Congress on Mathematical Education (ICME) or the Congress of the European Society for Research in Mathematics Education (CERME), and some websites, such as *La lettre de la preuve*,¹² offer a very good inventory of published papers and monographs on the subject. As for the consequences of proof in the teaching and learning of mathematics, synthetic works

¹²This international newsletter on the teaching and learning of mathematical proof, whose current name is simply *Preuve*, is available at <http://www.lettredelapreuve.org> (ISSN 1292-8763).

such as the book *Developing research in mathematics education: twenty years of communication, cooperation and collaboration in Europe* (Dreyfus, Artigue, Potari, Prediger, & Ruthven, 2018) show very well its transversal character in mathematical work. It is precisely by taking advantage of this idea of overlapping of the proof on several mathematical, scientific and thematic fields that we have supported our discourse. The added value of the three types of proofs we have defined in this paper is rooted from the outset in this transversality. But what we wish to highlight above all is that the recognition of these proofs, particularly the instrumental ones, implies an opening on the different ways that proof can get in mathematical work in school. In a culture of this difference, Pólya (1973) said:

When the solution that we have finally obtained is long and involved, we naturally suspect that there is some clearer and less roundabout solution: *Can you derive the result differently? Can you see it at a glance?* Yet even if we have succeeded in finding a satisfactory solution we may still be interested in finding another solution. We desire to convince ourselves of the validity of a theoretical result by two different derivations as we desire to perceive a material object through two different senses. Having found a proof, we wish to find another proof as we wish to touch an object after having seen it. (♣) Two proofs are better than one. “It is safe riding at two anchors”. (pp. 61–62)

If it is obvious that with his heuristics, Pólya was an apostle of the combined use of semiotics and discursive in his reasonings. We believe he would have been very comfortable with the concept of the discursive-graphic proofs. But he rarely addressed the issue of working with instruments, except in a classical perspective as “shall we draw the figures exactly or approximately, with instruments or free-hand?” (p. 105). Even so, knowing the influence that his book¹³ has had among some physicists and computer scientists, we can believe that if he had had a computer on hand easily as today, he would no doubt have found a way to engage us wisely in the instrumental opportunity:

The expert has, perhaps, no more ideas than the inexperienced, but appreciates more what he has and uses it better. *A wise man will make more opportunities than he finds. A wise man will make tools of what comes to hand. A wise man turns chance into good fortune.* Or, possibly, the advantage of the expert is that he is continually on the lookout for opportunities. *Have an eye to the main chance* (p. 224).

Following Pólya’s footsteps and willingly adding the discursive-graphic proofs, we deduce from this that the integration of instrumental proofs in mathematical work in school is a desirable enrichment of the means of validation, and in doing so an approach to the expert’s work. If these “other proofs”, compared to traditional written demonstrations in natural language, appear to involve more obviously heuristic characteristics, then the cross-cutting nature of proving could be formulated through the geneses of mathematical work, their coordination and the fibrations weaving the working space. Furthermore, while the ability to prove at school depends on the students’ resourcefulness, it also depends on the learning opportunities they have been offered, the quality of the milieu chosen by the teacher and the habits conveyed

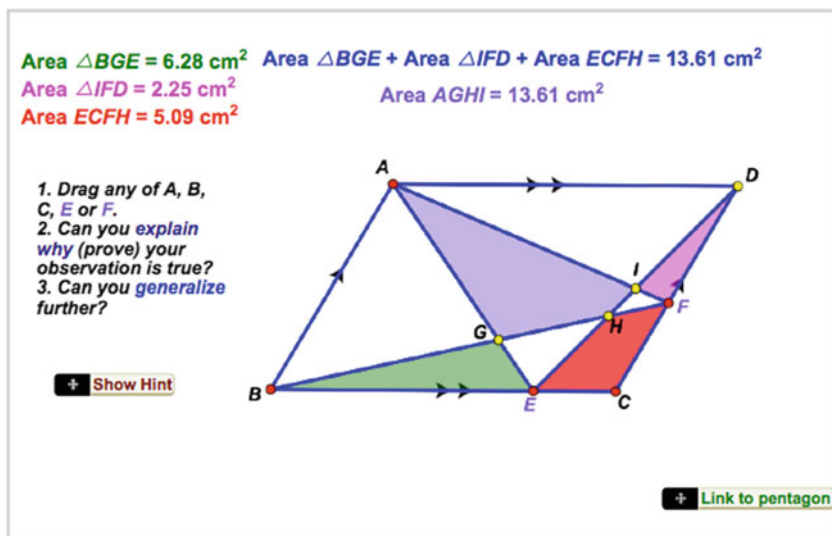
¹³The first English translation of *How to solve it* still dates from 1945.

by the didactic contract, particularly the tolerance to possible variations in the coordination of genes. For the notion of valence of proof activity to become relevant, it must be understood that mathematical proofs intervene in many different tasks and that the epistemic necessity is likely to vary greatly from one task to another. If mechanical proofs and algorithmic proofs serve very well this idea of valence, more exploration has to be done about their usefulness for teaching and research in mathematics didactics. If two proofs are better than one, now imagine three proofs!

Acknowledgements We wish sincerely to thank Prof. Annette Braconne-Michoux for her devoted and far-sighted work of linguistic review.

Appendix 1: Area Partition Activity

In an activity of dynamic geometry, de Villiers (2018)¹⁴ proposes the study of situations which allow us to show that “to the working mathematician, proof is not merely a means of verifying an already discovered result, but often also a means of exploring, analyzing, discovering and inventing new results”. It begins by proposing an interactive version of the proof without words presented in Sect. 2.4, automatically providing the measures on both sides of equality:



This is a proof that is both mechanical and discursive-graphic, as is our considerations around Fig. 9. At the interface of the situation, the author adds a help button that participates in the devolution of the problem (hint) and a generalization button

¹⁴From <http://dynamicmathematicslearning.com/area-parallelogram-partition-richard-theorem.html>.

that allows the user to move to the next situation (pentagon). In fact, the user can always move to a more general situation (pentagon or to another $(k+1)$ -gon, until an octagon) without even having solved the previous problem (parallelogram or k -gon):

Area $\triangle BIJ = 3.14 \text{ cm}^2$

Area $\triangle KLH = 2.98 \text{ cm}^2$

Area $MNEO = 4.46 \text{ cm}^2$

Area $AIMK = 8.05 \text{ cm}^2$

Area $\triangle JPN = 1.61 \text{ cm}^2$

Area $\triangle OQL = 0.92 \text{ cm}^2$

Area $\triangle BIJ + \text{Area } \triangle KLH + \text{Area } MNEO = 10.58 \text{ cm}^2$

Area $AIMK + \text{Area } \triangle JPN + \text{Area } \triangle OQL = 10.58 \text{ cm}^2$

1. Drag any of the **red** vertices or points.
2. Can you explain **why** (prove) your observation is true?
3. Can you **generalize** further?

+ Show Hint

+ Link to heptagon

ABCDEFGH is an octagon with opposite sides parallel

This dynamicity of the activity makes it possible to identify invariants in the equality of areas of figures and to reinvest them in a set of proof problems:

Carefully *reflect* on your proof, and consider how this same proof can also apply to a certain type of pentagon, hexagon, etc. Make *generalizations* and check your generalized conjectures by clicking on the *Link* buttons on the right to go to pentagons, hexagons, etc. with a similar area partition property. (extract from the activity instructions)

It would then be a particularly rich activity of proof which also makes it possible to engage an algorithmic proof.

Appendix 2: Transcription of Text from Dense Figures

This is the sequential transcription of Trahan’s approach structured in Fig. 6.

Step 1

Theorem (not surprising)

If you cut a polygon into a finite number of pieces to reform another, then both polygons will have the same area.

Theorem (more surprising)

Should any two polygons have the same area, it is possible to cut the first into a finite number of pieces to then form the second.

Step 2**Propositions** (program)

1. Any polygon can be cut into a finite number of triangles.
2. Any triangle can be cut to form a rectangle.
3. Any rectangle can be cut to form a square.
4. Any pair of squares can be cut to form a square.

Resolution (breakdown by case)

1. Demonstration in two cases: convex polygon and concave polygon.
2. Check the four corners of the rectangle and check the «joints» (alignment of points).
3. In a rectangle L by ℓ : case $\ell < L \leq 4\ell$, checking of the angles, of the «joints» then sides; case $L > 4\ell$, check also lengths and sides.
4. Verification of angles, “joints” then sides.

Conclusion

Any polygon can be cut into a finite number of pieces to then form a square.

Step 3**Deriving proposition**

If two polygons of the same surface can be cut into a finite number of pieces to each form a square, then there is a split of the first polygon to form the second.

Solution

Superposition of the two cuttings.

Can we do better?

Yes, the demonstration makes it possible to obtain a division between two polygons, but this is not optimal.

Verification with interesting cuttings (demonstrated in other research work)

- Square in equilateral triangle.
- Square in nonagon.
- Some cutting of the pentagon.

Step 4

New situations (to pose and solve)

- Generalization: curved surface?
- Generalizations in 3 dimensions
- (...)

Hilbert's 3rd problem: given two polyhedra of equal volume, can we cut the first polyhedron into polyhedra and bring them together to form the second polyhedron?

- No: Dehn found an invariant (Dehn invariant) that is preserved during a cut; the cube and the tetrahedron do not have the same invariant.
- Some possible cuttings...

References

- Alsina, C., & Nelsen, R. (2006). *Math made visual. Creating images for understanding mathematics*. Washington: The Mathematical Association of America.
- Appel, K., & Haken, W. (1977). The solution of the four-color-map problem. *Scientific American*, 237(4), 108–121. <https://doi.org/10.1038/scientificamerican1077-108>.
- Arzarello, F. (2006). Semiosis as a multimodal process. *Revista latinoamericana de investigación en matemática educativa*, 9(1), 267–299.
- Balacheff, N. (1987). Processus de preuve et situations de validation. *Educational Studies in Mathematics*, 18(2), 147–176.
- Balacheff, N., & Margolinas, C. (2005). Ckç, modèle de connaissances pour le calcul de situations didactiques. In A. Mercier & C. Margolinas (Eds.), *Balises pour la didactique des mathématiques* (pp. 75–106). Grenoble: La Pensée Sauvage.
- Barbin, É., Duval, R., Giorgiutti, I., Houdebine, J., & Laborde, C. (2001). *Produire et lire des textes de démonstration*. Paris: Éditions Ellipses.
- Bartle, R. G., & Sherbert, D. R. (2011). *Introduction to real analysis* (4th ed.). Wiley & Sons.
- Bartolini Bussi, M.G., & Maschietto, M. (2005). *Macchine Matematiche: Dalla storia alla scuola*. Springer.
- Blum, W., & Leiß, D. (2007). How do students and teachers deal with modelling problems? In C. Haines, P. Galbraith, W. Blum, & S. Khan (Eds.), *Mathematical modelling (ICTMA12): Education, engineering and economics* (pp. 222–231). Chichester, UK: Horwood Publishing. <https://doi.org/10.1533/9780857099419.5.221>.
- Borromeo-Ferri, R. (2006). Theoretical and empirical differentiations of phases in the modelling process. *ZDM—The International Journal on Mathematics Education*, 38(2), 86–95. <https://doi.org/10.1007/BF02655883>.
- Boutin, F. (2017). *Le cahier Transmath d'algorithmique Cycle 4 (5e/4e/3e)*. Nathan.
- Brousseau, G. (1998). *Théorie des situations didactiques*. Grenoble: La Pensée Sauvage.
- Brousseau, G. (2004). Introduction to the study of the teaching of reasoning and proof: Paradoxes. *La lettre de la preuve—International newsletter on the teaching and learning of mathematical proof*. Retrieved April 17, 2018, from <http://www.lettredelapreuve.org/OldPreuve/Newsletter/04Ete/04EteThemeUK.html>.
- Bryant, J., & Sangwin, C. (2008). *How round is your circle? Where engineering and mathematics meet*. Princeton University Press.
- Castelnuovo, E., & Barra, M. (1976). *Matematica nella realtà*. Torino: Boringhieri.
- Chevallard, Y. (1992). *La transposition didactique*. Grenoble: La Pensée Sauvage.
- Clairaut, A.C. (1741). *Éléments de Géométrie*. Paris: David Fils.
- Clairaut, A.C. (1881). *Elements of geometry*. Traduction du texte de l'édition de 1830 publié à Paris. London: C. Kegan Paul & Co.
- Cormack, L.B. (2017). Introduction: Practical mathematics, practical mathematicians, and the case for transforming the study of nature. In L.B. Cormack, S.A. Walton, & J.A. Schuster (Eds.), *Mathematical practitioners and the transformation of natural knowledge in Early Modern Europe* (pp. 1–8), *Studies in History and Philosophy of Science* 45. https://doi.org/10.1007/978-3-319-49430-2_1.

- Coutat, S., Laborde, C., & Richard, P. R. (2016). L'apprentissage instrumenté de propriétés en géométrie: propédeutique à l'acquisition d'une compétence de démonstration. *Educational Studies in Mathematics*, 1–27. <https://doi.org/10.1007/s10649-016-9684-9>.
- Derouet, C. (2016). *La fonction de densité au carrefour entre probabilités et analyse en terminale S: Étude de la conception et de la mise en oeuvre de tâches d'introduction articulant lois à densité et calcul intégral*. Thèse de doctorat, Sorbonne Paris Cité. <https://tel.archives-ouvertes.fr/tel-01431913/document>.
- Dreyfus, T., Artigue, M., Potari, D., Prediger, S., & Ruthven, K. (2018). *Developing research in mathematics education—Twenty years of communication, cooperation and collaboration in Europe*. Oxon, UK: Routledge (ISBN: 978-1-138-08027-0).
- Dupré, S. (2006). Visualization in Renaissance optics: The function of geometrical diagrams and pictures in the transmission of practical knowledge. In S. Kusukawa & I. Maclean (Eds.), *Transmitting knowledge: Words, images, and instruments in early modern Europe* (pp. 11–39). Oxford University Press.
- Dupré, S. (2017). The making of practical optics: Mathematica practitioners' appropriation of optical knowledge between theory and practice. In L.B. Cormack, S.A. Walton, & J.A. Schuster (Eds.), *Mathematical practitioners and the transformation of natural knowledge in Early Modern Europe* (pp. 131–148), *Studies in History and Philosophy of Science* 45. https://doi.org/10.1007/978-3-319-49430-2_7.
- Duval, R. (1995). *Sémiosis et pensée humaine: registre sémiotique et apprentissages intellectuels*. Berne: Peter Lang.
- Font, L., Richard, P.R., & Gagnon, M. (2018). Improving QED-Tutrix by automating the generation of proofs. In P. Quaresma & W. Neuper (Eds.), *Proceedings 6th International Workshop on Theorem Proving Components for Educational Software, (ThEdu'17), Electronic Proceedings in Theoretical Computer Science* (Vol. 267, pp. 38–58). <https://doi.org/10.4204/eptcs.267.3>.
- Gray, E., & Tall, D. (1994). Duality, ambiguity and flexibility: A perceptual view of simple arithmetic. *The Journal for Research in Mathematics Education*, 26(2), 115–141.
- Haspekian, M. (2011). The co-construction of a mathematical and a didactical instrument. In M. Pytlak, T. Rowland, & E. Swoboda (Eds.), *Proceedings of the Seventh Congress of the European Society for Research in Mathematics Education (CERME 7)* (pp. 2298–2307). Rzeszów, Poland.
- Hoffmann, D.L., Standish, C.D., García-Diez, M., Pettitt, P.B., Milton, J.A., Zilhão, J., ... Pike, W.G. (2018). U-Th dating of carbonate crusts reveals Neandertal origin of Iberian cave art. *Science*, 359(6378), 912–915. <https://doi.org/10.1126/science.aap7778>.
- Keller, A.G. (2017). Machines as mathematical instruments. In L.B. Cormack, S.A. Walton, & J.A. Schuster (Eds.), *Mathematical practitioners and the transformation of natural knowledge in Early Modern Europe* (pp. 115–127), *Studies in History and Philosophy of Science* 45. https://doi.org/10.1007/978-3-319-49430-2_6.
- Keller, O. (2004). *Aux origines de la géométrie: le Paléolithique, le monde des chasseurs-cueilleurs*. Paris: Vuibert.
- Kovács, Z. (2018). Motion with LEGOs and dynamic geometry. *Virginia Mathematic Teacher*, 44(2), 43–48.
- Kovács, Z., Recio, T., & Vélez, M. P. (2018a). Using automated reasoning tools in GeoGebra in the teaching and learning of proving in geometry. *International Journal of Technology in Mathematics Education*, 25(2), 33–50. https://doi.org/10.1564/tme_v25.2.03.
- Kovács, Z., Recio, T., & Vélez, M.P. (2018b). Detecting truth, just on parts, in automated reasoning in geometry. In F. Botana, F. Gago, & M. Ladra (Eds.), *Proceeding of the 24th Conference of Applications of Computer Algebra* (pp. 32–35).
- Kovács, Z., Richard, P.R., Recio, T., & Vélez, M.P. (2017). GeoGebra automated reasoning tools: A tutorial with examples. In G. Aldon, & J. Trgalova (Eds.), *Proceedings of the 13th International Conference on Technology in Mathematics Teaching* (pp. 400–404). <https://hal.archives-ouvertes.fr/hal-01632970>.
- Krieger, M.H. (2004). Some of what mathematicians do. *Notices of the AMS*, 51(10), 1226–1230.

- Kuzniak, A., & Richard, P.R. (2014). Espaces de travail mathématique. Point de vues et perspectives. *Revista latinoamericana de investigación en matemática educativa* 17.4(I), 5–40.
- Kuzniak, A., Richard P.R., & Michael-Chrysanthou, P. (2018). Chapter 1. From geometrical thinking to geometrical working competencies. In T. Dreyfus, M. Artigue, D. Potari, S. Prediger, & K. Ruthven (Eds.), *Developing research in mathematics education—Twenty years of communication, cooperation and collaboration in Europe*. New Perspectives on Research in Mathematics Education series (Vol. 1, pp. 8–22). Oxon, UK: Routledge (ISBN: 978-1-138-08027-0).
- Kuzniak, A., Tanguay, D., & Elia, I. (2016). Mathematical Working Spaces in schooling: An introduction. *ZDM—The International Journal on Mathematics Education*, 48(6), 721–737. <https://doi.org/10.1007/s11858-016-0812-x>.
- Lagrange, J. B., Recio, T., Richard, P. R., & Vivier, L. (2017). Synthesis of topic 2—Specificities of tools and signs in the mathematical work. In I. Gómez-Chacón, A. Kuzniak, K. Nikolantonakis, P. R. Richard, & L. Vivier (Eds.), *Actes du 5e symposium Espaces de Travail Mathématique (ETM 5)* (pp. 207–239). Greece: University of Western Macedonia.
- Laval, D. (2018). *L'algorithmique au lycée entre développement de savoirs spécifiques et usage dans différents domaines mathématiques*. Thèse de doctorat, Sorbonne Paris Cité.
- Leduc, N. (2016). *QED-tutrix: système tutoriel intelligent pour l'accompagnement des élèves en situation de résolution de problèmes de démonstration en géométrie plane*. Thèse de doctorat, École Polytechnique de Montréal. <https://publications.polymtl.ca/2450/>.
- Leduc, N., Tessier-Baillargeon, M., Corbeil, J. P., Richard, P. R., & Gagnon, M. (2017). Étude prospective d'un système tutoriel à l'aide du modèle des espaces de travail mathématique. In I. Gómez-Chacón, A. Kuzniak, K. Nikolantonakis, P. R. Richard, & L. Vivier (Eds.), *Actes du 5e symposium Espaces de Travail Mathématique (ETM 5)* (pp. 281–295). Greece: University of Western Macedonia.
- Modeste, S., Gravier, S., & Ouvrier-Buffer, C. (2010). Algorithmique et apprentissage de la preuve. *Repères IREM*, 79, 51–72.
- Moutet, L. (2016). *Diagrammes et théorie de la relativité restreinte: Une ingénierie didactique*. Thèse de doctorat, Sorbonne Paris Cité. <https://hal.archives-ouvertes.fr/tel-01611332/document>.
- Netz, W., & Noel, W. (2008). *Le codex d'Archimède—Les secrets du manuscrit le plus célèbre de la science*. Paris: JC Lattès.
- Polya, G. (1973). *How to solve it. A new aspect of mathematical method*. New Jersey: Princeton University Press.
- Rabardel, P. (1995). *Les hommes et les technologies, une approche cognitive des instruments contemporains*. Paris: Armand Colin.
- Recio, T., Richard, P. R., & Vivier, L. (2015). Synthesis of topic 2—Specific features of tools and signs in the mathematical work. In I. Gómez-Chacón, J. Escribano, A. Kuzniak, & P. R. Richard (Eds.), *Actes du 4e symposium Espaces de Travail Mathématique (ETM 4)* (pp. 197–226). Spain: Universidad Complutense de Madrid.
- Richard, P. R. (2003). Proof without words: Equal areas in a partition of a parallelogram. *Mathematics Magazine*, 76(5), 348.
- Richard, P. R. (2004a). *Raisonnement et stratégies de preuve dans l'enseignement des mathématiques*. Berne: Peter Lang.
- Richard, P. R. (2004b). L'inférence figurale: Un pas de raisonnement discursivo-graphique. *Educational Studies in Mathematics*, 57(2), 229–263.
- Richard, P.R., Gagnon, M., & Fortuny, J.M. (2018). Chapter 20. Connectedness of problems and impasse resolution in the solving process in geometry: A major educational challenge. In P. Herbst, U.H. Cheah, P.R. Richard, & K. Jones (Eds.), *International perspectives on the teaching and learning of geometry in secondary schools* (19 pp.). Cham, Switzerland: Springer (ISBN: 978-3-319-77475-6).
- Richard, P. R., Oller, A. M., & Meavilla, V. (2016). The concept of proof in the light of mathematical work. *ZDM—The International Journal on Mathematics Education*, 48(6), 843–859. <https://doi.org/10.1007/s11858-016-0805-9>.
- Sullivan, M. C. (2000). Knot factoring. *The American Mathematical Monthly*, 107(4), 297–315.

- Tanguay, D., & Geeraerts, L. (2014). Conjecture, postulats et vérifications expérimentales dans le paradigme du géomètre physicien: comment intégrer le travail avec des logiciels de géométrie dynamique? *Revista latinoamericana de investigación en matemática educativa* 17.4(II), 287–302.
- Tanguay, D., & Venant, F. (2016). The semiotic and conceptual genesis of angle. *ZDM—The International Journal on Mathematics Education*, 48(6), 875–894. <https://doi.org/10.1007/s11858-016-0789-5>.
- Tanguay, D., Kuzniak, A., & Gagatsis, A. (2015). Synthesis of topic 1—The mathematical work and mathematical working spaces. In I. Gómez-Chacón, J. Escibano, A. Kuzniak, & P. R. Richard (Eds.), *Actes du 4e symposium Espaces de Travail Mathématique (ETM 4)* (pp. 20–38). Spain: Universidad Complutense de Madrid.
- Tessier-Baillargeon, M. (2015). *GeoGebraTUTOR: Développement d'un système tutoriel autonome pour l'accompagnement d'élèves en situation de résolution de problèmes de démonstration en géométrie plane et genèse d'un espace de travail géométrique idoine*. Thèse de doctorat, Université de Montréal. https://papyrus.bib.umontreal.ca/xmlui/bitstream/handle/1866/15902/Tessier-Baillargeon_Michele_2016_these.pdf.
- Tessier-Baillargeon, M., Leduc, N., Richard, P.R., Gagnon, M. (2017). Étude comparative de systèmes tutoriels pour l'exercice de la démonstration en géométrie. *Annales de didactique et de sciences cognitives* (Vol. 22, pp. 91–117). IREM de Strasbourg: Université Louis Pasteur.
- Trahan, A. (2016, October). *Casse-tête de polygones*. Workshop session presented at the 60th Congress of the Association Mathématique du Québec under the theme *Des Mathématiques Surprenantes*, Québec, Québec.
- Trouche, L. (2005). Construction et conduite des instruments dans les apprentissages mathématiques: nécessité des orchestrations. *Recherche en Didactique des Mathématiques*, 25, 91–138.
- Venant, F. (2018). Programmer les mathématiques: la pensée informatique à l'école primaire. *Bulletin AMQ LVIII*(3), 57–70.
- Villiers, M. (1993). El Papel y la Función de la demostración en matemáticas. *Epsilon*, 26, 15–30.
- Vitrac, B. (1992). À propos de la chronologie des œuvres d'Archimède. In J.Y. Guillaumin (Ed.), *Mathématiques dans l'Antiquité* (pp. 59–93). Publications de l'Université de Saint-Etienne.

The Contribution of Information and Communication Technology to the Teaching of Proof



Maria Alessandra Mariotti

1 Introduction

Research on proof and proving in mathematics education has been carried out for a long time. It began with criticizing old models of teaching proof (Herbst, 2002) for their inefficiency in understanding the role of proof in mathematics and the development of students' skills in producing conjectures and constructing proofs. In many countries the reaction to this criticism led to abandoning the practice of proof at school, sometimes getting rid of theorems, and in general reducing the importance of proving in secondary school curricula. Nevertheless, studies in the field of teaching and learning proof have opened new perspectives and generated a lively stream of research. On the one hand, research has focused on epistemological reflections about the relationship between proof and proving in mathematical practice and education (e.g., Hanna, 1989; Hanna & Janke, 2007; Duval, 2007; Balacheff, 2008; Hanna, Jahnke, & Pulte, 2009). On the other, it has focused on student practices related to proof and proving (e.g., Harel & Sawder, 1998) from both a cognitive and didactic point of view. An overview and detailed discussion of these issues can be found in Mariotti (2006), Reid and Knipping (2010), Hanna and De Villiers (2012), and Stylianides, Bieda and Morselli (2016).¹

Most of the new proposals are based on short-term experiments limited to a very specific proof task (see, for instance, Miyazaki, Fujita, & Jones, 2015). Only a few, such as the examples found in Boero (2007), have been rooted in long-term design-based experiments. What I present here are some findings from teaching experiments that investigate the use of ICT to introduce secondary school students to proof and proving and, more generally, to developing specific mathematical meanings related

¹A rich source of references can be found at <http://lettredelapreuve.org>.

M. A. Mariotti (✉)
University of Siena, Siena, Italy
e-mail: mariotti21@unisi.it

to proof and proving. Some of these experiments were carried out by the author, sometimes in collaboration with other colleagues. Some of them lasted for years and involved different classes for a whole academic year while others were more circumscribed, involving individuals or pairs of students. The interviews aimed at observing students' behaviours in great detail. In elaborating on this wide base of findings, my objective is not to provide an overview of the rich research on proof in relation to technological settings, but to illustrate the potentials of a specific kind of software environment for fostering a sense of proof and, more widely, a theoretical perspective. Some of the results presented here have been published elsewhere; for instance, in Mariotti (2012, 2014) and Baccaglioni-Frank, Antonini, Leung and Mariotti (2018). In the following, I offer a synthesis of contributions presented in the past years based on the unifying lens of the theory of semiotic mediation (TSM) and, specifically, the construct of the semiotic potential of an artefact.

In the first section, I introduce the educational and epistemological perspectives that shape my discussion. I use the theoretical construct of semiotic potential in order to explain why and how certain designed activities in a Dynamic Geometry Environment² (DGE) may contribute to the construction of specific meanings constituting the mathematical meaning of proof. After introducing a specific characterization of a theorem that sheds light on the mathematical meaning of proof I start to discuss the potentialities offered by a DGE. I firstly elaborate on construction problems discussing how the solution of such problems may be related to the key meanings of axioms and theorems. In the following sections, I discuss how open construction problems and the related conjecturing process may give sense to the notion of conditional statement when carried out in a DGE, offering a context to relate spontaneous arguments to mathematical proof. The last issue concerns the potential offered by exploring impossible figures in a DGE with respect to indirect proof.

2 Theoretical Background

In this section, I discuss the theoretical foundations of my research on the use of ICT to foster student engagement with mathematical proof. These foundations include an educational perspective based on the theory of semiotic mediation, an epistemological perspective based on key mathematical meanings related to proof, and the notion of mathematical theorem.

²Following Sinclair and Robutti (2012), I use the term “dynamic geometry environment.” As the authors write, since at least 1996, this term has been used over *dynamic geometry software* “to underscore the fact that we are dealing with microworlds (including pre-existing sketches and designed tasks) and not just a software program.” (p. 571).

2.1 Educational Perspective

The Theory of Semiotic Mediation (TSM) (Bartolini Bussi & Mariotti, 2008; Mariotti, 2009) combines a semiotic and an educational perspective. It elaborates on the Vygotskian notion of semiotic mediation, which considers the role of human mediation in the teaching-learning process as crucial. Starting from the key notion of artefact,³ TSM interprets the teaching and learning process through a semiotic lens. It focusses on students' production of signs and the evolution of these signs from personal meanings emerging from the use of an artefact to the mathematical meanings that are the goal of teaching. A basic assumption is that personal meanings that emerge in accomplishing a task may be related to specific mathematical meanings, but also that such a relationship cannot be taken for granted. On the contrary, the *intentional* intervention of the teacher is needed to promote students' conscious construction of this relationship in social interaction. According to a Vygotskian approach, TSM sees both the individual production of signs and their collective elaboration within social activities as fundamental, and in particular within mathematical discussions (Bartolini & Mariotti, 2008).

In the past years, several long-term teaching experiments have been conducted to check, refine, and elaborate our assumptions. Different artefacts were involved, either concrete or digital, as well different school levels (see Bartolini Bussi, 1996; Mariotti, 2007, 2010). The theoretical framework of TSM originated and developed around two key elements: the notion of the semiotic potential of an artefact and the notion of a didactic cycle (Bartolini Bussi & Mariotti, 2008). Here I focus on the notion of @@@ semiotic potential which will be used in the following discussion.

When an artefact is used to accomplish a task, it may happen that an expert—a mathematician—recognizes the echo of specific mathematical notions. For instance, the use of an abacus may immediately bring to mind the mathematical notion of positional notation and the polynomial notation of numbers. As I discuss later, drawing a figure in a dynamic geometry system may evoke the classic notion of geometric construction by ruler and compass. However, if the user is not an expert, the meanings emerging from use of the artefact may not be immediately and consciously related to mathematical meanings. Instead, they are related to the specific context and the specific individual. They are 'personal meanings.'

To express the double relation linking the artefact and its use with, on the one hand, possible personal meanings and, on the other, with mathematical meanings, TSM introduces the notion of @@@semiotic potential (Bartolini Bussi & Mariotti, 2008, p. 754). This notion also expresses the double use of an artefact in an educational context. The artefact is used by the students to accomplish a task, but is simultaneously used by the teacher to exploit its semiotic potential and foster students' mathematical meanings. Consequently, analysing the semiotic potential of an artefact is at the core of designing and teaching a sequence of lessons. Such an analysis involves both cognitive and epistemological aspects. The former identifies

³The term artefact refers to any generic product of human culture purposefully designed to act or interact in a human setting.

meanings that can emerge in accomplishing a given task while the latter identifies the possible mathematical meanings evoked. In this chapter, I use the notion of semiotic potential to illustrate and discuss the educational potential of a DGE with respect to mathematical meanings related to proof.

2.2 *Epistemological Perspective*

Proof is one of the key elements of mathematics. It is the product of a process of validation that allows the inclusion of any new statement in a specific theory given that such a statement can be logically derived from the set of axioms previously assumed. Such a formal perspective (Arzarello, 2007) makes a proof independent of any interpretation and factual verification of the statements involved. In this respect, the specificity of proof contrasts with argumentation and any action or process of reasoning aimed at convincing others (or oneself) that something is true or false.

As Duval clearly stated (2007), argumentation and proof must be distinguished. Further, a cognitive gap might exist between the two processes in spite of their possible contiguity. The epistemological gap concerns the unbridgeable distance between the semantic level, where the interpretation of any statement finds reasons for its acceptability, and the theoretical level where the theoretical validity of a statement must be stated in accordance with laws of logic within a *hypothetical–deductive* approach. The cognitive gap concerns the distinction between the main function of argumentation—convincing oneself and others that a statement is true—and the main function of mathematical proof—logical validation of a statement within a specific theory.

According to this analysis, the educational challenge concerns the possibility of mistaking the two processes. In principle, a mathematical proof should not refer to any interpretation of the statement involved. However, it is not realistic for mathematicians or students to think that such interpretation does not play a crucial role in both producing and/or accepting a proof. Indeed, it is the interpretation given to the statements that determines the final epistemic value attributed to the statement that has been proved. Moreover, it is on the semantic plane that the explicative function any proof is expected to provide is based (Hanna, 1989). The semantic plane develops understanding of ‘why’ what has been proved is true (Dreyfus & Hadas, 1996).

In summary, the didactic question of proof requires resolving the potential conflict between the two main functions of proof: theoretically validating, and explaining why. This means developing a teaching intervention that enables students to develop a coherent intertwining of argumentation and proof, though preserving their specificity. In this perspective, in the following I elaborate a bit more on the notion of proof.

The term proof is often used, both in the current literature and in textbooks, without any clear reference to the other key elements involved. As said above, distinguishing argumentation from mathematical proof is based on differentiating their aims: on the one hand, stating the epistemic value of a statement and, on the other, validating a statement with a theory. Though these two aims may overlap, no clear idea of

mathematical proof is possible without explicitly linking it to the idea of *theory*—both the theoretical system defined by the axioms, definitions, and already proven theorems and the meta-theoretical system of the inference rules stating what is meant by *logically derived*.

Very often, when discussing the issue of proof, we take the perspective of mathematics experts and leave reference to a statement and a theory implicit. However, if we take the students' point of view, we realize that neither of these perspectives can be taken for granted and the complexity of the notion of theory cannot be underestimated. On the one hand, meanings must be developed in relation to the status and role of the different statements involved in a proof. That is, the mathematical meaning of terms like theory, axioms, definitions, and theorems must be developed. On the other hand, consciousness of the means of supporting any single step of a proof—the specific 'logical means' that can be used to validate a new statement—must also be developed. The centrality of the latter has been clearly pointed out by Sierpinska (2005):

Theoretical thinking asks not only, Is this statement true? but also What is the validity of our methods of verifying that it is true? Thus theoretical thinking always takes a distance towards its own results. [...] theoretical thinking is thinking where thought and its object belong to distinct planes of action. (pp. 121–23)

In the school context, the complexity of this meta-theoretical level seems to be ignored. It is commonly taken for granted that students' ways of reasoning are spontaneously adaptable to the sophisticated functioning of a theoretical system. Therefore, not much is said about it, and inference rules in particular and their functioning in the development of a theory are rarely made explicit.⁴

In fact, at least two aspects at the meta-level should be made explicit and discussed in the classroom: (1) the acceptability of some specific inference means, and (2) the fact that, except for those explicitly shared, no other inference means are acceptable. If meta-theoretical aspects remain implicit, students have no control of their arguments. Control remains totally in the hands of the teacher, with the consequence that students feel confusion, uncertainty, and lack of understanding. Awareness of a reference theory as a system of shared principles and inference rules is needed if we are to speak of proof in a mathematical sense. Indeed, "what characterises a Mathematical Theorem is the system of statement, proof, and theory" (Mariotti et al. 1997, p. 182).

Developing the interrelated meanings of the three components of the notion of Mathematical Theorem therefore becomes a crucial pedagogical objective. In the following, I discuss how teaching can be designed to address this objective.

⁴An exception is that of mathematical induction. But mathematical induction is very rarely presented in comparison to other modalities of proving, which are commonly considered natural and spontaneous ways of reasoning.

3 Introducing Students to Theorems

For some time different research studies have highlighted the potentials and pitfalls (de Villiers, 1998) of DGEs in offering powerful resources for introducing students to proof. As Hadas, Hershkowitz and Schwarz (2000) have pointed out:

[The] findings concerning the failure to teach proofs, the recognition of the multiple aspects of proving, and the existence of DG tools lead naturally to the design of investigative situations in which DG tools may foster these multiple aspects. (p. 130)

In the following, I describe the potentials of a DGE in relation to geometrical construction and situate it within the theoretical framework of TSM; that is, in terms of semiotic potential.

3.1 Geometrical Construction in a DGE

Let us start from the relationship, immediately evoked in the mind of any mathematician, between drawing a figure in a DGE and the mathematical meaning of geometrical construction; that is, drawing a figure by ruler and compass. In terms of TSM, such a relationship can be articulated through both an epistemological and cognitive analysis, and leads to outlining the semiotic potential of the artefact ‘DGE’ with respect to the meaning of Theorem.

Euclidean geometry is traditionally referred to as ‘ruler-and-compass geometry.’ However, despite referring to a concrete objective—e.g., producing a graphic trace on a sheet of paper or other surface—a geometrical construction has a pure theoretical nature. Solving a construction problem corresponds to proving a theorem that validates the construction procedure (Mariotti, 2007).

The use of ruler and compass generates a set of axioms defining the theoretical system of Euclid’s Elements. To appreciate the key role played by construction problems in Euclidean Geometry, it suffices to remember that the very first proposition of the first book of the Elements deals with the construction of an equilateral triangle, and that the solution to the long puzzling problem of trisecting an angle was definitely proved impossible to solve by ruler and compass. The constructability or non-constructability of a figure has been a central issue in mathematics (Arzarello et al., 2012). Although, as classic research studies have shown (Schoenfeld, 1985), the theoretical meaning of geometrical construction is complex and difficult to grasp, the centrality of its role in the history of geometry and the revival triggered by the advent of DGE make it worthy of consideration.

On the one hand, the use of virtual tools simulates the concrete use of traditional tools like the ruler and compass. On the other, the digital architecture of a DGE embedding the theoretical framework of Euclidean Geometry (Laborde & Sträßer, 1990) enables the user to implement the logical relationships between the geometrical properties constructed by the tools and the geometrical properties that are their consequences. Moreover, any DGE offers a dragging modality which represents the

core of the technological environment. The dragging modality allows the user to move any constructed figure after clicking and dragging one of its basic points. After a selected point has been dragged, the figure on the screen is redrawn and recalculated from the subsequent new positions, but maintains all the properties defined by the constructing procedure. As a consequence, the stability of dragging constitutes the standard test of correctness for any drawn figure. Thus, a solution is acceptable if and only if the figure on the screen is stable under the dragging test. Because any DGE embodies a system of relationships consistent with the broad system of a geometrical theory, solving construction problems in a DGE means not only accepting all the facilities of the software, but also accepting a logic system within which to make sense of the geometrical phenomena that occur in that environment.

A dynamic figure behaves according to its intrinsic logic: its elements are related by the hierarchical relationships stated by the constructing procedure. Such a hierarchy corresponds to a relationship of logical dependence among the properties in the sense that the final figure will show not only the constructed properties, but also all the properties that can be derived from them according to Euclidean Geometry. Specific tools on the DGE menu correspond to a set of theoretical construction tools in Euclidean Geometry (Laborde & Laborde, 1991). This makes it possible to state a correspondence between the control of dragging (dragging test) and validation by theorems (e.g., validation by mathematical proof within Euclidean Geometry theory).⁵

3.2 The Semiotic Potential of DGE Construction Tools

Interpreting the previous analysis in terms of semiotic potential we can recognize a double relationship between some tools of a DGE and, on the one hand, meanings emerging from their use in solving a construction task and, on the other, specific mathematical meanings related to the notion of Theorem. Specific construction tools can be related to a virtual dynamic drawing representing a geometrical figure whose acceptability as a solution of the construction problem can be controlled by checking its stability by dragging. At the same time, the use of these specific construction tools may evoke specific geometrical axioms and theorems that can be used for validating the construction procedure within Euclidean Geometry theory. In other words, the solution of a construction problem within a DGE can evoke the theoretical meaning of geometrical constructions. Exploiting the semiotic potential of a DGE may thus lead to developing the mathematical meaning of MT and specifically the meanings of proof referring to a particular theory.

⁵ Actually a DGE provides a larger set of tools, including for instance “measure of an angle,” “rotation of an angle,” and the like. This implies that the whole set of possible constructions does not coincide with that attainable only with ruler and compass. See Stylianides and Stylianides (2005) for a full discussion.

This was the design principle of a number of the long-term teaching experiments I conducted. It involved developing a sequence of didactic cycles using specific construction tools and semiotic activities aimed at the individual and social elaboration of signs (see Mariotti, 2001, 2009). As explained above, the semiotic potential of an artefact concerns the relationship between the meanings emerging from the activities with the artefact and the mathematical meanings evoked.

A *construction task* consists of:

- producing a DGE figure that should be stable by dragging;
- writing a description of the procedure used to obtain the DGE figure and producing a validation of the ‘correctness’ of such a procedure.

Thus a construction task consists of two types of requests. The first asks for interaction with the artefact, the second for producing a written text referring to the interaction. The request for *validating* the correctness of the procedure acquires its meaning in relation to the DGE environment: the construction problem is solved if the figure obtained on the screen passes the dragging test. Validating such a construction means explaining and gaining insight into the reason why it passes the test.

In the framework of TSM, students’ development of a theoretical perspective can be witnessed by the evolution of the sign “construction.” At the beginning, the term construction makes sense only in relation to using particular tools to draw a DGE figure and having that figure pass the dragging test. Later on, the meaning of the term construction acquires the theoretical meaning of geometrical construction (Mariotti, 2001) validated by a proof within a geometry theory. In other words, the evolution of meanings, accomplished in the mathematical discussion led by the teacher, occurs through the elaboration of a correspondence between specific DGE tools and their modes of use on the one hand, and Euclidean axioms and derived theorems and definitions on the other.

At the very beginning, starting from an empty menu, students are invited to discuss the choice of appropriate tools to introduce in the menu. At the same time a corresponding set of @@@construction axioms are formulated and stated as the first core of the geometry theory that any validation should refer to. I want to stress the power of the semiotic potential of a DGE with respect to the possibility of selecting the tools that are available; in other words, the semiotic potential that the artefact “*available menu*” has with respect to the mathematical meaning of theory and specifically to the property of growth—adding new theorems and definitions—that is crucial to understanding the hypothetical–deductive structure of any mathematical theory. As the results of a number of teaching experiments showed, students not only produced new statements and their proofs, but also became aware of the theory within which the proofs made sense, and how such a theory is developed. As long as new problems are solved and new constructions are produced, the corresponding theorems can be validated, added to the set of shared validating principles, and reported in students’ notebooks. The students participated in two parallel processes of evolution: the enlargement of the available menu in the DGE and the corresponding development of a geometry theory.

In summary, a DGE offers a rich and powerful context for introducing students to a theoretical perspective. It provides an environment for phenomenological experiences of the mathematical meanings of:

- axioms that correspond to the use of specific construction tools
- geometrical theorems that validate specific geometrical constructions
- meta-theoretical actions related to the development of the theory by adding new theorems and definitions.

Experiences in the classroom over the course of our teaching experiments confirmed both the unfolding of the semiotic potential and the evolution of an interlaced sense of proof and theory. Different aspects of this evolution are presented and discussed in several papers (Mariotti, 2001, 2007, 2009). The following two excerpts are examples from our findings that show the theoretical meaning of construction and its relationship with the mathematical meaning of theorem.

Example. The theorem of the angle bisector.

The first excerpt shows a student's answer to the task of constructing the angle bisector of an angle using only specific DGE tools such as line, ray, segment (point, point), and compass (point, segment). The students had already learned the correspondence between the use of these tools and the classic three criteria of congruence that constitute the germ of the available theory.

Excerpt 1

Max produces a stable figure in the DGE and the list of the construction steps. Then he writes:

Prove that the angle bisector by construction is an angle bisector by congruence criterion (ita. criterio di uguaglianza)

$AB = AC$ by circle

AO is in common

$OB = OC$ by circle

center A and B

The two triangles are equal because of the third criterion of congruence ($\triangle ABO = \triangle AOC$)

Equal sides correspond to equal angles and thus

$\angle OAC = \angle BAO$

AO is the angle bisector of BAC

As was to be proved.

From the point of view of mathematics, this text is still very rough. However, it is possible to recognize the germ of a proof and, overall, to see how the student is explicitly relating the construction steps to the theoretical elements available. What is particularly significant is the reformulation of the task at the beginning of the proof text ("Prove that the angle bisector..."). It demonstrates the student's need to anticipate interpreting the list of 'theoretical statements' according to the construction.

Excerpt 2

After a first sequence of activities, the teacher opened a collective discussion with the aim of revising the students' personal notebooks. From a comparison of the notebooks, the teacher then guided a mathematical discussion on ordering the sequence of the theoretical elements and giving them the right status: are they axioms, theorems, or definitions?

After the discussion, each student was asked to write a report on the activity. During the discussion some time was devoted to the construction of angle bisectors and the proof of the corresponding "*bisector theorem*." Different proofs were proposed based on applying different theorems. Traces of this part of the discussion can be found in the following report by another student, Stefano. It shows how he grasped the sense of theory both in terms of conventionality and a logically ordered system of statements. He writes:

We then switched to examine the proof of the *bisector theorem*. One of my classmates stated that the *bisector theorem* could be proved also with the isosceles triangle, but to do that we would have needed to have the last theorem concerning the perpendicular. If I say that even having the theorem, we couldn't use it, it doesn't mean that we are fools but simply that when we began [the proof] we didn't have it, and our means for proving were in minor quantity.

In commenting on the intervention of one of his classmates, Stefano explicitly states the need for a proof to refer to the theory available.

4 More About the Semiotic Potential of a DGE

In this section I elaborate a bit more on the potential of a DGE with respect to the notion of mathematical theorem. In particular, I consider the semiotic potential offered by dragging in relation to the third component of MT—the statement.

Difficulties often arise in the interpretation of a given statement to be proved. These difficulties concern the meaning of the premise and the conclusion, as well as the meaning of the logical dependency between them. Not many studies have been devoted to this specific issue. However, an interesting exception is the work of Selden and Selden (1995) which discussed the specific phenomenon of "unpacking an informal statement." This refers to the challenge that students often face of making the formal elements—for instance, the logical quantifiers—of the statement to be proved explicit.

This difficulty has a parallel in the challenges students face when asked to formulate a conjecture in the form of a conditional statement—"if ... then ..." (Boero, Garuti, & Lemut, 1999). The failure to manage conditionality and to grasp the different status of premises and conclusions may be a true obstacle to developing a correct meaning of MT. Developing the mathematical meaning of a conditional statement can therefore be considered a crucial issue in the general context of developing the meaning of MT.

In the current literature, there is a shared opinion about the fundamental role that open problems and conjecturing activities play in developing a sense of proof

and fostering a productive relationship between ‘spontaneous’ argumentation processes and theoretical validation (Arsac & Mante, 1983; Arsac, 1992; Pedemonte, 2002). Different contexts allow for open questions in different ways, thereby offering different potentials for posing and solving open problems and, consequently, for formulating conjectures. In the following section, I focus on conjecturing tasks and the very particular context of Dynamic Geometry. Specifically, I illustrate the semiotic potential of specific dragging modalities performed in a DGE context while solving conjecturing tasks.

Previous studies carried out by Boero and his colleagues focused on different aspects of students’ real world experience, but showed how dynamic aspects of the phenomena under investigation were fundamental. Their studies confirmed what other studies claimed (Simon, 1996; Harel & Sawder, 1998)—that dynamicity seemed to foster transformational mental processes that are key to producing conditional statements. The formulation of a conjecture can be described as a “crystallization” of a dynamic exploration—a specific moment, and a specific position, when the occurrence of one fact in a conditional statement has the occurrence of another fact as a consequence (Boero et al., 1999, 2007). This makes it reasonable to address the role of modes of dragging in conjecture production and to consider the solution of conjecturing open problems in a DGE.

4.1 Conjecturing in a DGE: Dragging as a Semiotic Mediator of Conditionality

I use the term ‘conjecture open problem’ in the following to refer to a task that explicitly asks the solver to formulate a conjecture (Mariotti, 2014). This is a very common case in geometry and involves asking the solver to formulate a conditional statement expressing a possible logical dependency between the geometrical properties of a given configuration. In a DGE, preliminary explorations are expected that involve, firstly, the construction of a dynamic figure implementing the initial configuration and, then, active transformations of the figure in search of a possible answer. This means that while observing the dynamic image on the screen, the solver has to interpret the perceptual data coming from the screen and transform them into geometrical properties that formulate a statement expressing a conditional relationship between the properties.

Several studies done on students’ exploration processes show both different dragging modalities and the potential of such modalities in assisting the conjecturing process (Arzarello, Olivero, Paola, & Robutti, 2002; Olivero, 2003; Hölzl, 1996; Leung & Lopez-Real, 2002; Lopez-Real & Leung, 2006). Elaborating on these results, it is possible to outline the semiotic potential of particular modalities of dragging with respect to the mathematical meaning of *conditional statement* in a geometry context. Dragging modalities can be considered as specific artefacts used to solve

an open problem, and the meanings emerging from their use can be related to the mathematical meanings of premise, conclusion, and the logical dependency between them.

4.2 *Invariants by Dragging and Their Relationship*

The notion of *invariant* by dragging is at the core of any DGE. As discussed above, when a figure is acted upon, two kinds of properties simultaneously appear as invariants—those stated by the commands used in the construction and all the resulting properties within Euclidean Geometry. This means that a specific *relationship between invariants* is preserved by dragging, and this relationship corresponds to the validity of a logical implication between properties of a geometrical figure. This becomes a crucial element when solving an open problem asking for a conjecture.

Because of their simultaneity, it may be difficult to maintain control of the logical hierarchy between the different invariants. Nevertheless, a careful analysis of the movement of the different elements of a figure (see Mariotti, 2014) reveals an asymmetry between the two kinds of invariants. In other words, two different movements occur that are worth distinguishing and analysing carefully. One movement—*direct motion*—is the variation of an element in the plane under the direct control of the mouse. The second movement—*indirect motion*—is the variation of any other element as a consequence of direct motion.

During a dynamic exploration, the solver can ‘feel’ motion dependency through a conscious use of the dragging tool. This allows him/her to distinguish between *direct invariants* and *indirect invariants* and interpret their dynamic relationship in terms of the logical consequences between geometrical properties, and eventually express it as a conditional statement between a premise and a conclusion.

Let us consider the following conjecturing open problem: *given a quadrilateral and the midpoint of its sides, what can we say about the quadrilateral that has these midpoints as vertices.*

Once the quadrilateral and its midpoints have been constructed, explorations of the possible configurations make rather evident the emergence of new properties concerning both the parallelism and the equality between the sides of the new quadrilateral. This may lead to the conjecture (Varignon’s Theorem): “Given a quadrilateral and the midpoint of its sides, the quadrilateral that has these midpoints as vertices is a parallelogram.”

The distinction between direct and indirect movement produces a new interpretation of the classic results on different dragging explorations. The modality of dragging previously described as *Dummy locus dragging* (or *Lieu muet dragging*) is especially worthy of attention. This modality consists of dragging a configuration with the intention of maintaining a specific property; that is, achieving a constrained movement of the original figure *as if* a specific property were ‘invariant.’ This type of invariant named *Indirectly Induced Invariant* (Baccaglini-Frank & Mariotti, 2010) corresponds to the consequence of the combination of all the properties given by the

construction plus a new hypothesis corresponding to the constrained dragging. In other words, via the constrained dragging that we call *Maintaining Dragging* (MD) (Baccaglioni-Frank & Mariotti, 2010), a new property is added to the initial premises. This corresponds to what mathematicians commonly refer to as exploring “under which condition...a certain property occurs.”

What is meaningful for my purpose here is that using MD to solve a conjecturing open problem, the student can directly and intentionally control the distinction between which property is maintained and which property is searched. This distinction corresponds to the distinction between the premises and conclusion of a conditional statement: the conclusion is the property the solver decides to maintain, the premise is the property corresponding to the constrained movement, and the conditional relationship between these properties corresponds to the simultaneity of their occurrence.

Taking the perspective of semiotic mediation, I claim that the different dragging modalities, together with the different types of invariants, offer rich semiotic potential with respect to the mathematical notion of conjecture and specifically to the mathematical meaning of a conditional statement as the logical relationship between premises and a conclusion. The asymmetry of the relationship between invariants offers the possibility of distinguishing the logical status of the properties of a DGE figure; that is, their status as premise or conclusion. Thus, according to the previous analysis, it is possible to outline the following semiotic potential of the different means of dragging in solving a conjecture-production task in respect to the mathematical meaning of conditionality. The semiotic potential is recognizable in the relationship between:

- the indirectly induced invariant (the property the solver intends to achieve) and the mathematical meaning of the conclusion of the conjecture statement
- the invariant constrained by the specific goal-oriented movement (the property that must be assumed in order to obtain the induced invariant) and the mathematical meaning of the premise of the conjecture statement
- the haptic sensation of causality relating the direct and the indirect movement and the mathematical meaning of logical dependence between premise and conclusion.

Results from several studies show how different meanings related to the notion of conjecture may emerge and how the different kinds of invariant can be characterized by their specific status in the activity of exploration. These results can be used by teachers to exploit the semiotic potential of dragging and specifically of MD (see Baccaglioni-Frank & Mariotti, 2010).

5 Impossible Figures and Proof by Contradiction

In the previous sections, I discussed specific aspects of the didactic potential of a DGE for introducing students to mathematical theorems. In this section, I focus on the potential offered by a DGE with respect to indirect proof—that is, proof by

contradiction and proof by contraposition. Before showing examples, I present a short account of the model of indirect proof.

Given a *principal statement*, there are two levels at which a proof develops: the *theoretical* level and the *meta-theoretical* level. The very beginning of the proving process consists of a shift to a new statement characterized by new premises. It is usually introduced by the claim “let us start from negating the conclusion.” We call this new statement the *secondary statement*. This new statement is related to the principal statement by the fact that its premise includes the negation of the conclusion of the principal statement. The validation of the secondary statement is reached through a direct deductive proof.

The relationship between the validation of the secondary statement and the expected proof of the principal statement is usually taken for granted—commonly ratified by the generic assertion “thus the theorem is proved.” However, after proving the secondary statement, something remains unsolved, as clearly explained by Leron (1985):

Formally, we must be satisfied that the contradiction has indeed established the truth of the theorem (having falsified its negation), but psychologically, many questions remain unanswered. What have we really proved in the end? What about the beautiful constructions we built while living for a while in this false world? Are we to discard them completely? And what about the mental reality we have temporarily created? I think this is one source of frustration, of the feeling that we have been cheated, that nothing has been really proved, that it is merely some sort of a trick—a sorcery—that has been played on us. (p. 323)

The crucial point lies in the final laconic assertion: “thus the theorem is proved.” Validating the principal statement pertains to the meta-theoretical level and condenses a meta-theorem relating the validation of the secondary statement to the validation of the principal statement. What is often missing is something that could bridge the gap between the validation of the principal statement and the absurd conclusion resulting from the proof of the secondary statement. In order to clarify the source of such difficulties for students, investigations focused on posing problems in a DGE. The aim was to explore if and how a DGE offers a base for bridging the gap.

5.1 *Dragging Impossible Figures*

The appearance of conflicting or impossible configurations is one of the critical elements of the production of an indirect proof (Fischbein, 1993). In the case of DGE figures, Leung & Lopez-Real (2002) introduced the notion of *pseudo object* to refer to a figure on which the user forces an assumption so that it is “biased with extra meaning.” “This biased DGE,” they maintain, “exists as a kind of hybrid state between the visual-true DGE (a virtual representation of the Euclidean world) and a pseudo-true interpretation” (p. 22).

Interesting behaviours are described when the solution of a conjecturing task involves impossible robust figures. They show how the aim of restoring harmony between the figural and theoretical aspects (Fischbein, 1993) can help not only to

overcome a possible impasse, but also construct the argument providing the missing step for validating the falsity of an assumption (Antonini & Mariotti, 2008). The link between premises and conclusions, expressed by the relationship between invariant properties observed on the screen after dragging a figure, may contribute to bridging the (logical) gap between the absurd conclusion coming from the proof of the secondary statement and the validation of the principal statement. In other words, in a DGE, this potential bridge can be realized with the support of dragging modes that induce the solver to conceive a pseudo object. The dynamic of the figure induces the solver to interpret the constructed figure as simultaneously representing properties that are contradictory within Euclidean theory.

In the following, I report some results from our classroom research via two exemplary cases. They refer to two different formulations of an open problem, both leading to a conjecture that is expected to be supported by an indirect argument, and consequently providing an introduction to indirect proof.

Case 1. The case of Paolo and Riccardo

This first case concerned the task: What can you say about the angle formed by two angle-bisectors in a triangle?

Exploring the possible configurations can lead the solver to consider the case of orthogonality between two angle-bisectors, and to the conjecture that this case is impossible. Among the protocols of solving processes collected in our studies (Mariotti & Antonini, 2009), we found examples of indirect arguments leading to a contradictory conclusion. The following example is drawn from the interview of a pair of grade 12 students, Paolo and Riccardo.

In the first part of the exploration, Paolo and Riccardo consider the case that the angle between the angle bisectors is an obtuse angle. They then exclude this possibility and move on to consider the case of orthogonality.

61 P: As for 90 [degree], it would be necessary that [...] $K/2 = 45$, $H/2 = 45$ [...].

62 I: In fact, it is sufficient that [...] $K/2 + H/2$ is 90.

63 R: Yes, but it cannot be.

64 P: Yes, but it would mean that $K + H$ is ... a square [...]

65 R: It surely should be a square, or a parallelogram

66 P: $(K-H)/2$ would mean that [...] $K + H$ is 180° ...

67 R: It would be impossible. Exactly, I would have with these two angles already 180, that surely it is not a triangle. [...]

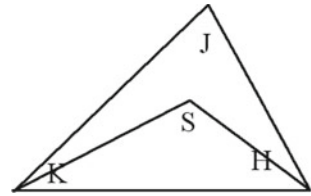
71 R: We can exclude that [the angle] $\frac{\pi}{2}$ is [right] because it would become a quadrilateral. [...]

81 R: [the angle] is not 90° because I would have a quadrilateral, in fact the sum of the two angles would be already 180, without the third angle.

Then the only possible case is that I have a quadrilateral, that is, the sum of the angles is 360 (Fig. 1).

Using an abductive argument, implicitly based on the theorem about the sum of the measures of the angles of a triangle, Paolo and Riccardo arrive at the conjecture: “it is sufficient that [...] $K/2 + H/2$ is 90.” But Riccardo acknowledges the

Fig. 1 Paolo and Riccardo's figure



impossibility of this condition (63), immediately followed by Paolo who identifies an immediate consequence of the configuration: “it would mean $K + N$ is ...” Realizing the absurd, he seeks a figural interpretation of the ‘absurd’ conclusion that generates the adaptation of the figure: “*it surely should be a square, ...*” (65). The falsity of the original assumption is now acceptable “*because it would become a quadrilateral,*” i.e., not a triangle. More specifically, a new interpretation of the image on the screen is achieved that fulfils the given properties, but also leads to a new conclusion allowing the students to overcome the previous contradictory conclusion. This new interpretation gives sense and opens the development of an indirect argument.

This protocol shows the dynamics of a pseudo object: the initial appearance of an impossible figure is overcome by the image of a square, immediately generalized into a parallelogram. This new image solves the distress of the absurd without canceling its origin.

Because of its nature as pseudo object—representing a contradictory relation and likely to turn into a coherent representation—the dynamic figure acted upon by the solver has the potential, on the one hand, to offer support to the proof of the secondary statement and, on the other, to maintain the relationship between the secondary statement and the principal statement.

According to the model above, the secondary statement “If S (the angle between the angle bisectors) is right, then the configuration becomes a quadrilateral” can be interpreted as “It is not possible that S is right because otherwise the triangle would become a quadrilateral.” This interpretation allows the solver to relate the secondary statement to the principal statement.

Case 2. The Case of Stefano and Giulio

Let us now consider the second case (Baccaglioni-Frank et al., 2013). The structure of the problem is still a conjecturing open problem, but the text explicitly requests a geometrical construction and an explanation for a negative answer.

The task sounds like this: Is it possible to construct a triangle with two perpendicular angle bisectors? If so, provide steps for a construction. If not, explain why.

In the following example, similar to what happened in the previous case, we can observe how the emergence of a pseudo object can be related to a first awareness about the impossibility of a construction. However, we can also observe the role played by the DGE figure in that emergence. The excerpt is drawn from the interview of a pair of high school students (grade 12), Stefano and Giulio.

1. Stefano: No, the only way is to have 90 degree angles... [unclear which angles these are as he was not constructing the figure or looking at the screen]

2. Giulio: That for a triangle is a bit difficult! [giggling]... So... they have to be...
3. Stefano: If triangles have 4 angles...
4. Giulio: No, I was about to say something silly...

Stefano immediately states that there is only a possibility. Some elements of an impossible configuration are mentioned and then the students quickly move on to constructing a figure in the DGE. Giulio constructs two perpendicular lines and refers to them as the bisectors of the triangle (Fig. 2a).

5. Stefano: Yes, these are bisectors, right?
6. Interviewer: Yes.
7. Giulio: So, now we need to get... bisectors... how can we have an angle from the bisector?
8. Giulio: the symmetric image?... It's enough to do the symmetric of this one.

The solvers have constructed a figure with two robust angle bisectors that intersect perpendicularly (Fig. 2b).

9. Stefano: The only thing is that this (Fig. 2b) isn't a triangle!
10. Giulio: Therefore now we could do like this here [drawing the lines through the symmetric points and the two drawn vertices of the triangle]
11. Interviewer: Yes.
12. Stefano: It's that something atrocious comes out!
13. Giulio: And here... theoretically the point of intersection should be ... the points... very small detail...hmmm
14. Stefano: No, we proved that this is equal to this [pointing to angles], and this is equal to this because they are bisectors... these two are equal so these are parallel.
15. Stefano: These two [referring to the two parallel lines] have a hole so it is not a triangle.

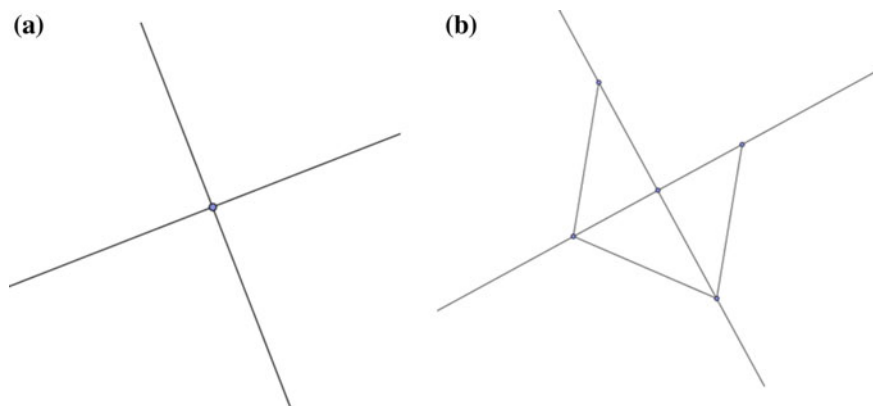


Fig. 2 a Giulio's construction of the angles' bisectors. b The completed figure

The solvers use the DGE to construct two perpendicular lines. Then the property of being bisectors is realized through constructing the symmetric image of a generic segment that has its extremes each on a different line. Once the construction is completed, it is possible to observe the properties that are consequences of these two robustly constructed properties. The students notice that “the figure must have two adjacent angles with two parallel sides” (Lines 9–14) and, at the same time, that there is “a hole” in what was expected to be a triangle (Line 15). The pseudo object emerges. It has a base as a triangle should have and two parallel sides as a parallelogram has. The figure on the screen has the property “triangle” projected onto it, though it is clearly not a triangle. This pseudo object shows that it is not possible to robustly construct what they were asked to construct, but at the same time shows why it is not possible. This figure, because of its nature as a pseudo object, enables connecting the principal statement “the requested construction is not possible” with the secondary statement “if the angle bisectors are perpendicular, then the triangle is a quadrilateral.”

Our investigations of indirect argumentation and indirect proof produced by students have shown that they may be supported by dragging exploration leading to perceiving what we have called pseudo objects (Baccaglini-Frank et al. 2013, p. 65). Nevertheless, the effectiveness of the appearance of a pseudo object for triggering an indirect argument is related to the logical control of the dynamic figure. In other words, what is essential is awareness of the logical meaning of the dynamic figure—awareness that allows the solver at the same time to project on it the expected properties and to recognize in it the consequences of the constructed properties. Such a double logical interpretation of the dynamic figure allows the solver to make sense of the “absurd” and provides him/her with a bridge to fill the logical gap between the principal and the secondary statement.

Based on these results, it seems possible to claim that a DGE offers a suitable context for handling indirect proof because dragging to produce an impossible configuration or pseudo object provides an informal language to talk about the absurd. This reminds us of Thompson’s (1996) claim:

If such indirect proofs are encouraged and handled informally, then when students study the topic more formally, teachers will be in a position to develop links between this informal language and the more formal indirect-proof structure. (p. 480)

Specifically, within a DGE context, open problems that ask about the construction of geometrically impossible figures may play a crucial role in unfolding the semiotic potential of dragging with respect to the mathematical meanings of deriving the absurd. In this way, they contribute to developing meanings related to indirect proof.

6 Conclusions

I begin my concluding arguments by reflecting on the distance between the way in which experts and novices approach the discovery of a new theorem. As Polya wrote (De Villiers, 2001), mathematicians are highly motivated to search for a proof:

[H]aving verified the theorem in several particular cases, we gathered strong inductive evidence for it. The inductive phase overcame our initial *suspicion* and gave us a strong *confidence* in the theorem. Without such *confidence* we would have scarcely found the courage to undertake the proof which did not look at all a routine job. When you have satisfied yourself that the theorem is *true*, you start *proving* it. (Polya, 1954, pp. 83–84, emphasis added)

Such experiences, so natural for experts, do not belong to the reality of novices and students who commonly gravitate toward empirical approaches to truth and often disagree only on the number of confirmations needed. In spite of the fact that proof lies at the heart of mathematics, research on mathematics education has shown the complexity of fostering students' sense of proof and, more generally, of introducing a theoretical perspective in school mathematics. The advent and development of new technological devices has opened new directions of research and posed crucial questions about the possibilities and challenges of supporting the transition from informal to formal proof in mathematics.

In this chapter, I selected the specific technological context of a DGE and, elaborating on results from my investigations, discussed its potential for teaching at the secondary school level. I used TSM to frame the educational context and focused on the notion of semiotic potential to describe the relationship between using specific DGE tools for specific tasks, the situated meanings that are expected to emerge, and their connection with the mathematical meanings related to proof or, more broadly, mathematical theorem. I explained the semiotic potential that emerged in solving a construction task and showed how it relates, on the one hand, to the meaning of drawing constrained by the use of specific tools and, on the other, to the mathematical meaning of proving a statement within a set of shared assumptions and theorems. Exploiting this semiotic potential allows teachers to guide students in constructing and intertwining different meanings of MT. Specifically, it allows teachers to guide the co-emergence of the mathematical meaning of proof and theory.

Further analysis of dragging modalities highlighted the semiotic potential of conjecturing open problems in a DGE. Different dragging modalities can be related to different types of invariants and also to the different logical status of the properties of the geometrical object represented on the screen. Some properties correspond to the premise of a statement and others to the conclusion, while their simultaneity corresponds to the logical relationship between them. In other words, the mathematical meaning of a conditional statement can be related to the relationship between the specific invariant elements that emerge in exploring a configuration and the intention of producing a conjecture.

Additional semiotic potentials of dragging emerged in the solving of non-constructability tasks. The semiotic potential of a pseudo object was shown to relate to the mathematical meaning of indirect proof. Specifically, I claim that, in a DGE, rigid

construction combined with intentional and controlled dragging may lead the solver to perceive the figure as degenerating into a pseudo object. The hybrid nature of pseudo objects seems to support making sense of indirect proof and creates a bridge between the principal statement to be proved and the absurd configuration emerging from the proof of the secondary statement.

Beyond illustrating the educational potential of a specific technology, the discussion offers insight into the complexity of managing problem solving productively in a DGE. The strict dependence between the experience of acting *geometrically* in a DGE and the development of a coherent web of mathematical meanings enabling theoretical control over what is drawn and moved on the screen gives us a key for interpreting both the strength and the fragility of using DGE activities in school practice. In particular, experimenting with conjecturing and proving in a DGE requires adequate training in a mathematician's eye and feel for theory. After all, interpreting one's own perceptions as mathematical evidence in terms of geometric properties and logical relationships is not a spontaneous or immediate process. On the contrary, developing a theoretical eye is the result of a complex learning process that is inconceivable without a teacher's expertise in setting specific tasks and guiding student awareness of the link between their personal experience and mathematical knowledge.

References

- Antonini, S., & Mariotti, M. A. (2008). Indirect proof: What is specific to this way of proving? *ZDM: Mathematics Education*, 40(3), 401–412.
- Arsac, G., & Mante, M. (1983). Des "problème ouverts" dans nos classes du premier cycle. *Petit x*, 2, 5–33.
- Arsac, G. (1992). *Initiation au raisonnement au college*. Presse Universitaire de Lyon.
- Arzarello, F., Olivero, F., Paola, D., & Robutti, O. (2002). A cognitive analysis of dragging practises in Cabri environments. *Zentralblatt für Didaktik der Mathematik*, 34(3), 66–72.
- Arzarello, F. (2007). The proof in the 20th century: From Hilbert to automatic theorem proving. In P. Boero (Ed.), *Theorems in school from history and epistemology to cognitive and educational issues* (pp. 43–64). Rotterdam: Sense Publishers.
- Arzarello, F., Bartolini Bussi, M. G., Leung, A. Y. L., Mariotti, M. A., & Stevenson, I. (2012). Experimental approaches to mathematical thinking: Artefacts and proof. In G. Hanna, & M. De Villier, *Proof and proving in mathematics education* (pp. 1–10). Springer, New ICMI Study Series, Volume 15, 2012.
- Baccaglioni-Frank, A., & Mariotti, M. A. (2010). Generating conjectures in dynamic geometry: The maintaining dragging model. *International Journal of Computers for Mathematical Learning*, 15(3), 225–253.
- Baccaglioni-Frank, A., Antonini, S., Leung, A., & Mariotti, M. A. (2013). Reasoning by contradiction in dynamic geometry. *PNA*, 7(2), 63–73.
- Baccaglioni-Frank, A., Antonini, S., Leung, A., & Mariotti, M. A. (2018). From pseudo-objects in dynamic explorations to proof by contradiction. *Digital Experiences in Mathematics Education*, 1–23. <https://doi.org/10.1007/s40751-018-0039-2>.
- Balacheff, N. (2008). The role of the researcher's epistemology in mathematics education: an essay on the case of proof. *ZDM: The International Journal on Mathematics Education*, 40, 501–512.

- Bartolini Bussi, M.G. (1996). Mathematical discussion and perspective drawings in primary school. *Education Studies in Mathematics* 31, 11–41.
- Bartolini Bussi, M. G., & Mariotti, M. A. (2008). Semiotic mediation in the mathematics classroom: Artifacts and signs after a Vygotskian perspective. In L. English, M. Bartolini Bussi, G. Jones, R. Lesh, & D. Tirosh (Eds.), *Handbook of international research in mathematics education, Second revised edition* (pp. 746–805). Lawrence Erlbaum, Mahwah, NJ.
- Boero, P., Garuti, R., & Lemut, E. (1999). About the generation of conditionality of statements and its links with proving. *Proceedings of the Conference of the International Group for*, 2, 137–144.
- Boero, P., Garuti, R., & Lemut, E. (2007). Approaching theorems in grade VIII: Some mental processes underlying producing and proving conjectures, and conditions suitable to enhance them. In P. Boero (Ed.), *Theorems in school: From history, epistemology and cognition to classroom practice* (pp. 247–262). Rotterdam, The Netherlands: Sense Publishers.
- de Villiers, M. (1998). An alternative approach to proof in dynamic geometry. In R. Lehrer, & D. Chazan (Eds.), *Designing learning environments for developing understanding of geometry and space* (pp. 369–394). Erlbaum, Mahwah.
- De Villiers, M. (2001). Papel e funcoes da demonstracao no trabalho com o Sketchpad. *Educacao Matematica*, 63, 31–36. (retrived online: <https://mzone.mweb.co.za/residents/profmd/proofc.pdf>).
- Dreyfus, T., & Hadas, N. (1996). Proof as answer to the question why. *Zentralblatt fur Didaktik der Mathematik/International Reviews on Mathematical Education*, 28(1), 1–5.
- Duval, R. (2007). Cognitive functioning and the understanding of mathematical processes of proof. In P. Boero (Ed.), *Theorems in school: From history, epistemology and cognition to classroom practice* (pp. 138–162). Rotterdam, The Netherlands: Sense Publishers.
- Fischbein, E. (1993). The theory of figural concepts. *Educational Studies in Mathematics*, 24(2), 139–162.
- Hadas, N., Hershkowitz, R., & Schwarz, B. (2000). The role of contradiction and uncertainty in promoting the need to prove in dynamic geometry environments. *Educational Studies in Mathematics*, 44(1–3), 127–150.
- Hanna, G. (1989). More than formal proof. *For the Learning of Mathematics*, 9(1), 20–25.
- Hanna, G., & Jahnke, H. N. (2007). Proving and modelling. In W. Blum, P. L. Galbraith, H.W. Henn, & M. Niss (Eds.), *Applications and modelling in mathematics education. The 14th ICMI study* (pp. 145–152). Dordrecht: Springer.
- Hanna, G., Jahnke, H. N., & Pulte, H. (Eds.). (2009). *Explanation and proof in mathematics: Philosophical and educational perspectives*. Berlin: Springer.
- Hanna, G., & De Villiers, M. (Eds.). (2012). *Proof and proving in mathematics education: The 19th ICMI study* (Vol. 15). Springer Science & Business Media.
- Harel, G., & Sowder, L. (1998). Students' proof schemes: Results from exploratory studies. In A. Schonfeld, J. Kaput, & E. Dubinsky (Eds.) *Research in collegiate mathematics education III*. (Issues in Mathematics Education, Vol. 7, pp. 234–282). American Mathematical Society.
- Herbst, P. G. (2002). Establishing a custom of proving in American school geometry: Evolution of the two-column proof in the early twentieth century. *Educational Studies in Mathematics*, 49(3), 283–312.
- Hölzl, R. (1996). How does “dragging” affect the learning of geometry. *International Journal of Computer for Mathematical Learning*, 1(2), 169–187.
- Laborde, C. & Laborde, J. M. (1991). Problem solving in geometry: From microworlds to intelligent computer environments. In J. P. Ponte, J. F. Matos, & D. Fernandes (Eds.), *Mathematical problem solving and new information technologies* (pp. 177–192). NATO ASI Series F, New York: Springer.
- Laborde, J. M., & Strässer, R. (1990). Cabri-géomètre: a microworld of geometry for guided discovery learning. *Zentralblatt für Didaktik der Mathematik*, 90(5), 171–177.
- Leron, U. (1985). A Direct approach to indirect proofs. *Educational Studies in Mathematics*, 16(3), 321–325.

- Leung, A., & Lopez-Real, F. (2002). Theorem justification and acquisition in dynamic geometry: A case of proof by contradiction. *International Journal of Computers for Mathematical Learning*, 7, 145–165.
- Lopez-Real, F., & Leung, A. (2006). Dragging as a conceptual tool in dynamic geometry. *International Journal of Mathematical Education in Science and Technology*, 37(6), 665–679.
- Mariotti, M. A. (2001). Justifying and proving in the Cabri environment. *International Journal of Computer for Mathematical Learning*, 6(3), 257–281.
- Mariotti, M.A. (2006). Proof and proving in mathematics education. In A. Gutiérrez & P. Boero (Eds.) *Handbook of research on the psychology of mathematics education* (pp. 173–204). Rotterdam, The Netherlands: Sense Publishers.
- Mariotti, M. A. (2007). Geometrical proof: The mediation of a microworld. In P. Boero (Ed.), *Theorems in school: From history epistemology and cognition to classroom practice* (pp. 285–304). Rotterdam, The Netherlands: Sense Publishers.
- Mariotti, M. A. (2009). Artifacts and signs after a Vygotskian perspective: The role of the teacher. *ZDM: The International Journal on Mathematics Education*, 41, 427–440.
- Mariotti, M. A. (2010). Proofs, semiotics and artefacts of information technologies. In G. Hanna, H. N. Jahnke, & H. Pulte (Eds.), *Explanation and proof in mathematics: Philosophical and educational perspectives* (pp. 169–190). Springer.
- Mariotti, M. A. (2012). Proof and proving in the classroom: Dynamic geometry systems as tools of semiotic mediation. *Research in Mathematics Education* 14(2), 163–185 (2012).
- Mariotti, M. A. (2014). Transforming images in a DGS: The semiotic potential of the dragging tool for introducing the notion of conditional statement. In S. Rezat, M. Hattermann, & A. Peter-Koop (Eds.), *Transformation. A fundamental idea of mathematics education*. Springer New York.
- Mariotti, M. A. & Antonini, S. (2009). Breakdown and reconstruction of figural concepts in proofs by contradiction in geometry. In F. L. Lin, F. J. Hsieh, G. Hanna, & M. de Villers (Eds.), *Proof and proving in mathematics education, ICMI study 19 conference proceedings* (Vol. 2, pp. 82–87).
- Mariotti, M.A., Bartolini Bussi, M., Boero, P., Ferri, F., & Garuti, R. (1997). Approaching geometry theorems in contexts: from history and epistemology to cognition. In E. Pehkonen (Ed.), *Proceedings of PME-XXI*, (Vol. 1, pp. 180–195). Lathi, Finland.
- Miyazaki, M., Fujita, T., & Jones, K. (2015). Flow-chart proofs with open problems as scaffolds for learning about geometrical proofs. *ZDM: International Journal on Mathematics Education*, 47(7), 1–14.
- Olivero, F. (2003). Proving within dynamic geometry environments, Doctoral Dissertation, Graduate School of Education, Bristol. [https://telearn.archives-ouvertes.fr/file/index/docid/190412/ filename/Olivero-f-2002.pdf](https://telearn.archives-ouvertes.fr/file/index/docid/190412/filename/Olivero-f-2002.pdf).
- Pedemonte, B. (2002) *Etude didactique et cognitive des rapports de l'argumentation et de la démonstration en mathématiques*, (Unpublished) Thèse de Doctorat, Université Joseph Fourier, Grenoble. <http://tel.archives-ouvertes.fr/tel-00004579/>.
- Reid, D. A., & Knipping, C. (2010). *Proof in mathematics education: Research, learning and teaching*. Rotterdam, The Netherlands: Sense Publisher.
- Schoenfeld, A. H. (1985). *Mathematical problem solving*. New York: Academic press.
- Selden, J., & Selden, A. (1995). Unpacking the logic of mathematical statements. *Educational Studies in Mathematics*, 29(2), 123–151.
- Sierpinska, A. (2005). On practical and theoretical thinking. In M. H. G. Hoffmann, J. Lenhard, & F. Seeger (Eds.), *Activity and sign—Grounding mathematics education. Festschrift for Michael Otte* (pp. 117–135) New York: Springer.
- Simon, M. A. (1996). Beyond inductive and deductive reasoning: The search for a sense of knowing. *Educational Studies in Mathematics*, 30(2), 197–209.
- Sinclair, N., & Robutti, O. (2012). Technology and the role of proof: The case of dynamic geometry. In *Third international handbook of mathematics education* (pp. 571–596). New York, NY: Springer.
- Sinclair, N., Bussi, M. G. B., de Villiers, M., Jones, K., Kortenkamp, U., Leung, A., & Owens, K. (2017). Geometry education, including the use of new technologies: A survey of recent research.

- In *Proceedings of the 13th international congress on mathematical education* (pp. 277–287). Cham: Springer.
- Stylianides, G. J., & Stylianides, A. J. (2005). Validation of solutions of construction problems in dynamic geometry environments. *International Journal of Computers for Mathematical Learning*, 10(1), 31–47.
- Stylianides, A. J., Bieda, K. N., & Morselli, F. (2016). Proof and argumentation in mathematics education research. In *The second handbook of research on the psychology of mathematics education* (pp. 315–351). Rotterdam: Sense Publishers.
- Thompson, D. R. (1996). Learning and teaching indirect proof. *The Mathematics Teacher*, 89(6), 474–482.

Journeys in Mathematical Landscapes: Genius or Craft?



Lorenzo Lane, Ursula Martin, Dave Murray-Rust, Alison Pease
and Fenner Tanswell

1 Prelude

In 1993 Andrew Wiles announced the proof of Fermat's Last Theorem. A subsequent interview by the US PBS (Wiles, 2000) plays to the popular notion of a lone genius, waiting for inspiration to strike, and highlights the "passion and emotion" of mathematics, lingering on the moment where Wiles says

And sometimes I realized that nothing that had ever been done before was any use at all. Then I just had to find something completely new; it's a mystery where that comes from,

followed by a dramatic pause.

Mathematical genius and creativity have received much attention (Robinson, 2011). Yet the interview also highlights more day-to-day aspects of mathematicians' work, which in this paper we designate as the "craft" of mathematics, as in the paragraph immediately preceding the quote above

L. Lane · U. Martin (✉)
University of Oxford, Oxford, UK
e-mail: Ursula.Martin@maths.ox.ac.uk

L. Lane
e-mail: lorenzo.lane@hotmail.co.uk

D. Murray-Rust
University of Edinburgh, Edinburgh, Scotland
e-mail: d.murray-rust@ed.ac.uk

A. Pease
University of Dundee, Dundee, Scotland
e-mail: a.pease@dundee.ac.uk

F. Tanswell
University of St Andrews, St Andrews, Scotland
e-mail: f.tanswell@lboro.ac.uk

I used to come up to my study, and start trying to find patterns. I tried doing calculations which explain some little piece of mathematics. I tried to fit it in with some previous broad conceptual understanding of some part of mathematics that would clarify the particular problem I was thinking about. Sometimes that would involve going and looking it up in a book to see how it's done there. Sometimes it was a question of modifying things a bit, doing a little extra calculation.

These small explorations form part of a larger whole, cast as exploring an unknown landscape:

Perhaps I can best describe my experience of doing mathematics in terms of a journey through a dark unexplored mansion. You enter the first room of the mansion and it's completely dark. You stumble around bumping into the furniture, but gradually you learn where each piece of furniture is. Finally, after six months or so, you find the light switch, you turn it on, and suddenly it's all illuminated. You can see exactly where you were. Then you move into the next room and spend another six months in the dark. So each of these breakthroughs, while sometimes they're momentary, sometimes over a period of a day or two, they are the culmination of—and couldn't exist without—the many months of stumbling around in the dark that proceed them.

This notion of “stumbling” inspired a previous paper (Martin, 2015), where online and computational mathematics were analysed to shed light this fine-grained mathematical practice of the craft of mathematics.

This paper is an initial study of this craft, attempting to reconcile the contrasting notions of genius and craft through viewing the mathematician as crafting a journey through a mathematical landscape, with mathematical education providing wayfarer with the necessary skills and tools. We look first, in Sect. 2, at mathematicians' metaphors of journeys in space; then in Sect. 3 give an initial indication of how these might be framed in terms of literary studies, social science and philosophy, suggesting that ideas of explorations of a fixed landscape might be broadened to consider how mathematicians themselves create that landscape. In Sect. 4 we contrast such notions of genius and inspiration in traversing the landscape with notions of mathematics education as developing skills in the learner. In Sect. 5 we discuss the “polymath” online collaborations, a form of “social machine”, and their use in mathematics education. In Sect. 6 we suggest how theories of craft, in particular Ingold's notion of crafting as wayfaring, open up new possibilities for framing the practice of mathematics, and shed further light on the educational role of polymath collaborations.

2 Mathematicians on Mathematics: The Journey in Space

Newton's remark that

I know not what I may seem to the world, but as to myself, I seem to have been only like a boy playing on the sea-shore and diverting myself in now and then finding a smoother pebble or a prettier shell than ordinary, whilst the great ocean of truth lay all undiscovered before me. (Turnow, 1806), cited in (Schaffer, 2009, p. 243).

is often taken as the epitome of the lone mathematician exploring the natural world, and modestly presenting his (always his) activities to a small coterie of followers. Schaffer (2009) observes that the remark, or alleged remark, only reported some years after Newton's death, is likely to have been taken up so that Newton's isolation and intellectual approach might add authority to his ideas. Newton worked among informants skilled in so-called "practical mathematics", the arithmetic and geometry needed for accounting, surveying, navigation and warfare, and a staple of education for anyone above the labouring classes. The nineteenth century saw this tradition somewhat at odds with the rise of abstraction in mathematical education and research, following more rigorous approaches emerging from Europe.

Augustus De Morgan, a celebrated educator in this newer more abstract tradition, writes in the 1842 preface to his influential (and monumental) calculus textbook (De Morgan, 1842) that

the way to enlarge the settled country [of mathematics] has not been by keeping within it, but by making voyages of discovery (De Morgan, 1842, p. vii)

and quotes Newton's supposed remark approvingly to his pupil Ada Lovelace (Hollings et al., 2017)

That which you say about the comparison of what you do with what you see can be done was equally said by Newton when he compared himself to a boy who had picked up a few pebbles from the shore ... so that you have respectable authority for supposing that you will never get rid of that feeling; and it is no use trying to catch the horizon [quoted in Hollings et al., 2017, p. 208; original in LB 170, 15 September 1840, f. 14r]

Such metaphors of exploration and colonisation are unsurprising for the time, and sat comfortably with sensibilities of later British mathematicians: Cambridge's G H Hardy, a keen climber himself, and friend of the climbers Mallory and Irving, who were lost attempting to scale Everest, wrote (Hardy, 1929)

I have myself always thought of a mathematician as in the first instance an *observer*, a man who gazes at a distant range of mountains and notes down his observations. His object is simply to distinguish clearly and notify to others as many different peaks as he can. There are some peaks which he can distinguish easily, while others are less clear. He sees A sharply, while of B he can obtain only transitory glimpses. At last he makes out a ridge which leads from A, and following it to its end he discovers that it culminates in B. (Hardy, 1929, p. 18)

Hardy extends this metaphor to reflect on the nature of proof:

B is now fixed in his vision, and from this point he can proceed to further discoveries. In other cases perhaps he can distinguish a ridge which vanishes in the distance, and conjectures that it leads to a peak in the clouds or below the horizon. But when he sees a peak he believes that it is there simply because he sees it. If he wishes someone else to see it, he points to it, either directly or through the chain of summits which led him to recognise it himself. When his pupil also sees it, the research, the argument, the *proof* is finished. (Ibid.)

Hardy's notion of education is striking to the modern reader: the pupil needs to see the result is true, but nothing is said about the process of learning to do proofs.

The analogy is a rough one, but I am sure that it is not altogether misleading. If we were to push it to its extreme we should be led to a rather paradoxical conclusion; that there is, strictly, no such thing as mathematical proof; that we can, in the last analysis, do nothing but point; that proofs are what Littlewood and I call gas, rhetorical flourishes designed to affect psychology, pictures on the board in the lecture, devices to stimulate the imagination of pupils. (Ibid.)

Gilbert Ryle takes the metaphor to the jungle, (Ryle, 1971)

the pioneering path-finder, Pythagoras say, has no tracks to follow ...through the jungle [it may be that].... he will have made a track along which he can now guide docile companions safely and easily right through the jungle. How does he achieve this? Not by following tracks, since there are none to follow. Not by sitting down and wringing his hands. But by walking over ground where tracks certainly do not exist, but where, with luck, assiduity and judgement, tracks might and so perhaps will exist. All his walkings are experimental walkings on hypothetical tracks or candidate-tracks or could-be tracks, or tracks on appro; and it is by so walking that, in the end, while of course he finds lots and lots of impasses, he also finds (if he does find), a viable track. (Ryle, 1971, p. 224)

and the contemporary mathematician and author du Sautoy (2015), channelling Hardy, argues, with a nod to Tolkein, that:

A proof is like the mathematician's travelogue. Fermat gazed out of his mathematical window and spotted this mathematical peak in the distance, the statement that his equations do not have whole number solutions. The challenge for subsequent generations of mathematicians was to find a pathway leading from the familiar territory that mathematicians had already navigated to this foreign new land. Like the story of Frodo's adventures in Tolkien's *Lord of the Rings*, a proof is a description of the journey from the Shire to Mordor.

A successful proof is like a set of signposts that allow all subsequent mathematicians to make the same journey. Readers of the proof will experience the same exciting realisation as its author that this path allows them to reach the distant peak. Very often a proof will not seek to dot every i and cross every t, just as a story does not present every detail of a character's life. It is a description of the journey and not necessarily the re-enactment of every step. The arguments that mathematicians provide as proofs are designed to create a rush in the mind of the reader.

In recent years a number of mathematicians have written accounts of the discipline for the general reader, with the "landscape" metaphor remaining prominent. For 2010 Fields medallist Villani (2015), whose book largely consists of transcriptions of emails sent as his work developed,

The complexity of the mathematical landscape ... makes my head spin (Villani, 2015, p. 80)

and he notes the role of analogy:

The ability to detect connections between different areas of mathematics is what has made my reputation. These connections are invaluable. It is a bit like a game of Ping-Pong: every discovery you make on one side helps you discover something new on the other. The connections make it possible to see more of the landscape on both sides. (Villani, 2015, p. 135)

Harris (2015), in a wide-ranging account of the mathematical process, draws on the more contemporary analogy of video-games, in describing the sense of being an avatar in the virtual world, while John Conway (Roberts, 2015) graphically describes the physicality of the geometrical worlds he is investigating:

For a time I was thinking so geometrically about these things that I used to imagine myself with lots and lots of arms and legs, extra limbs. Because if I have two arms and point 'em out, then they both lie in a plane. And I'll use a leg as well, and now they are lying in three-dimensional space. To form an adequate idea, an adequate geometric visualisation, of what is going on in 24 dimensions is more or less impossible. In large dimensional space, there are large numbers of directions to point, so you would seem to need quite a lot of arms and legs. I distinctly remember imagining myself stuck in the middle of this space, and waving all my arms and legs in the air, and trying to understand things, looking up at the stars, pretending they are the lattice points, and just sort of daydreaming.

3 The Journey in Space: Broader Reflections

Such metaphors have been cast more broadly by authors in philosophy, social science, sociology of knowledge and literary theory. Jenkins (2007) unpicks approaches to understanding and categorisation of the “spaces” and “landscapes” of knowledge, from the Romantic movement in literature onwards. As Jenkins observes,

the same kinds of mental processes that allow us to perceive the organization of a landscape are analogous to the ones that allow us to perceive the organization of a body of knowledge ... strategies used to regulate access to knowledge and to manage the twin pulls towards spread and containment of information. (Jenkins, 2007, p. 7)

These metaphors can be construed as the manifestation of relations of class, power and colonialism:

Some of these strategies involved use of spatial metaphor, for instance by imagining knowledge as a landscape in which certain kinds of journeys and certain kinds of traveller were permitted and others excluded. (Ibid.)

Lane (2017) draws on Bourdieu (1985) and Sewell (1992) in interpreting substantial ethnographic observation to show how mathematical perception is built up from the domains of physical, conceptual and discourse space. Bourdieu sees such categorizations as the manifestations of the social world (Bourdieu, 1985), and Lane argues that the crafting of ideas changes as a function of space, these spaces being socially determined. These domains share schemas which are mobilised during problem solving and proof construction, to guide mathematicians' intuitions; and are utilised during communicative acts, in order to create common ground and common reference frames. Different structuring principles are utilised according to the contexts in which the act of knowledge production or communication take place. Lane argues further that the degree of formality, privacy or competitiveness of environments affects the presentation of mathematicians' selves and ideas, and that mathematicians' perceptions of mathematical phenomena are dependent upon their positions and relations in this “social space”.

Metaphors are important. These mathematicians present mathematics as a fixed Platonic landscape, open to conquest and colonisation by intrepid mathematician-explorers, who show others the paths they have made and the things they have dis-

covered. To join them the pupils too have to be fearless and intrepid: and little is said about how to learn the skills of exploration.

Yet the approach of sociologists opens up the question of how the mathematicians themselves might be creating that landscape, and what skills that might require. It is beyond our scope here to look at broader philosophical issues, for example the contrast between mathematics as theory-building, and mathematics as problem-solving, as in Gowers (2000). But mathematicians themselves can sometimes question if landscape metaphors are too constraining: for example Jim Propp, quoted in (Roberts, 2015), wonders if John Conway:

is the rare sort of mathematician whose ability to connect his pet mathematical interests makes one wonder if he isn't, at some level, shaping mathematical reality and not just exploring it. The example of this that I know best is a connection he discovered between sphere packing and games. These were two separate areas of study that Conway had arrived at by two different paths. So there's no reason for them to be linked. But somehow, through the force of his personality, and the intensity of his passion, he bent the mathematical universe to his will. (Roberts, 2015, p. 2)

The account by Ehrhardt (2010) of the work of Evariste Galois gives a nice example of how such apparently fixed landscapes can be both post hoc constructions, ways of organising knowledge, as indicated by Jenkins; and be shaped by social as well as scientific forces as indicated by Bourdieu and Lane. Galois's work on the solution of equations was dismissed by Poisson and Lacroix, and his contribution was only recognised once later mathematicians had developed the theory of equations in a broader context. As Ehrhardt remarks:

Indeed, the meaning of a mathematical text is the product of a long social and scientific process, one that, in the case of Galois's text, took over one hundred years. During this long period, Galois's text was read, interpreted and recast by a large number of actors who did not agree as to its meaning and mostly construed it through local lenses. Only at the beginning of the 20th century, when Galois theory entered the realm of teaching in European countries, did it acquire a more unified meaning.

A further philosophical aspect beyond our scope is that of the aesthetic of mathematics: the philosopher Thomas (2016) offers a different metaphor for the choices a mathematician makes in shaping the landscape:

Much mathematical effort is more like landscape gardening than like picture drawing. I take picture drawing to begin with a blank sheet ... Mathematical creation is not so free, hence the contrasting analogy of the landscape gardener, who needs a good grasp of the topography before getting down to creating something beautiful (Thomas, 2016, p. 124)

4 Mathematicians on Mathematics: Genius Versus Craft

Yet, while such metaphors of journeys of discovery in a space of mathematical possibilities are widespread, much less attention is paid in such popular writing on mathematics to the practice of mathematics. While Wiles's presentation of the messy

day-to-day business mathematicians' work, above, as "stumbling around in the dark", seems credible, for others there is a more idealistic view. For du Sautoy (2015) such practice is seemingly about a world of choices based on narrative impact:

When I am creating a new piece of mathematics the choices I will make will be motivated by the desire to take my audience on an interesting mathematical journey full of twists and turns and surprises. I want to tease an audience with the challenge of why two seemingly unconnected mathematical characters should have anything to do with each other. And then as the proof unfolds there is a gradual realisation or sudden moment of recognition that these two ideas are actually one and the same character.

Roberts (2015), while acknowledging that her subject is an unreliable narrator, makes light of the labour of one of Conway's major discoveries:

Conway had expected to keep to his house-arrest work ethic for weeks or months or beyond. Locking himself away that first Saturday, he unfurled an unused roll of wallpaper backing paper and sketched out all he knew about the problem. By that very evening, he'd figured it out. He'd deduced the Leech lattice's number of symmetries.

Such unrealistic representations play to the further self-presentation of the mathematician as an idiosyncratic genius possessed of an indefinable charisma: an observant account is presented by Michael Harris in his book *Mathematics without Apologies* (Harris, 2015). Harris frames such mathematical charisma, including his own, by quoting Bourdieu:

The charismatic leader manages to be for the group what he is for himself, instead of being for himself, like those dominated in the symbolic struggle, what he is for others. He 'makes' the opinion which makes him; he constitutes himself as an absolute by a manipulation of symbolic power which is constitutive of his power since it enables him to produce and impose his own objectification. (Bourdieu, 1984, p. 208)

It is beyond our scope to consider in detail how such self-perpetuating acculturation influences all aspects of the doing of mathematics: not just the public perception of mathematics, or how people become mathematicians, or the career and prestige of individuals, but also choice and acceptability of problems, and credibility of proposed proofs.

A striking example is provided by the discussion by leaders of the field, on Frank Calegari's well-respected blog (Calegari, 2017), of the recent claims by Mochizuki to have proved the long-standing ABC conjecture. Mochizuki's claims, because of his previous work, or "charisma", initially carried some credibility. Doubts increased, due to the difficulty in understanding the papers, his disinclination to present the work in public, and reports that it was to be published in a journal of which he is himself the editor. Terence Tao pointed out how unusual it was that the lengthy development did not contain within it a "proof of concept"—a smaller result which would give the reader some confidence in the direction of travel (Tao, 2017)

It seems bizarre to me that there would be an entire self-contained theory whose only external application is to prove the abc conjecture after 300+ pages of set up, with no smaller fragment of this setup having any non-trivial external consequence whatsoever.

Tanswell (2017) proposes framing this debate in terms of the philosophical theory of mathematical “virtues”, identifying a tension between Mochizuki’s defence of the rigour of his collaborators, the expectation of the virtues of significant labour and humility on the part of even his expert readers, and the expectation of those readers of the virtues of transparency and clarity and links to other mathematical material. Tanswell’s “Moderate Proposal” is that virtues and vices of mathematicians are relevant to mathematical knowledge, and virtues, vices and values can be incorporated more generally into philosophy of mathematics. Mason and Hanna (2016) extend this to education, identifying the tension between values of care for students, and care for mathematics, in choices of expository style.

Thus valuable and credible as the reflections of mathematicians are, in shedding light on controversies in mathematics, and how practitioners think about their own discipline, they still tell us less about the “how” of doing mathematics, or of learning how to do mathematics. Indeed, by reinforcing stereotypes of the mathematician as an inspired genius, and mathematics as a competitive sport, they contribute to a perception of mathematical ability as a fixed trait. This view has been strongly challenged by researchers in mathematical education, notably Dweck (2006) and Boaler (2016), who argue that such a “fixed mindset”, seeing mathematical ability as unusual and unchangeable, hampers student learning, and that achievement increases when students shift to a “growth mindset” of believing that their abilities can be developed and their intelligence is malleable.

Terry Tao, a Fields medallist and respected mentor, teacher and mathematical innovator, who was himself a child prodigy, makes similar points. A prolific blogger on education, his 2007 blog post against the notion of genius forcefully presents mathematical ability as a skill to be learned:

Does one have to be a genius to do mathematics?

The answer is an emphatic NO. In order to make good and useful contributions to mathematics, one does need to work hard, learn one’s field well, learn other fields and tools, ask questions, talk to other mathematicians, and think about the “big picture”. And yes, a reasonable amount of intelligence, patience, and maturity is also required. But one does not need some sort of magic “genius gene” that spontaneously generates *ex nihilo* deep insights, unexpected solutions to problems, or other supernatural abilities. (Tao, 2007)

concluding, in the spirit of amassing “capital” in the form of understanding and contributions to a collective effort, but at odds with Harris’s more flamboyant notions of “charisma”, that

It’s also good to remember that professional mathematics is not a sport (in sharp contrast to mathematics competitions). The objective in mathematics is not to obtain the highest ranking, the highest “score”, or the highest number of prizes and awards; instead, it is to increase understanding of mathematics (both for yourself, and for your colleagues and students), and to contribute to its development and applications. For these tasks, mathematics needs all the good people it can get. (Ibid.)

The tools of ethnography, and the emerging field of “mathematical practice”, give a more realistic account of the day to day activities of mathematicians.

Ethnographers observe mathematicians' day to day activity, alone and with others, on notepads and blackboards, as they strive to understand and develop ideas (Barany, 2014):

We call attention to the vast labor of decoding, translating, and transmaterializing official texts without which advanced mathematics could not proceed. ... tentative, transitory marks that try to produce new orders out of old ones (with a crucial stage of disorder in between) (Barany, 2014, p. 108)

This labour, as in the examples above, plays down the notion of genius, replacing it with the idea of detailed skilled work in developing ideas and working out possibilities. Lane (2017) sees the blackboard as a tool for assembling and manipulating mathematical objects: by erasing and “boxing”, drawing arrows and relating, the mathematician is more quickly able to discover patterns and perceive order. The blackboard thus becomes a space for envisioning possibilities and crafting structure, rather than just a space for proving and refining arguments, enabling the exploration of the embodied processes involved in picturing, intuiting and manipulating mathematical spaces. Mathematical practice becomes perceived as a set of skilled actions, habits, and bodily sensations, with the mathematician a craftsperson, skilfully using the physical tools of the mathematician, chalk and blackboard, and the intellectual tools of a variety of mathematical techniques.

This language of craft resonates with many accounts of learning how to do mathematics, by both mathematicians and educators. Polya's famous problem solving techniques (Polya, 1945) are often presented as the “craft of discovery” or similar terms, (Davis, 1995; Zeitz, 2006). Tao's extensive and influential blog posts on learning mathematics (Tao, 2007) resonate with Boaler's work on growth mindset, advising mathematicians to continually refine their craft through mastery of a toolbox of techniques, both developing skill with existing tools, and acquiring new ones.

In the final section we return to writing on craft for a framing of these observations, but first we consider a new area for ethnographic enquiry, the online collaborations known as “polymath”.

5 Crafting Online Collaboration

Tao, with his fellow Fields medallist Tim Gowers and others, is responsible for “polymath”, an endeavour for tackling significant mathematical problems through collective online activity: at the time of writing the sixteenth such project under way. The infrastructure consists solely of postings on a blog, with “house rules” (Gowers, 2009), established through collective discussion, designed to encourage interaction, accessibility and rapid exchange of informal ideas. These include, for example “It's OK for a mathematical thought to be tentative, incomplete, or even incorrect”, “better to have had five stupid ideas than no ideas at all” and “An ideal polymath research comment should represent a ‘quantum of progress’.”

Polymath is sometimes described as “crowdsourced science”, though the crowd is a small and expert one. The unstructured development of the ideas through complex threading of multiple blog comments allows a variety of perspectives and serendipitous connections, with lines of enquiry, some fruitful, some not, weaving together in a manner much more akin to a novel than a conventional scientific paper.

As Gowers and Nielsen observe (Gowers, 2009)

Who would have guessed that the working record of a mathematical project would read like a thriller? (Gowers, 2009, p. 880)

Polymath is an example of a social machine, a concept due to Berners-Lee, defined as “purposeful human interaction on the web”, where machines enable mass human collaboration, rather than acting as mechanical problem solving agents. Social machines cover phenomena as diverse as Wikipedia, twitter, or Zooniverse, and this enmeshed nature of contributor threads, allowing serendipitous interactions, has been identified as a powerful element of their success.

Polymath conversations rapidly become too unwieldy and interwoven to self-organise: a leader draws together the threads from time to time, suggesting the most appropriate next direction, and restarting the discussion with a substantial new blog post. These posts, presented in a more conventional mathematical style, then form the basis of the eventual published paper. Though all polymath projects seem to have produced something useful, not all have proved their target result, with attempts failing through finding a counter-example, or for of lack of participants or fruitful ideas. The most successful have led to published papers, under the pseudonym “D H J Polymath”, where the initials refer to the Density Hales-Jewett theorem, which was the subject of the first Polymath. Participants themselves (Polymath, 2014) are aware of the complex social space thus created, for example reflecting on the opportunities and risks of collaboration behind a pseudonym, rather than a more modest sole contribution.

A recent book by Neale (2017) presents the most notable polymath to date, which extended work of Yitang Zhang to massively reduce the bound on the so called “Twin Primes conjecture”. Neale’s book starts in a landscape, not Hardy’s distant views of lofty peaks, but hands-on climbing in the manner of Ryle’s jungle walks:

You stand looking at the sheer surface of your mathematical problem, searching for toeholds and crevices that might give a way up. After a long time looking, you start to make out an indistinct crack to the left, and a slight pattern in the rock up and to the right that reminds you of a climb you heard about once. Putting together all the features you’ve noticed, you can sketch out a possible route up the rock face, although it’s not quite clear whether that small ledge will make a good toehold and there’s a pretty ambitious reach near the top that might well be a stretch too far.

Still, now that you have a possible route in mind, you can step off the ground, and hope that the details will become clearer along the way. Perhaps that reach will be too big, but when you get a bit closer maybe there’ll be a crack in just the right place for your fingers.

Unfortunately, when you’re three-quarters of the way up a sliver of rock breaks away, your toehold disappears from beneath your feet, and you drop back some way. Eventually, however, if you persevere you might reach the top. (Neale, 2017, p. 1)

The polymath blogs display mathematical proofs, and attempts at proofs, in exactly this fashion: discussion of possible partial approaches, working through the details, resolving bottle-necks and retreating from dead-ends, perhaps by refining current techniques, perhaps by trying something new, and vividly demonstrating how advance may come from sharing and refinement of small insights, as well as from one big breakthrough. The contrast with Hardy's view of an educator's exposition of a completed proof, "rhetorical flourishes designed to affect psychology" is striking.

By contrast with an isolated researcher working on one idea at a time, a polymath project can pursue several lines of attack simultaneously, adding to the potential for fruitful interaction at the cost of greater attention to the ideas of others. We have discussed elsewhere (Martin, 2015) how polymath, and similar mathematical social machines, shed light on the everyday practice of mathematics. We highlighted how few of the blog comments are actual steps in the final proof, with other phenomena such as examples, conjectures, concept formation, and planning, playing key roles in exploring the landscape, and indicated the importance of dead ends and mistakes in increasing understanding, and the value of collaboration in providing diverse skills, capturing mistakes and allowing more risks to be taken.

Lakatos's (1976) account of the development of proofs, presented in an educational framework, seems a much tidier view, in which every action has a clear logical role in the development of the final proof, presented using a theory of responses to counterexamples. However Lakatos was providing a rational reconstruction, and, just as in an account of a successful rock-climb, pruning some of the dead-ends and abandoned lines of enquiry makes for greater readability without altering the purpose of the narrative.

The developers of "polymath" were motivated not just by finding new ways of to solve problems, but also by a strong interest in mathematical education, and in showing their readers, far more clearly than in a standard textbook or lecture, the messy day to day process of doing mathematics, as well as the final proof that emerged. Originally they had hoped to encourage newcomers to take part, an aim not entirely realised, as taking part required a level of specialist knowledge, a commitment of time, and a willingness to make mistakes in public. However, the educational value of polymath is undisputed, in showing, as Tao put it "how the sausage is made", with educators following the proofs as they developed, enabling students to see the sheer excitement of doing mathematics, as well as seeing that even top mathematicians get stuck, make mistakes and need to ask for help (Martin, 2015). The MIT-based "crowdmath" project follows the model of polymath, providing structured and mentored environment for high-school and college students to collaborate on research level problems, and has led to several published research papers (Crowdmath, 2015). Other online experiments, in which technology is used to enhance learning outcomes include an experimental MOOC developed by Boaler et al., designed to encourage participation, interaction, and a move to a "growth mindset". It has attracted over 160,000 participants, and has been used to demonstrate correlations between this intervention and both academic achievement and attitudes towards mathematics.

6 Mathematics and Craft

Craft has a scholarly literature of its own, and in this final section we reflect on what it might contribute to understanding the practice of mathematics. Heidegger (1962) characterised crafting as an embodied process of bringing objects/concepts/structures into being in the world, a form of skilled work by which such objects/concepts/structures are built up dynamically through encounters between the subject of the craft-person and the object which is being crafted. He argued that the distinction between subject and object is dissolved through the process of crafting, as the thing being materialised is imbued with the character and will of the craft-person.

This concept of craft is closely linked to Levi-Strauss's (1966) idea of bricolage (assembly, or making). Bricolage is undertaken by a bricoleur, who assembles diverse objects together into a coherent assemblage, through uniting material objects within the framework of an idea: what transforms the material assemblage of a bricolage from a mess into a craft-work is not the identification of each of the elements as isolated wholes, but rather the higher conceptual structure within which these elements are related, as part of an intentional composition by the craft-person. Mackenzie (2003) introduces the idea of mathematical practice as bricolage, which he characterises as "creative tinkering" guided by broader principles, in his work on the creation the Black-Scholes equation. In a mathematical proof it is not the individual elements which give insight, but rather their relationships within a wider discourse structure, which orients them towards a certain purpose.

As Ingold, in his work on "making" (Ingold, 2011) indicates, a deeper history of craft stretches back to the classical era, where craft or "practice" (*technê*) is contrasted with knowledge or "theory" (*Epistêmê*). Craft is concerned with skills or practices, obtained through apprenticeship with a master craft-person, with such skills developed through practice, so they become a form of know-how or embodied knowledge and habit. In the case of mathematics, the mathematician directs their craft skills to the goal of understanding and manipulating mathematical objects. Tanswell's thesis (Tanswell, 2017) develops this in a discussion of Ryle's distinction between knowing-how and knowing-that, showing that both are necessary, and intertwined in the process of doing mathematics.

David Pye, a furniture maker and eminent scholar of craft, characterises craft as (Pye, 1968, p. 20)

simply any kind of technique or apparatus, in which the quality of the result is not predetermined, but depends on judgment, dexterity and care which the maker exercises as he works. The essential idea is that the quality of the result is continually at risk during the process of making; and so I shall call this kind of workmanship "The workmanship of risk"

He contrasts this with the "workmanship of certainty" where every step and hence the outcome is prescribed, leaving no decisions to the maker, and observes the need for both.

Much writing on craft is concerned with the physicality of tool use, the numerous small choices made in controlling a saw for example. In developing a mathemati-

cal proof the “tools” might be techniques or approaches: “find a minimum”, “look for a bound” and so on, each requiring its own skill in application – the process of “stumbling around in the dark” so articulately described by Wiles. Processes like Wiles’s “a little more calculation” are routine and certain in their outcome, the “workmanship of certainty”, whereas “modifying things a bit” is more akin to the “workmanship of risk”. A polymath proof development shows exactly the choices and refinements being made, as participants debate the choice of different “tools” at each stage, mitigating the “risk” by having others check or comment on their work, and sharing out the routine labour which has more “certainty”, for example doing a routine calculation.

Ingold (2011) gives a close description of using a saw to illustrate the precessional quality of tool use, where precise phases are not delineated, but each contains the seeds of the next as part of an overall “umbrella plan”, a notion similar to Alan Bundy’s proof plans (Bundy, 1988), and characterises the essence of skill in such activities as “the improvisational ability of practitioners to disassemble the constructions of technology and creatively to incorporate the pieces”. Ingold, and we recall here Wiles, Hardy, Ryle, du Sautoy and Neale, compares the activity to a journey:

It does not take just one step, however, to saw a plank. It takes many steps; moreover these steps are no more discrete or discontinuous than those of the walker. That is to say, they do not follow one another in succession, like beads on a string. Their order is precessional, rather than successional. In walking, every step is a development of the one before and a preparation for the one following. The same is true of every stroke of the saw. Like going for a walk, sawing a plank has the character of a journey, (Ingold, 2011, p. 53)

Ingold thinks of the craftsman as a “wayfarer”, and Murray Rust and others (2015) have identified this wayfaring as characteristic of crafting a path through the landscape of a social machine, like polymath, in terms that nicely fit Neale’s climber:

a journeyer situated in a landscape, with signs which can be read, and possible directions to explore. Rather than a top-down map of the world, on which routes can be meticulously planned out, navigation is local and responsive. The wayfarer is engaged in a constant exchange with their environment, deciphering, orienting and acting. (Murray-Rust, 2015, p. 1144)

Ingold uses the term “meshwork” for the collection of paths taken, offering signs to the wayfarer, and acting as records of their passage; such paths are not a well-organised network, but in the entanglings offer new creative possibilities, much like Ryle’s “ground where tracks certainly do not exist, but where, with luck, assiduity and judgement, tracks might and so perhaps will exist”, which others can then follow, or the multiple paths through a space of mathematical possibilities.

Ingold, like du Sautoy, identifies such journeys with stories:

landscape tells – or rather is – a story. ... To perceive the landscape is therefore to carry out an act of remembrance, and remembering is not so much a matter of calling up an internal image, stored in the mind, as of engaging perceptually with an environment that is itself pregnant with the past. (Ingold, 1993, p. 189)

We have but scratched the surface in this essay, and philosophers, ethnographers, social scientists, humanists, educators and scholars of craft have much to say about matters we have left unaddressed.

In our reading, Newton's beach or Hardy's Himalaya or Ryle's jungle or Wiles's cellar or Neale's cliff-face or Thomas's garden are comprehended and communicated as wayfarings in landscapes. These landscapes, as articulated by Jenkins, are metaphors for mathematicians' internal representations, themselves made of a collection proof attempts/journeys/stories, and constantly reshaped through their own new proof attempts/journeys/stories, and through learning of those of others. Ryle's distinction between knowing-what and knowing-how becomes a matter of degree rather than a matter of kind: in the most general terms Hardy surveying the distant peaks knows that there is a proof/route and convinces others, and Neale, scrambling up the rock-face, knows how to enact it. But Hardy sometimes gets his fingernails dirty, and Neale sometimes draws back and inspects the route.

For the educator, the view of mathematics as a craft activity, and of mathematical ability as a skill to be developed, rather than a fixed talent, is not new, and Tao's emphasis on continually extending one's knowledge and skills is a good antidote to unrealistic notions of genius. Activities like polymath allow learners to better understand and learn the craft of how mathematics is done through seeing others exercising those craft skills, and offer an opportunity to develop their own skills by taking part.

The knowledge and skills of both learners and established mathematicians is continually moderated by their own journeys and those of others, raising further questions as to the nature of this mathematical material that is being crafted, and in turn crafting its crafters, the mathematicians. Yet what material is as vital as mathematics, in its ability to affect change on the world, and to push back on the hands and minds of practitioners?

Acknowledgements We thank Dave de Roure and Pip Willcocks for helpful discussions, and the referees for their thoughtful comments. Support from the UK Engineering and Physical Sciences Research Council is acknowledged under grants EPSRC EP/K040251/2 (Martin, Lane, Tanswell), EP/J017728/2 (Murray-Rust) and EP/P017320/1 (Pease).

References

- Barany, M., & Mackenzie, D. (2014). Chalk: Materials and concepts in mathematics research. In *Representation in scientific practice revisited* (pp. 107–130). MIT Press.
- Boaler, J. (2016). *Mathematical mindsets: Unleashing students' potential through creative math, inspiring messages and innovative teaching*. San Francisco, CA: Wiley & Sons.
- Bourdieu, P. (1984). *Distinction: A social critique of the judgement of taste*, tr. Richard Nice. Harvard University Press.
- Bourdieu, P. (1985). The social space and the genesis of groups. *Theory and Society*, 14, 723–744.
- Bundy, A. (1988). The use of explicit plans to guide inductive proofs. In *International Conference on Automated Deduction*.

- Calegari, F. (2017). galoisrepresentations.wordpress.com/2017/12/17/the-abc-conjecture-has-still-not-been-proved/.
- Crowdmath. (2015). <https://artofproblemsolving.com/polymath>.
- Davis, P. J. & Hersh, R. (1995). *The mathematical experience* (study ed.). Birkhauser.
- De Morgan, A. (1842). *The differential and integral calculus* (p. vii). Baldwin and Cradock.
- du Sautoy, M. (2015). How mathematicians are storytellers and numbers are the characters. *The Guardian*. www.theguardian.com/books/2015/jan/23/mathematicians-storytellers-numbers-characters-marcus-du-sautoy.
- Dweck, C. S. (2006). *Mindset: The new psychology of success*. New York, NY: Random House Incorporated.
- Ehrhardt, C. (2010). A social history of the “Galois Affair” at the Paris academy of sciences. *Science in Context*, 23(1), 91–119.
- Gowers, T. (2000). The two cultures of mathematics. In V.I. Arnold (Ed.), *Mathematics: Frontiers and perspectives*. AMS. <https://www.dpmms.cam.ac.uk/~wtg10/2cultures.pdf>.
- Gowers, T. (2009). <https://www.gowers.wordpress.com/2009/01/27/ismassively-collaborative-mathematicspossible/>. Gowers, T., & Nielsen, M. (2009). Massively collaborative mathematics. *Nature*, 461, 879–881.
- Hardy, G. H. (1929). *Mathematical proof*. Mind.
- Harris, M. (2015). *Mathematics without apologies*. Princeton.
- Heidigger, M. (1962). *Being and time*, tr J. Macquarrie & E. Robinson. Harper & Row.
- Hollings, C., Martin, U., & Rice, A. (2017). The Lovelace–De Morgan mathematical correspondence: A critical re-appraisal. *Historia Mathematica*, 44(3), 202–231.
- Ingold, T. (1993). The temporality of the landscape. *World Archaeology*, 152–174.
- Ingold, T. (2011). *Being alive: Essays on movement, knowledge and description*. Taylor & Francis.
- Jenkins, A. (2007). *Space and the ‘March of Mind’: Literature and the physical sciences in Britain 1815–1850*. OUP.
- Lakatos, I. (1976). *Proofs and refutations*. Cambridge: Cambridge University Press.
- Lane, L. (2017). *The bridge between worlds: Relating position and Ddsposition in the mathematical field*. Ph.D. thesis, University of Edinburgh
- Levi-Strauss, C. (1966). *The savage mind*. University of Chicago Press.
- Mackenzie, D. (2003). An equation and its worlds: Bricolage, exemplars, disunity and performativity in financial economics. *Social Studies of Science*, 33, 831–868.
- Martin, U. (2015). Stumbling around in the dark: Lessons from everyday mathematics. In A. P. Felty & A. Middeldorp (Eds.), *Proceedings of CADE-25. Lecture Notes in Artificial Intelligence* (Vol. 9195). Springer.
- Mason J., & Hanna, G. (2016). Values in caring for proof. In B. Larvor (Ed.), *Mathematical cultures*. Springer Trends in the History of Science.
- Murray-Rust, D., et al. (2015). On wayfaring in social machines. In *Proceedings of the 24th International Conference on the World Wide Web* (pp. 1143–1148).
- Neale, V. (2017). *Closing the gap, the quest to understand prime numbers*. Oxford University Press.
- Polya, G. (1945). *How to solve it*. Princeton University Press.
- Polymath, D. H. J. (2014). The ‘bounded gaps between primes’ polymath project: A retrospective analysis. *Newsletter of the European Mathematical Society*, 94, 13–23.
- Pye, D. (1968). *The nature and art of workmanship*. Cambridge University Press.
- Roberts, S. (2015). John Horton Conway, the world’s most charismatic mathematician. *The Guardian*. www.theguardian.com/science/2015/jul/23/john-horton-conway-the-most-charismatic-mathematician-in-the-world.
- Robinson, A. (2011). *Genius, a very short introduction*. Oxford University Press.
- Ryle, G. (1971). Thinking and self-teaching. *Journal of Philosophy of Education*, 5, 216–228.
- Simon, S. (2009). Newton on the beach: The information order of *Principia Mathematica*. *History of Science*, xlv. Science History Publications Ltd.
- Sewell, W. H. (1992). A theory of structure: Duality, agency, and transformation. *American Journal of Sociology*, 98, 1–29.

- Tanswell, F. (2017). *Proof, rigour and informality: A virtue account of mathematical knowledge*. Ph.D. thesis, University of St Andrews.
- Tao, T. (2007). Does one have to be a genius to do maths? terrytao.wordpress.com/career-advice/does-one-have-to-be-a-genius-to-do-maths/.
- Tao, T. (2017). Blog comment on “The ABC conjecture has (still) not been proved”. galoisrepresentations.wordpress.com/2017/12/17/the-abc-conjecture-has-still-not-been-proved/#comment-4563.
- Thomas, R. (2016). Beauty is not all there is to aesthetics in mathematics. *Philosophia Mathematica*, 25, 116–127.
- Turnor, E., *Collections for the history of the town and soke of Grantham* (London, 1806, Vol. 173, No. 2), where it is claimed this was said by Newton “a little before his death”.
- Villani, C. (2015). *Birth of a theorem: A mathematical adventure*. Farrar, Straus and Giroux.
- Wiles, A. (2000). Transcription of interview by PBS. www.pbs.org/wgbh/nova/physics/andrew-wiles-fermat.html.
- Zeitz, P. (2006). *The art and craft of problem solving*. Wiley.

Suggestions for the Use of Proof Software in the Classroom

Using Automated Reasoning Tools to Explore Geometric Statements and Conjectures



Markus Hohenwarter, Zoltán Kovács and Tomás Recio

1 Introduction

GeoGebra, a software tool for dynamic mathematics, has recently been enhanced with an automated reasoning subsystem able to confirm/deny the truth of any geometry statement displayed on the screen. In addition, if the statement is labeled as *false*, *GeoGebra* can exhibit required modifications to the hypotheses that make the statement true. The free availability and portability of *GeoGebra* have made it possible for millions of students worldwide to harness these novel techniques on tablets, smartphones, and computers.

The mathematical background of this reasoning method—based on automatically algebraizing a given geometric statement and associated construction and then applying effective algebraic geometry tools—goes back to the work of Wen-Tsün Wu in the 1970s (see Wu, 1978; Chou, 1987). Wu's highly performing approach was the starting point for the developing and implementing different algebraic geometry based, automated reasoning algorithms in a large collection of programs. However, there has never been a program as effective as *GeoGebra* in:

- merging dynamic geometry and computer algebra,
- addressing non-experts, and
- achieving an impact in the educational community worldwide.

M. Hohenwarter
Johannes Kepler University of Linz, Linz, Austria
e-mail: markus.hohenwarter@jku.at

Z. Kovács (✉)
The Private University College of Education of the Diocese of Linz, Linz, Austria
e-mail: zoltan@geogebra.org

T. Recio
University of Cantabria, Santander, Spain
e-mail: tomas.recio@unican.es

In this chapter we propose the exploration of a new kind of workflow in the teaching/learning of elementary geometry with the help of the GeoGebra Automated Reasoning Tool (ART). Let us remark that the very recent launching of this tool makes our work a proposal not yet supported by experience. However, as we summarize below and argue in more detail in the conclusion, we think it is important to reflect on the potential uses and misuses of these tools in education early on.

It is well known that dynamic geometry systems (DGS), even without these specific features, can be useful (as well as challenging) tools in the teaching/learning of reasoning and proof. DGS allow students to formulate certain geometric facts (e.g., as intermediate steps in establishing the proof of a given statement) by drawing auxiliary diagrams and then getting convinced of the truth/falsity of the proposed assertion by checking its validity, in many instances, after randomly dragging some elements of the figure (see Hohenwarter, Kovács, & Recio, 2017).

Moreover, DGS can help students making conjectures about a certain construction by allowing them to drag some of its free objects and observe the behaviour of those that depend on them. This dynamic visualization could enhance approaches listed in Polya (1962) such as the “pattern of two loci”—one of Polya’s four “patterns of thought.” An even simpler example of the potential relevance of dragging and observing could be the study of the relation between segments AP and BP while P moves in a circle with diameter AB .

A more sophisticated context of geometric reasoning involving DGS basic features arises when the user does backward conjecturing; that is, if the user makes, first, a certain construction and then states some property that he/she wants to hold true over the resulting figure. Obviously it does not hold true in general and the user has to look for necessary changes in the construction. Dynamic geometry programs facilitate the dragging of free objects and therefore the possibility of conjecturing how to restrict the position of these objects for the desired property to emerge. For example, consider a segment AB and a free point P . The user might want to find where to place P so that AP , BP are perpendicular. By measuring the angle $\angle APB$ and tracing the movements of P so the angle keeps close to 90° , the user can conjecture that P should be placed in a circle with diameter AB .

All of these useful features—and their associated impact in the classroom—belong, in some sense, to the past. Now, with GeoGebra ART, we can go much further—both for conjecturing that some property actually happens in a given construction and for discovering how to modify the construction for a given property to hold. A first novelty is that GeoGebra can take the lead and formulate the conjectures by itself. Thus, the program can, by simply comparing some geometric objects in a particular instance of a construction by means of the *Relation* tool, suggest a property holding between these elements as a conjectural truth. For example, in the above example with P located in a circle of diameter AB , we can directly ask GeoGebra for some relation between segments AP , BP produced by a concrete position of P in the circle, yielding, as output, that for this particular instance, and computing with the numerical coordinates of A , B , P , the two segments seem to be perpendicular.

A second improvement is that, in this context, GeoGebra can automatically verify the general validity of this conjecture. In the affirmative case, it uses computation

with symbolic coordinates to establish a mathematical proof; that is, a proof not based on visual verification, approximate numerical calculations, or probabilistic analysis. To avoid confusion, the proof is performed in the program background and not shown to the user since it could fill hundreds of pages of algebraic formulas. Eventually, a message highlighting the need to avoid some degenerate cases for the truth of the given statement appears on the screen.

If the answer to the conjecture is negative and the user insists on requiring that the established conjecture holds, he/she can ask GeoGebra via the *LocusEquation* command for the precise formulation of necessary changes to the statement hypotheses.

We situate this workflow in a technology-mediated paradigm where the machine behaves like a mentor in the learning process, helping students in the intermediate steps of developing their own explanations of the truth of some geometric facts and fostering a *creativity spiral* as new discoveries are made by restarting the workflow again and again.

In the next section, we present a short overview of maths apps and programs related to dynamic geometry and theorem proving. We also give a basic description of the mathematical algorithm and system requirements behind our GeoGebra ART. In section three, we explore the use of these tools in education to help students discover geometric facts both experimentally and by proving. Section four focuses on a particularly novel feature of ART—the automatic discovery of missing hypotheses for a given (false) statement to become true. We also emphasize how this feature can help students “create” interesting, new geometric statements. Finally, section five collects some arguments and conclusions on the possible effect of GeoGebra ART in the learning and teaching of geometry.

2 Currently Available Maths Apps: A Short Overview

No doubt, there is a growing interest in developing convenient maths applications for contemporary students who are very experienced in using new technology, including smartphones and tablets. One of the most vibrant places to experiment with what the new technology offers is the market of Android applications, in particular those that are freely available and at the disposal of a very large set of users. Here we refer to a recent survey by Corpuz (2017) that summarizes some of the most popular Android and iOS apps as of October 2017. Corpuz’s collection consists of flexible scientific calculators for special purposes such as scientific or financial inputs, equation solvers, graphing calculators, and computational knowledge engines. Some of these popular software tools support handwriting recognition and/or capturing and interpreting camera shots of handwritten/printed formulas. On the other hand,

... when it comes to mathematics, it isn’t just getting the final answer that’s important; if anything, correct step-by-step solutions are far more important when it comes to teaching and learning math. (Corpuz, 2017, slide 9)

In this context, let us remark that one of the examples in the survey, the *Mathway* app, is able to provide the user not only with the answer to a problem (mainly in Linear Algebra, Calculus, or Trigonometry, introduced by typing or scanning handwritten input via the smartphone camera) but also with each step of the solution.

Photomath is not listed in Corpuz's collection but was the most popular maths app on *Google Play* in March 2018. *Photomath* provides step-by-step solutions to a comparable range of problems to the *Mathway* app free of charge in 36 languages.

It is, however, incautious to conclude that these examples represent a real change in the educational paradigm, opening the door to a new approach based on *computer-mediated thinking*. In fact, these apps are just enlarged pocket calculators that closely follow school traditions. Some of the answers they provide, such as the detailed steps in solving a linear univariate equation, are a usual requirement in the school curriculum in many countries.

On the other hand, step-by-step solvers for other fields of the maths curriculum (such as geometry) are not yet so popular. There are, however, remarkable attempts in some other areas such as:

- *Edukera* (www.app.edukera.com) that teaches step-by-step proving in the fields of logic, sets, calculus, and analysis,
- *Euclidea* (www.euclidea.xyz) that teaches Euclidean geometry constructions as puzzles (120 puzzles are provided), with some basic dynamic geometry features.

These software tools are, however, only useful for finding solutions to pre-programmed problems. Even though the input seems to be a wide-open set of formulas, they are limited to solving a concrete set of close-ended problems. Thus, what they offer for a substantial part of mathematical activity—namely, for *discovery*—is very limited. Of course for most learners, these wonderful pieces of software are still inexhaustible, and their mathematical activity indeed simulates a kind of endless discovery.

In the following, we focus on a radically different approach. In our proposal, discovery plays a key role and the questions being posed challenge not only the underlying software, but also the user—both human and machine collaborating in the learning of something new!

Let us quickly recall that automated deduction of known or not yet discovered geometric results has a wide literature going back to the appearance of the first computers (see Botana et al., 2015 for a detailed overview). In particular, in planar Euclidean geometry, the first successful attempts in the 1950s (see Gelernter, 1959) led to a line of work in Formal Logic and Artificial Intelligence. As mentioned above, another important approach was based on Algebraic Geometry methods and was started by Wu (1978) and his early followers, including Chou (1987), and—focusing on the Gröbner bases method—Kapur (1986), and Kutzler and Stifter (1986), among others.

This is precisely the method used in the current GeoGebra implementation and that we roughly describe as follows: first, geometric statements are internally translated in terms of algebraic equations. For example, assume that the translation of the

hypotheses H and the thesis T of a given statement $\{H \Rightarrow T\}$ is a collection of polynomial equations $H = \{h_1 = 0, \dots, h_r = 0\}$ (respectively $T = \{f = 0\}$). Then, the geometric instances verifying the hypotheses (respectively the thesis) are identified with the solution set of the corresponding polynomial system. And, if a non-empty and large subset (technically speaking: if a Zariski-open subset) of the algebraic variety of solutions of H is contained in the solution set of T , the theorem is labelled as *generally true*. This key inclusion test is performed by using Elimination procedures with Gröbner basis and checking whether the result is zero or not (see Recio & Velez, 1999 for further technical details).

DGS became very popular and well known with the breakthrough of personal home computers. Beginning with the *Geometric Supposer* (Schwartz & Yerushalmy, 1983) in 1981, widely used tools such as *The Geometer's Sketchpad* (Jackiw, 1995), *Cabri Geometry* (Baulac, Bellemain, & Laborde, 1994), and *Cinderella* (Kortenkamp, 1999) were available commercially. Another breakthrough, *GeoGebra's* (Hohenwarter, 2002) free availability for millions of users, opened the road to considering dynamic geometry as a natural education tool in the classroom.

Combining automated deduction in geometry (ADG) and DGS was a somewhat newer topic, but was already present in the 1990s as more than a research concept in the first versions of several DGS (de Villiers, 1999). These pioneer approaches, however, mostly used numerical and statistical methods for verifying properties holding among elements of a geometric figure. Just a few years later, in the second half of the 1990s, software tools appeared that used pure symbolic methods to prove or visualize geometric facts (see Botana & Valcarce, 2002; Oldenburg, 2008; Ye, Chou, & Gao, 2011).

We emphasize that a further requirement was essential to making substantial improvements in harnessing the possibilities of DGS; namely, accessing symbolic computations reliably and quickly, and connecting them dynamically with geometric objects. This is now possible in GeoGebra thanks to Giac (Kovács & Parrisé, 2015). GeoGebra is currently able to perform symbolic computations that allow the user to manipulate (simplify, expand, etc.) algebraic expressions in a way as rigorous as any well known computer algebra system.¹

Finally, the application of these symbolic computations tools to automatic reasoning in geometry in GeoGebra was initiated by Recio, Botana and Abánades in 2010, and continued by many others, including Kovács, Weitzhofer, Parrisé, and Sólyom-Gecse over the last years (for more precise technical information and further references see Botana et al., 2015; Kovács & Parrisé, 2015). The outcome of this work, *GeoGebra ART*, will be discussed in the following sections.

¹https://en.wikipedia.org/wiki/Computer_algebra_system.

3 Discovery and Creativity

Corless (2004) describes *computer-mediated thinking* by citing Peter Jones' notion of "intelligent partnership" (Jones, 1996) between the student and the computer (see Martinovic, Muller, & Buteau, 2013 for a detailed description). Corless mentions several examples of significant computational results achieved during this fruitful partnership. They are not only non-trivial for undergraduate students, but also surprising for many researchers, too. In fact, as Corless recalls by quoting the Portuguese Jewish philosopher, Abarbanel, mathematical discovery *should be a surprising activity*:

"I have absolutely no interest in proving things I know are true." (Corless, 2004, p. 10)

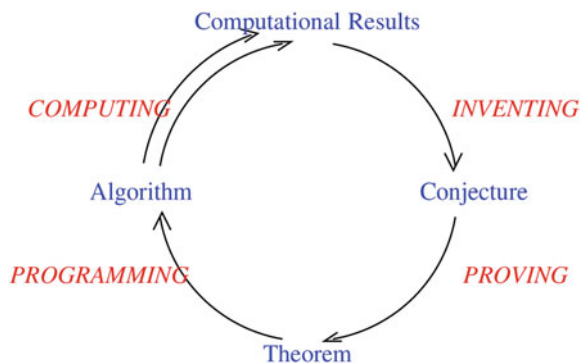
The relevance of student-computer cooperation in apprehending mathematical ideas through discovery is also emphasized in Buchberger's graphic *creativity spiral* describing the learner's workflow (Buchberger & The Theorema Working Group, 1998, see Fig. 1).

According to this concept, a continuous workflow can be identified starting with *computational results*. These results lead by intuition (or nowadays we could say by "big data" mining) to the invention of new *conjectures* that, in turn, yield to the formulation and eventually the proof of new *theorems*. From there, as Buchberger emphasizes, these results could lead to new *algorithms* and, by programming these algorithms, to new *computational results* about some mathematical fact. The spiral then continues with further inventions and conjectures (see also Kovács, 2018).

Of course, this process describes not only the learner's attitude to knowing mathematics better, but the researcher's position as well. In fact, both could be considered as quite similar, if we recall Halmos' idea (Halmos, 1982, p. vii) that "the only way to learn mathematics is to do mathematics."

Let us emphasize that both Corless' notion and Buchberger's spiral assume that the discovery process requires a computer. Thus, in the following, we will apply these basic principles of discovering geometric facts to the mediation of DGS. Also, we suppose—following the Jaworski's concept on the *teaching triad*: management of

Fig. 1 Buchberger's concept, the creativity spiral



learning, sensitivity to students, and mathematical challenge (Jaworski, 1994)—that the related learning process will involve the presence of a human *mentor* as well. The mentor serves as a “personal coach and influencer” who attempts to affect the way a student behaves during a geometric task by suggesting concrete activities for finding the solution. On the other hand, as with sports coaches, we think it is useful to minimize the role of the teacher in the discovery process. The role of teachers in this new context is a delicate issue that needs precise research. One example of such research (described in Martinovic & Manizade, 2014) shows future teachers exploring activities designed to develop both technological and geometric skills.

We summarize our proposal for a novel approach to geometry learning through the use of the newly implemented ART in GeoGebra (see Kovács, Recio, & Vélez, 2017), as follows:


1. (a) The teacher provides the students with a kind of demo or tutorial on the use of DGS automated reasoning tools.
(b) The teacher poses a problem. The nature of the problem is an open-ended question (in the sense of Foster, 2013) like “find all points P in the plane that have a certain property” (an “implicit locus problem”). More generally, we are thinking of questions for which the student has not been taught to follow a predetermined path for finding a solution. Of course, this is a very subjective situation. Using computers to “discover” shows that there is indeed a predetermined path to success, but not for the human user!
2. Some *computations* are performed with the DGS, based on a construction made by the student. In many cases this results in using the software tool for either random experiments or for experiments planned by the teacher. As an example, we will focus on computing a particular implicit locus (say, a circle).
3. A *conjecture* describing the characteristics of the output curve (e.g., a circle going through the three vertices of a given triangle) is made by the student.
4. The conjecture is checked numerically and symbolically by the DGS. We accept this result without further verification as a basic step. We now have a *theorem*. If applicable, the proof can be worked out by paper and pencil as well.
5. The next activity suggested by Buchberger’s spiral—the “programming” step—can be interpreted here as using the obtained result as a “piece” towards achieving more involved statements. These new pieces can then be assembled into new *algorithms*.

In this way, the obtained theorems and algorithms could be considered as a mere step towards the design and execution of “further experiments” by the student, whether controlled by the teacher or not, involving new *computations* with DGS (Step 1, second round). The process of Buchberger’s spiral then continues with Step 2.

4 An Example: Implicit Loci in GeoGebra

Using the steps above we now give some examples of how GeoGebra ART supports the discovery of an implicit locus. Let us consider the following simple implicit locus problem: *Given a line segment, describe the position of all points that are equidistant from the segment's endpoints.*

The solution to this basic introductory question is well known, both theoretically and through exploration with a DGS. In fact, this “direct” locus problem can be considered as an elementary instance of Voronoi diagrams, well known for their manifold real-life applications (Lindenbauer & Reichenberger, 2015). Yet, the teacher may propose this problem in an open-ended form (Step 0 (b)).

For the student, Step 1 may be to draw a triangle in GeoGebra by using the *Polygon* tool . By default, triangle ABC with sides a , b and c is created (here and in the rest of the paper we assume that sides a , b and c are opposite vertices A , B and C respectively). Now the student can consider points A and B as fixed during the observation, and C as arbitrary and freely draggable. In the *Algebra View* it can be observed how side lengths a and b change when point C is dragged.

After collecting the results for a sufficiently high number of positions of C , a conjecture about the locus of C can be made. For some students this will be easy, but for others, probably not. To help the latter, additional colouring of the trace of C when a and b are close to each other could be enabled (Fig. 2, see Losada, 2014 for more details). This visual help may be unnecessary in this simple example, but for more difficult questions it could be quite relevant (see, e.g., Losada, Recio, & Valcarce, 2011).

Alternatively, GeoGebra can numerically compute the solution for the particular case when A and B are fixed. The student just needs to enter the command `LocusEquation(a==b,C)`² to obtain the algebraic equation of the resulting geometric curve; that is, the bisector line of segment AB , with the equation $-8800x + 250y = -8607$. Figure 3 shows the result.

Of course, learners need to have some minimal knowledge of analytic geometry to be able to recognize that the obtained curve is indeed a line. This is, however, not strictly required. The conjecture could instead be formulated by simply observing the visual output of the locus in the *Graphics View*.

To confirm this, the student will also want to drag the points A or B . This could generalize the obtained result by creating several other setups of the input points. Figure 4 shows a few particular cases where A was moved from the original position to the position $(-2, 3)$. Of course, tracing should not be enabled in this step. It was done here to demonstrate the number of video frames shown during a short dragging interval.

²Notice that, according to the syntax of GeoGebra, the equation sign must be entered *twice*; this information is available in the GeoGebra Help or in Kovács (2017), but should also be remarked by the teacher in Step 0 (a).

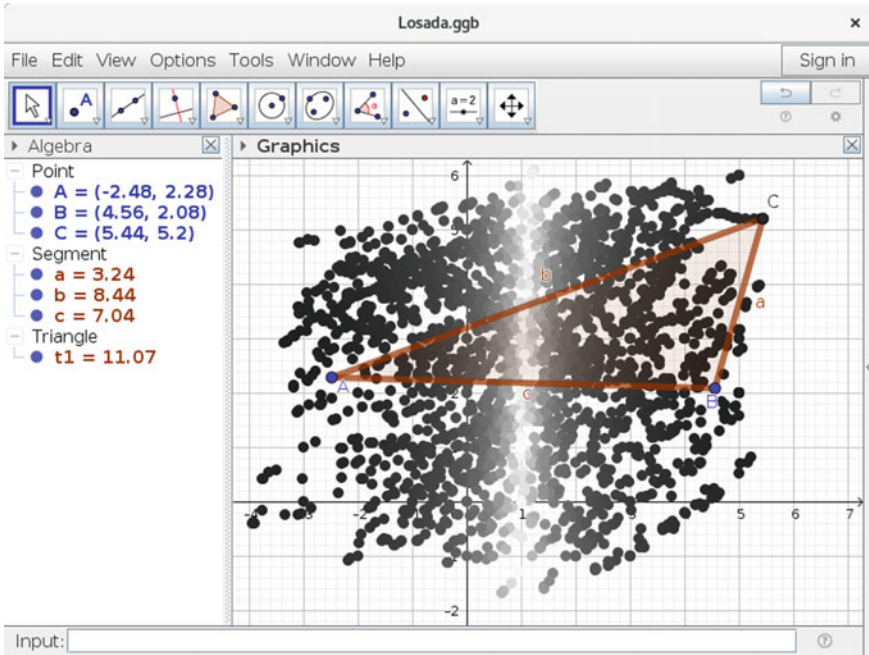





Fig. 2 Colouring the plane to find the locus. Here tracing is enabled for point C and *Dynamic Colors* are set for each RGB component by using the formula $\frac{1}{|a-b|+1}$

In the final position, we obtained a linear equation again (here $-328x + 46y = -303$). Hopefully, most students have the correct conjecture at this point; namely, that the locus is the perpendicular bisector of AB.

To continue with Buchberger’s schema, the student then goes on to Step 3—proving the conjecture numerically and symbolically. This substantial step requires a somewhat different construction. After disabling or deleting the obtained implicit curve d in the construction, the student should create the perpendicular bisector f of AB by using the *Perpendicular Bisector* tool . Then, by attaching an arbitrary point D on f , and—by using the *Segment* tool —creating segments $g = AD$ and $h = BD$, point D will be freely draggable on f only and the *Algebra View* will show up-to-date information on the lengths of g and h .

Finally, using the *Relation* tool  with respect to g and h , the student tests (first, numerically) the validity of conjecture about the equidistance of all the points in the perpendicular bisector to the extremes of the segment. The *Relation* tool asks for any potential relation between these two segments.

Things are actually more complicated than simply concluding that $g = h$ in all cases. Figure 5 shows an apparent counterexample concerning the lengths of these two segments after setting *Options/Rounding* to *15 Decimal Places*. Luckily, when comparing them by with the *Relation* tool, GeoGebra knows that a difference in

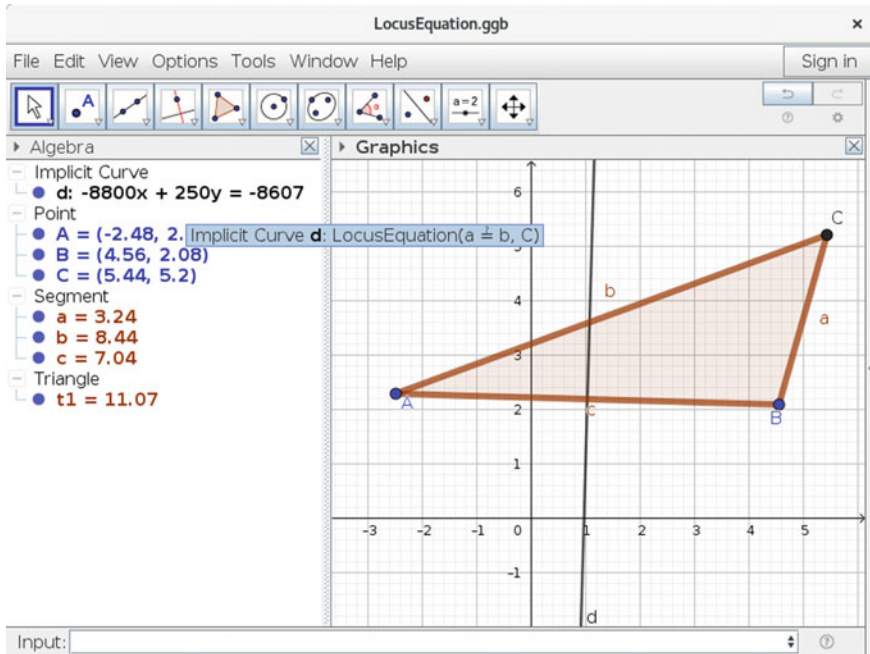


Fig. 3 Computing the locus equation in a particular case

precisely the last digit may be just a numerical error and assumes—just for the output of *Relation*—that the numerical comparison resulted in equality (Fig. 6).

Finally, overcoming all these potential numerical inaccuracies, by clicking the button “More...”, a symbolic check (that is, a mathematically rigorous proof) rigorous proof will be performed. ‘Symbolic’ here means a lot. Indeed, the input data are not anymore points with numerical coordinates, but with coordinates expressed by means of variables. The construction steps do not yield equations with numerical coefficients, but parametric equations depending on the parameters describing the coordinates of the free points, etc. Then, using sophisticated computer algebra algorithms, a complete proof consisting of up to a few millions of algebraic steps is done in the background. Only the final result “always true” is shown to the user (Fig. 7; see Kovács, 2015 for details).

Now the student has reached a point where there is more than enough evidence, based on the DGS automated reasoning tools, that the searched locus is the perpendicular bisector. The teacher (or the student) may then want to find a non-automated proof, and also a chain of reasons *why* the theorem holds. In this paper we do not focus on how this could be done in the best way.

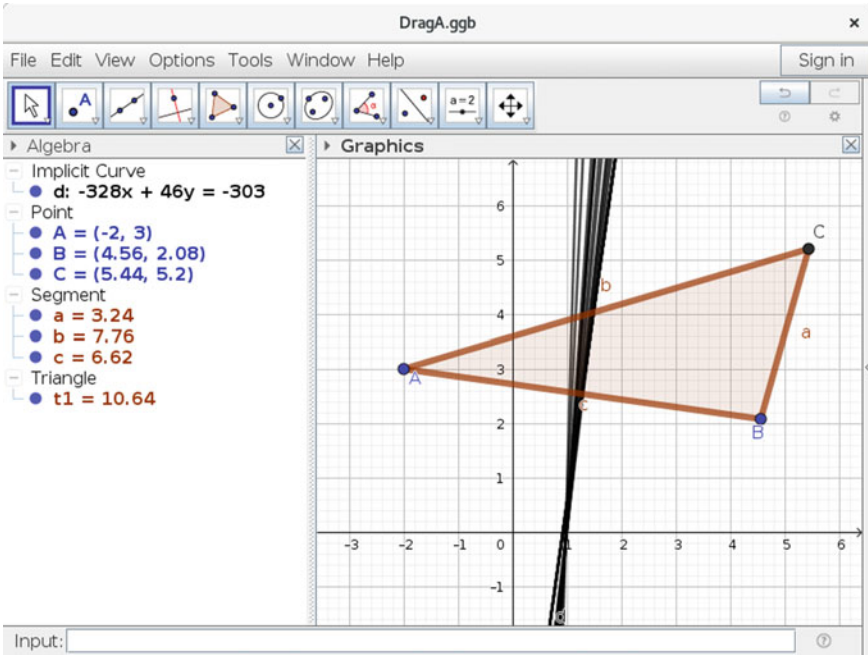


Fig. 4 Computing the locus equation for more than one case

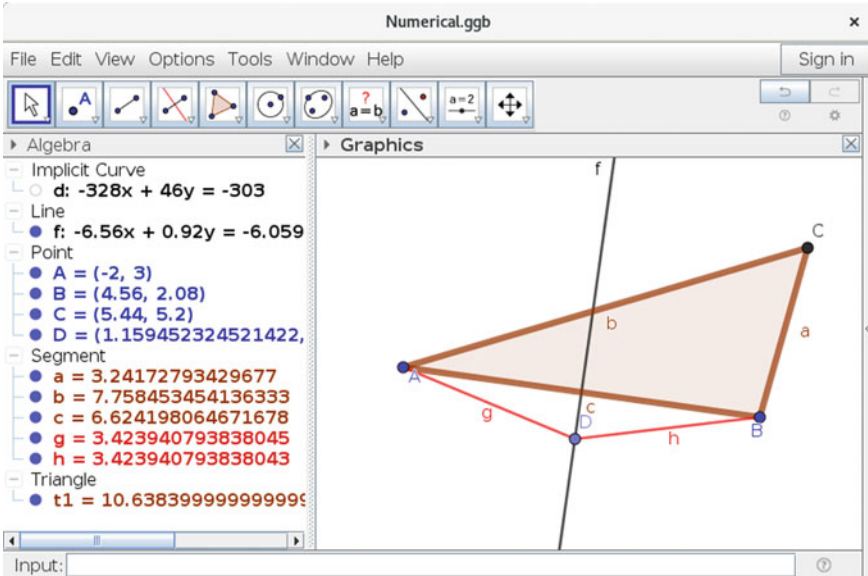


Fig. 5 Rounding errors may lead to confusing results

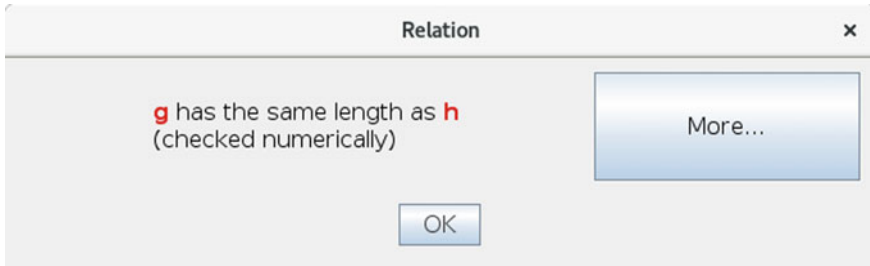


Fig. 6 The *Relation* tool assumes that a minor numerical error may still not be contradictory to the conjecture

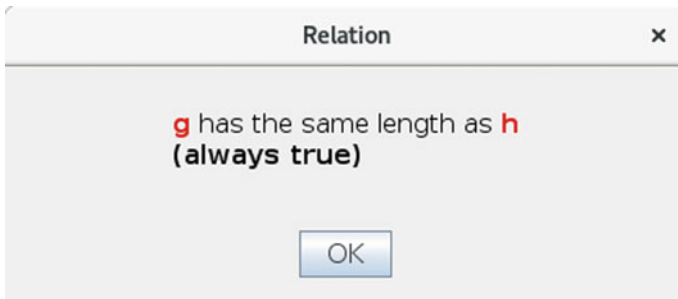


Fig. 7 Symbolic check in the *Relation* tool

4.1 A Second Round

Instead, we suggest going for a second round in the creativity spiral. Step 4 is about “programming”—here it could mean constructing a new figure or observing something different but related to the first round. There are several ideas to think about, but we will focus on the following: *Can we change the proposed statement by using a different formula than $a == b$?* This idea involves just a minimal modification of the context, but results in a surprising change of the output. For example, let us consider the formula $a == 2b$.

The students can start again with Step 1 by doing various computations, including entering `LocusEquation(a==2b,C)`. Figure 8 shows the result.

It may be not obvious to the student what type of curve is shown as the output. Here the equation $x^2 + 8x + y^2 - 4y = -4$ is shown, which is equivalent to the formula $(x + 4)^2 + (y - 2)^2 = 4^2$ —clearly a circle. The student should, however, do several other attempts by dragging points *A* and/or *B* to verify other particular cases (Fig. 9). In the final position $A = (3, 4)$, $B = (4, 2)$, the implicit curve $3x^2 - 16x + 3y^2 - 28y = -80$ is obtained, which is, again, a circle, with a more difficult equivalent form. The intermediate curves also seem to be circles; that is, we may have a strong conjecture that the equation is a circle *in general*.

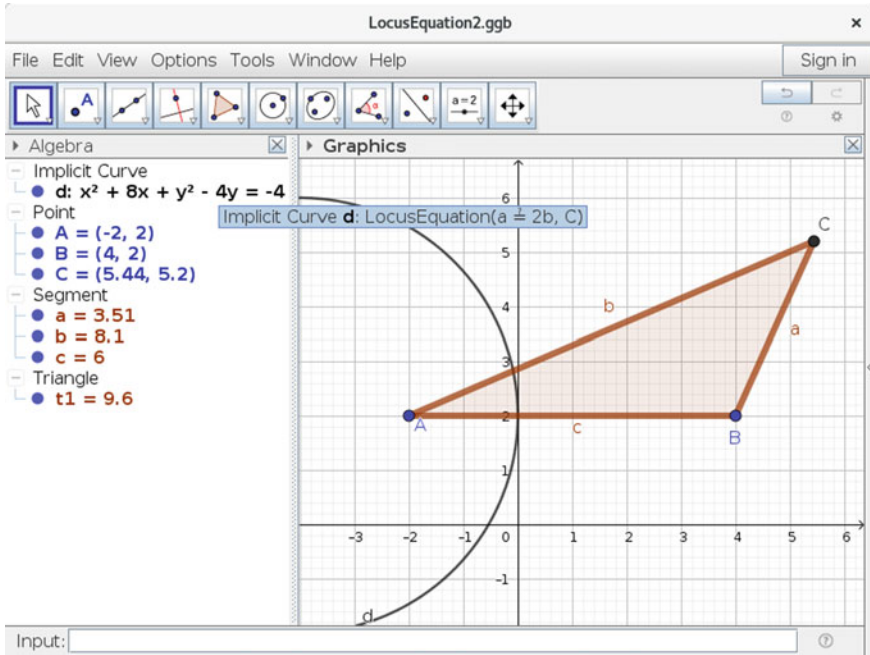


Fig. 8 Computing the locus equation in a particular case (second round)

We leave the reader to find the equivalent formula for the final circle in Fig. 9. It should help to discover the exact position of the conjectured circle. In addition, Fig. 10 discloses how the conjectured circle can be constructed by using a compass and a straightedge ruler. For some GeoGebra implementation issues, these tools are required to enable symbolic checks in the *Relation* tool.

Here we observe that the obtained circle is a *circle of Apollonius* (of Perga), with foci *A* and *B* and ratio 2.

4.2 Other Rounds

Clearly, other interesting questions can arise in this context, with surprising answers and involving unusual geometric facts. Here we only mention one simple possible modification of the previous formula; namely, considering `LocusEquation(a=2b+1, C)`, as shown in Fig. 11.

The set of obtained curves are of various geometry. In some cases, two closed curves are shown as the locus, but in other cases, just a subset of points in the locus output fulfills the formula requirements. For example, in Fig. 11 only the external curve provides points such that $a = 2b + 1$, while the internal one fulfills the

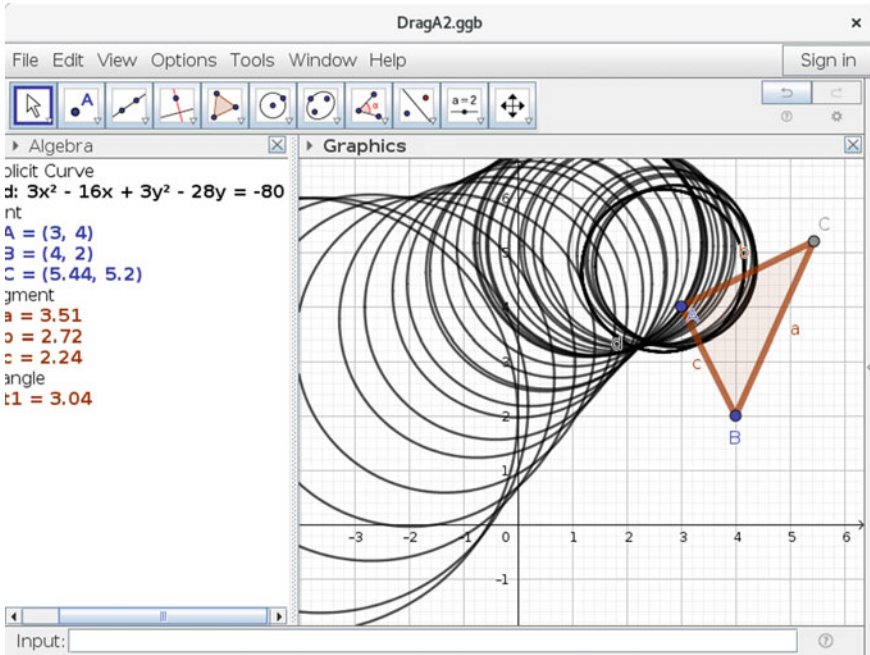


Fig. 9 Computing the locus equation in some particular cases (second round)

condition $a = 2b - 1$. The reason behind this unexpected effect has deeper roots in algebraic geometry and cannot be discussed in the classroom or in this chapter. Instead, we would like to emphasize that seemingly simple questions can actually lead to involved issues. In our opinion, however, this should not prevent us from *asking freely*—even more so, if our students ask questions on their own and get curious to discover the answer!

Finally, let us mention here the proposal of Krause (1975) that introducing a minimal change in the geometric assumptions of a given context results in a surprising shift towards a substantially new theory. We think the best educational setting happens when students are surprised by the answers they get and inspired to spend more time on discovering the richness of mathematics.

In this section we have discussed just one particular topic in planar geometry. However, many other topics can be raised in the classroom by using the “implicit locus approach”. Some typical curriculum topics include:

1. (The converse of) *Thales’ circle theorem* (see Artigue, 2012; Kovács & Schiffler, 2017): Given a segment AB , what is the locus of points C such that $AC \perp BC$? (A generalization of this approach yields the inscribed angle theorem.)
2. A variation of the *triangle inequality*: Given a triangle ABC with side lengths a , b and c , where the points A and B are fixed, what is the locus of points C such that $a + b = c$?

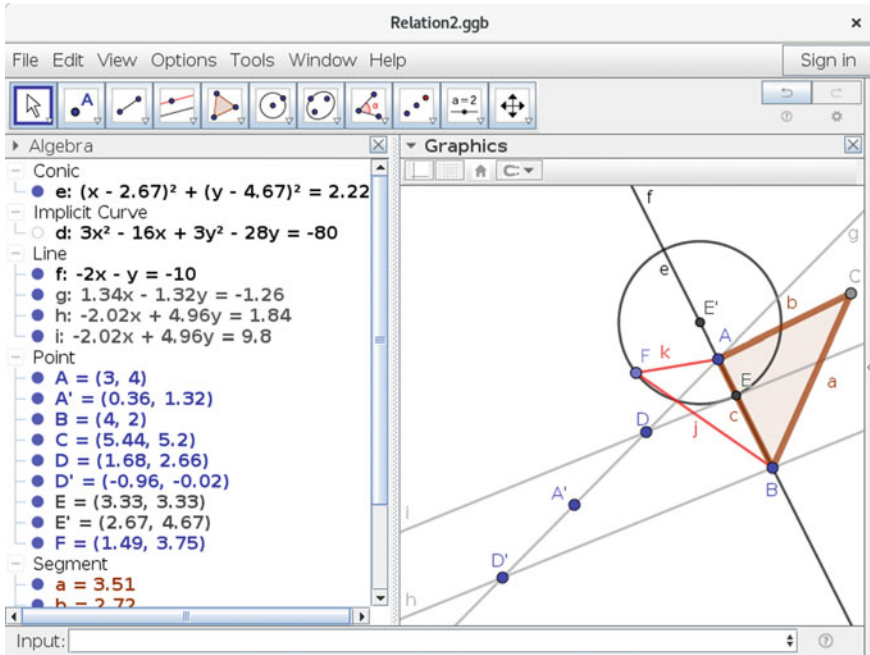


Fig. 10 Constructing the solution by using steps defined only by a compass and a ruler

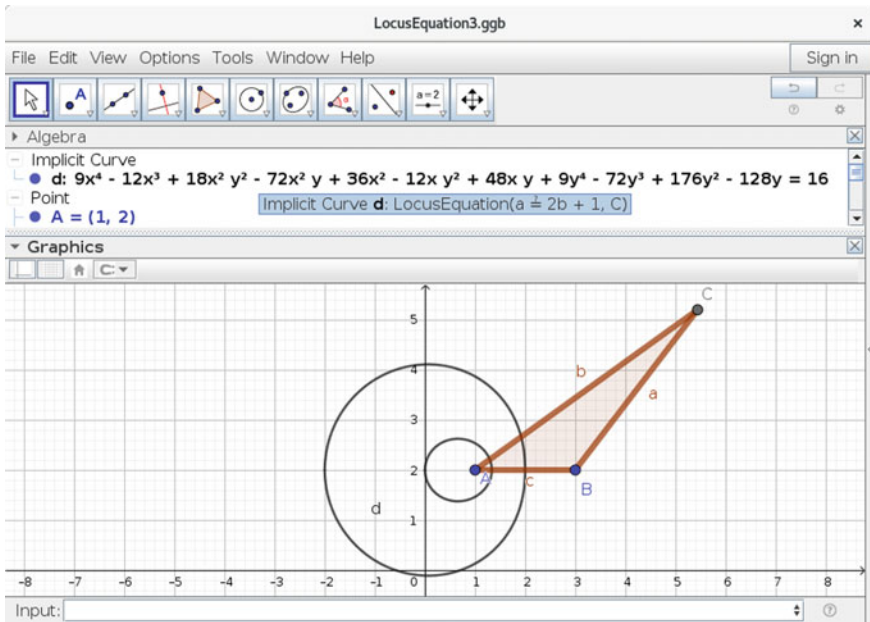


Fig. 11 Sometimes the locus equation can be a “yet unknown”, “strange” curve

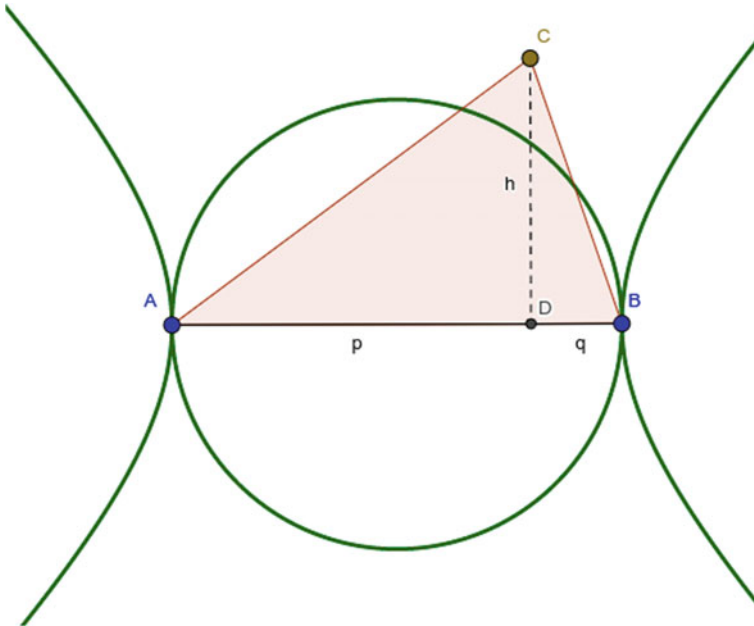


Fig. 12 GeoGebra ART detects an unusual locus when analyzing a traditional statement

3. (The converse of) *Pythagoras' theorem*: Given a triangle ABC with side lengths a , b and c , where the points A and B are fixed, what is the locus of points C such that $a^2 + b^2 = c^2$?
4. (The converse of) the *right triangle altitude theorem* (also known as *geometric mean theorem*): Given a triangle ABC with altitude h (with respect to C) and two line segments p and q that it creates on side c , what is the locus of points C such that $h = \sqrt{pq}$ (or, equivalently, $h^2 = pq$)? (See Fig. 12).

Let us highlight here that the obtained locus in the last example is not just a circle—according to the right triangle being assumed in the traditional theorem—but also a hyperbola (Abánades, Botana, Kovács, Recio, & Sólyom-Gecse, 2016a). This provides an immediate but not well known generalization of the right altitude theorem.

On the other hand, implicit loci are just a part of the new possibilities in GeoGebra ART. Other commands like direct *Prove*, *ProveDetails* or *Envelope* computation, are also available.

We refer to Botana (2016), Hašek (2017), Kovács and Vajda (2017), Kovács (2017, 2018), Abánades et al. (2016a, b), Kovács et al. (2017) for further details and examples of these recent methods in GeoGebra and to (Kovács et al., 2017) for full documentation.

5 Conclusion

It was stated 30 years ago in the futurist ICMI Study “School Mathematics in the 1990s” (Howson & Wilson, 1986) that

... even if the students will not have to deal with computers till they leave school, it will be necessary to rethink the curriculum, because of the changes in interests that computer have brought.

It was also long ago that an inspiring paper (Davis, 1995) by Philip Davis included a section specifically on the power of computer-based proofs to “transfigure” geometry statements and therefore to affect the potential role in mathematics education of software programs dealing with automatic theorem proving.

Another seminal paper in this context is Gila Hanna’s “Challenges to the Importance of Proof” (Hanna, 1995). Hanna considered the pedagogical and epistemological impact of different ways of using the computer to prove mathematical facts—for example, verifying a finite number of cases that remain to be checked for completing the proof of a statement, the so-called zero-knowledge proofs, and the case of proofs with a high probability of being correct. She also raised another challenge: the increasing habit of using the computer to establish the truth of a statement (visually or numerically) through experiments rather than formal proof.

Most of these messages about the relation between computers and proofs were essentially warning signs about issues that could take place in future worlds. Yet we think they are still quite present today. In fact, we see three basic concerns regarding dynamic geometry and mathematical reasoning. The first relates to the negative influence of the versatile visualization features of dynamic geometry programs. As Lin, Yang, Lee, Tabach, and Stylianides emphasize in (2012),

[The] increased availability in school mathematics instruction of ... dynamic geometry systems ... raised the concern that such programmes would make the boundaries between conjecturing and proving even less clear for students.

[They] allow students to check easily and quickly a very large number of cases, thus helping students ‘see’ mathematical properties more easily and potentially ‘killing’ any need for students to engage in actual proving.

Indeed, what is involved in the easy checking of a large number of cases is the “dragging” feature of DGS and, therefore, the above worries apply very particularly to all DGS.

The second concern relates only to the few DGS that currently provide automated reasoning tools, with at least the ability to confirm/deny the mathematical (not probabilistic) truth of a geometric statement. In the case of GeoGebra, as mentioned above, these tools are enlarged with other features for automatic discovery and beyond.

In fact, DGS with ART features can be considered “geometry calculators” and therefore as reflecting the already well known concerns about arithmetic or scientific calculators in mathematics education:

Can students be intellectually attracted to compute 23456769×98765432 , once they know there is an algorithm that yields the correct answer 2316717923609208 and that it has been

implemented in their personal, say, tablet or phone? Likewise, will they be interested in finding whether the three heights of a triangle meet always at one point, if their pocket phone is able to guarantee, with a mathematical algorithm, that they certainly do so? (Hohenwarter et al., 2017)

Furthermore, we could ask: will they be interested in developing a local “deductive theory” (de Villiers, 1990) when insight into mathematical evidence does not seem to require this ability anymore, but, instead, a different kind of “cooperative reasoning” with the machine as described above?

The answer is unclear to us. It is possible the two contexts (arithmetic, geometry calculators) are not actually that parallel. In fact, the elementary geometry curriculum has always been deeply affected by the choice of “geometry tools” such as the ruler or the compass. Perhaps it is now the turn of a different geometry based on the existence and possibilities of Dynamic Geometry ART. Perhaps we should not just keep using new technology for old problems (old problems that are intimately “old tools”-driven). And perhaps we should consider the new technology as something other than a source of “concerns” (as outlined in Lin et al., 2012).

A third concern, going beyond the educational world, and considering GeoGebra ART just as a tool for the professional mathematician, has to do with one of Hanna’s “challenges to the importance of proof.” Although very relevant and nowadays increasingly present in professional math activity, the methods referred in Hanna (1995) for deciding the truth of a statement (using the computer to verify a large set of instances; zero-knowledge proofs; probabilistic proofs) have little to do with our approach, except for one collateral coincidence. In fact, as we have already emphasized, GeoGebra ART algorithms yield an exact (not numerically approximate, probabilistic, visual, or experimental) proof that would be accepted by the mathematical community as *if it were made by hand by a human*. This highlighted phrase is, perhaps, the crucial point. As Hanna poses in the above article:

Should mathematicians accept proofs that can not be verified by others?

Well, no, in our opinion. But, as described above, the main role we are proposing for the educational use of GeoGebra ART is not ‘acceptance,’ but “suggesting” ideas and ways for producing a formal, human readable, proof. The problem, of course, does not end with this declaration. We could also ask: what does “verified by others (humans)” mean today? Is using a calculator to verify a numerical computation such as 23456769×98765432 forbidden in this context? Is “the verification” more acceptable only if the algorithm for this multiplication is done by hand on a piece of paper? How do we verify the correction of the algorithm and its implementation (by hand, by the human)? And, finally, if humans are allowed to use calculators to verify computations, should we restrict this to numerical calculators (and not to those performing algebraic computations)? Should the algorithmic expansion of $(x + y)^{100}$ performed by GeoGebra on a mobile phone app be considered less “human verifiable” than the same computation done by a human on a blackboard?

We think there is a need to address urgently the extended impact of the merging, in one single tool, of the above three challenges related to DGS and Automatic Reasoning: (1) the potential, discouraging influence of powerful visualization features, (2)

the availability of an exact “geometry calculator”, and (3) the precise meaning today of “human readable” or “human verifiable” in a society where computers, laptops, and mobile phones are everywhere from kindergarten up.

Yet, the very recent survey (Sinclair et al., 2016) by Sinclair, Bartolini Bussi, de Villiers, Jones, Kortenkamp, Leung and Owens does not refer at all to automated reasoning tools. We think it is an urgent issue to address in future surveys of this kind, given the large expansion of GeoGebra in the classrooms worldwide—over 100 million users of its apps and website in 2017—and the fact that ART features have recently been included in it (see Kovács, 2015). This quantitative fact, indeed, has made a qualitative difference: as with pocket calculators, people will probably use ART for checking geometric facts with or without the consensus of the pedagogical community on its role.

We do not feel competent at this point to propose a concrete route towards geometry teaching and learning in this new context. We need feedback from the educational community (researchers, teachers, students) concerning what it means to *teach* and to *learn* and what *elementary geometry* means in this extended framework.

We simply feel GeoGebra’s automated reasoning capabilities can help our students to do mathematics better or faster, just as we think it is beneficial to have an electronic calculator to compute the square root of a number much faster than using the traditional, mechanical method by hand (which, not surprisingly, is no longer part of most curricula).

On the other hand, in the previous sections we have shown that even a simple question can yield difficult or surprising issues. It is in addressing these issues that mathematical creativity and reasoning can be fully developed but in a non-traditional way: human and machine equally ready to explore the geometric context. In this way we could argue that the intention of ART is not just to do the same kind of mathematics better and faster, but to do “a better kind of mathematics.” Let us borrow Kaput’s visionary words, cited by Balacheff (1997): instead of *doing (old) things better* we should focus on *doing better things*.

Acknowledgements We thank the referees for many interesting suggestions and comments and, in particular, for pointing us to several relevant bibliographic references. Special thanks to Gila Hanna, Dragana Martinovic, Chris Sangwin, Arleen Schenke, and David A. Reid for their direct help in improving the text of this chapter.

The second and third authors have been partially funded by Spanish and EDF Research Grant MTM2017-88796-P.

References

- Abánades, M., Botana, F., Kovács, Z., Recio, T., & Sólyom-Gecse, C. (2016a). Towards the automatic discovery of theorems in GeoGebra. In G. M. Greuel, T. Koch, P. Paule, & A. Sommese (Eds.), *Mathematical Software—ICMS 2016. 5th International Conference*, Berlin, Germany, July 11–14, 2016, Proceedings. Volume 9725 of Lecture Notes in Computer Science (pp. 37–42). Cham: Springer International Publishing.

- Abánades, M., Botana, F., Kovács, Z., Recio, T., & Sólyom-Gecse, C. (2016b). Development of automatic reasoning tools in GeoGebra. *ACM Communications in Computer Algebra*, 50, 85–88.
- Artigue, M. (2012). What is inquiry-based mathematics education (IBME)? In M. Artigue & P. Baptist (Eds.), *Inquiry in mathematics education* (pp. 3–13). Fibonacci Project.
- Balacheff, N. (1997). ICME 8, TG19 followup report. Computer-Based Learning Environments: “CBILE”. <http://mathforum.org/mathed/seville/followup.html>.
- Baulac, Y., Bellemain, F., & Laborde, J. M. (1994). *Cabri geometry II*. Dallas: Texas Instruments.
- Botana, F., & Kovács, Z. (2016). New tools in GeoGebra offering novel opportunities to teach loci and envelopes. [arXiv:1605.09153](https://arxiv.org/abs/1605.09153).
- Botana, F., & Valcarce, J. L. (2002). A dynamic-symbolic interface for geometric theorem discovery. *Computers and Education*, 38, 21–35.
- Botana, F., Hohenwarter, M., Janičić, P., Kovács, Z., Petrović, I., Recio, T., et al. (2015). Automated theorem proving in GeoGebra: Current achievements. *Journal of Automated Reasoning*, 55, 39–59.
- Buchberger, B., & The Theorema Working Group. (1998). *Theorema: Theorem proving for the masses using Mathematica*. In *Invited Talk at the Worldwide Mathematica Conference*, Chicago, June 18–21, 1998.
- Chou, S. C. (1987). Mechanical geometry theorem proving. Springer Science+Business Media.
- Corless, R. M. (2004). Computer-mediated thinking. <http://www.apmaths.uwo.ca/~rcorless/frames/PAPERS/EDUC/CMTpaper.pdf>.
- Corpuz, J. (2017). Best math apps. <https://www.tomsguide.com/us/pictures-story/1300-best-math-apps.html>.
- Davis, P. (1995, March). The rise, fall, and possible transfiguration of triangle geometry: A mini-history. *The American Mathematical Monthly*, 102, 204–214.
- de Villiers, M. (1999). *Rethink proof with sketchpad*. Emeryville: Key Curriculum Press.
- de Villiers, M. (1990). The role and function of proof in mathematics. *Pythagoras*, 24, 17–24.
- Foster, C. (2013). Mathematical études: Embedding opportunities for developing procedural fluency within rich mathematical contexts. *International Journal of Mathematical Education in Science and Technology*, 55, 765–774.
- Gelernter, H. (1959). Realisation of a geometry-proving machine. In *Proceedings of the International Conference on Information Processing*, Paris, June 15–20, 1959 (pp. 273–282).
- Halmos, P. R. (1982). *A Hilbert space problem book* (2nd ed.). New York, Heidelberg, Berlin: Springer.
- Hanna, G. (1995). Challenges to the importance of proof. *For the Learning of Mathematics*, 15, 42–49.
- Hašek, R., Kovács, Z., & Zahradník, J. (2017). Contemporary interpretation of a historical locus problem with the use of computer algebra. In I. S. Kotsireas & E. Martínez-Moro (Eds.), *Applications of Computer Algebra: Kalamata*, Greece, July 20–23, 2015. Volume 198 of Springer Proceedings in Mathematics & Statistics. Springer.
- Hohenwarter, M. (2002). GeoGebra: Ein Softwaresystem für dynamische Geometrie und Algebra der Ebene. Master’s thesis, Paris Lodron University, Salzburg, Austria.
- Hohenwarter, M., Kovács, Z., & Recio, T. (2017). Deciding geometric properties symbolically in GeoGebra. In R&E-SOURCE (2017) Special Issue #6: 13th International Congress on Mathematical Education (ICME-13).
- Howson, G., & Wilson, B. (1986). *ICMI Study Series: School mathematics in the 1990s*. Kuwait: Cambridge University Press.
- Jackiw, N. R. (1995). *The Geometer’s Sketchpad*, v3.0. Berkeley, CA: Key Curriculum Press.
- Jaworski, B. (1994). *Investigating mathematics teaching: A constructivist enquiry*. Studies in Mathematics Education Series: 5. The Falmer Press.
- Jones, P. L. (1996). Handheld technology and mathematics: Towards the intelligent partnership (pp. 87–96). <http://ued.uniandes.edu.co/roless-calc.html>.
- Kapur, D. (1986). Using Gröbner bases to reason about geometry problems. *Journal of Symbolic Computation*, 2, 399–408.

- Kortenkamp, U. (1999). Foundations of dynamic geometry. Ph.D. thesis, ETH Zürich.
- Kovács, Z. (2015). Computer based conjectures and proofs in teaching Euclidean geometry. Ph.D. thesis, Johannes Kepler University, Linz.
- Kovács, Z. (2017). Real-time animated dynamic geometry in the classrooms by using fast Gröbner basis computations. *Mathematics in Computer Science*, 11.
- Kovács, Z. (2018). Automated reasoning tools in GeoGebra: A new approach for experiments in planar geometry. *South Bohemia Mathematical Letters*, 25.
- Kovács, Z., & Parisse, B. (2015). Giac and GeoGebra—Improved Gröbner basis computations. In J. Gutierrez, J. Schicho, & M. Weimann (Eds.), *Computer algebra and polynomials*. Lecture Notes in Computer Science (pp. 126–138). Springer.
- Kovács, Z., & Schiffler, K. (2017). Unterstützung des Mathematikunterrichts mit automatischem Beweisen mit GeoGebra. In: PH forscht II, Linz, Austria.
- Kovács, Z., & Vajda, R. (2017). A note about Euler's inequality and automated reasoning with dynamic geometry. [arXiv:1708.02993v2](https://arxiv.org/abs/1708.02993v2).
- Kovács, Z., Recio, T., & Vélez, M. P. (2017). gg-art-doc (GeoGebra Automated Reasoning Tools. A tutorial). A GitHub project. <https://github.com/kovzol/gg-art-doc>.
- Kovács, Z., Recio, T., Richard, P. R., & Vélez, M. P. (2017). GeoGebra automated reasoning tools: A tutorial with examples. In G. Aldon & J. Trgalova (Eds.), *Proceedings of the 13th International Conference on Technology in Mathematics Teaching*. (pp. 400–404). <https://hal.archives-ouvertes.fr/hal-01632970>.
- Krause, E. F. (1975). *Taxicab geometry: An adventure in non-Euclidean geometry*. Addison-Wesley.
- Kutzler, B., & Stifter, S. (1986). On the application of Buchberger's algorithm to automated geometry theorem proving. *Journal of Symbolic Computation*, 2, 389–397.
- Lin, F. L., Yang, K. L., Lee, K. H., Tabach, M., & Stylianides, G. (2012). Principles of task design for conjecturing and proving. In G. Hanna & M. de Villiers (Eds.), *Proof and Proving in Mathematics Education. The 19th ICMI Study* (pp. 305–326). Springer.
- Lindenbauer, E., & Reichenberger, S. (2015). Voronoi-Diagramme. GeoGebra Materials. <https://www.geogebra.org/m/sAaFMcTA>.
- Losada, R. (2014). El color dinámico en GeoGebra. *La Gaceta de la Real Sociedad Matemática Española*, 17(3), 525–547.
- Losada, R., Recio, T., & Valcarce, J. L. (2011). Equal bisectors at a vertex of a triangle. In B. Murgante, O. Gervasi, A. Iglesias, D. Taniar, & B. Apduhan (Eds.), *Computational Science and Its Applications—ICCSA 2011*. Lecture Notes in Computer Science (Vol. 6785, pp. 328–341). Berlin, Heidelberg: Springer.
- Martinovic, D., & Manizade, A. G. (2014). Technology as a partner in the geometry classrooms. *The Electronic Journal of Mathematics and Technology*, 8, 69–87.
- Martinovic, D., Muller, E., & Buteau, C. (2013). Intelligent partnership with technology: Moving from a math school curriculum to an undergraduate program. *Comput. Schools*, 30, 76–101.
- Oldenburg, R. (2008). *FeliX - mit Algebra Geometrie machen (German)* (pp. 15–17). Computeralgebra Rundbrief: Sonderheft zum Jahr der Mathematik.
- Polya, G. (1962). *Mathematical discovery: On understanding, learning, and teaching problem solving*. London, UK: Wiley.
- Recio, T., & Vélez, M. P. (1999). Automatic discovery of theorems in elementary geometry. *Journal of Automated Reasoning*, 23, 63–82.
- Schwartz, J. L., & Yerushalmy, M. (1983). *The Geometric Supposer*. Pleasantville, NY: Sunburst Communications.
- Sinclair, N., Bartolini Bussi, M. G., de Villiers, M., Jones, K., Kortenkamp, U., Leung, A., et al. (2016). Recent research on geometry education: An ICME-13 survey team report. *ZDM Mathematics Education*, 48, 691–719.
- Wu, W. T. (1978). On the decision problem and the mechanization of theorem-proving in elementary geometry. *Scientia Sinica*, 21(2). Reprinted In W. W. Bledsoe, & D. W. Loveland (Eds.), *Automated Theorem-Proving: After 25 Years*. Providence, RI: AMS (1983).

Ye, Z., Chou, S. C., & Gao, X. S. (2011). An introduction to Java Geometry Expert. In *Automated Deduction in Geometry, 7th International Workshop, ADG 2008*, Shanghai, China, September 22–24, 2008. Revised Papers, Lecture Notes in Computer Science (Vol. 6301, pp. 189–195). Springer.

Computer-Generated Geometry Proofs in a Learning Context



Pedro Quaresma and Vanda Santos

1 Introduction

Given its formal, logical, and spatial properties, geometry is well suited to teaching environments that include dynamic geometry systems (DGSs), geometry automated theorem provers (GATPs), and repositories of geometric problems. These tools enable students to explore existing knowledge in addition to creating new constructions and testing new conjectures. With the help of a DGS, students can visualise geometric objects and link the formal, axiomatic nature of geometry (e.g., Euclidean geometry) with its standard models and corresponding illustrations (e.g., the Cartesian model). With the help of GATPs, students can check the soundness of a construction (e.g., if two given lines are parallel) and also create formal proofs of geometric conjectures. Supported by repositories of geometric knowledge, these tools provide teachers and students with a framework and a large set of geometric constructions and conjectures for doing experiments.

In this chapter, we trace the evolution of current automatic proving technologies, how these technologies are beginning to be used by geometry practitioners in general to validate geometric conjectures and generate proofs with natural language and visual rendering, and foresee their evolution and applicability in an educational setting. Following Hanna's (2000, p. 8) argument that "the best proof is one that also helps understand the meaning of the theorem being proved: to see not only that it is true, but also why it is true," and the large number of articles on proof and proving in mathematics education from the ICMI Study 19 Conference (Lin, Hsieh, Hanna, & de Villiers, 2009a, 2009b), we focus our attention on practices of verification, explanation, and discovery in the teaching and learning of geometry.

In the classroom, the fundamental question a proof must address is "why?" In this context, then, it is only natural to view proofs first and foremost as explanations and, as

P. Quaresma (✉) · V. Santos
University of Coimbra, 3001-501 Coimbra, Portugal
e-mail: pedro@mat.uc.pt

V. Santos
e-mail: vsantos7@gmail.com

a consequence, to give more value to those that provide a better explanation. Dynamic geometry systems encourage both exploration and proof because they make it so easy to pose and test conjectures. The feature that preserves manipulations allows students to explore “visual proofs” of geometric conjectures. Such a powerful feature gives them strong evidence that a theorem is true and reinforces the value of exploration by giving them confidence in a theorem.

The challenge facing classroom teachers is how to use the excitement and enjoyment of exploration to motivate students while also explaining that visual exploration is not a proof. Visual exploration is a useful aid, but is still only the exploration of a finite number of cases. One reason for giving students a formal proof is that exploration does not reflect the need for rigour in mathematics. Indeed, mathematicians aspire to a degree of certainty that can only be achieved by a proof. A second reason is that students should come to understand the first reason. As most mathematics educators would agree, students need to be taught that exploration, useful as it may be in formulating and testing conjectures, does not constitute a proof (Hanna, 2000; Hanna & Sidoli, 2007). A proof is a means of obtaining certainty about the validity of a conjecture (proof as a validation tool) and a strategy to further understand a formulated conjecture (proof as an instrument of understanding).

Geometry automated theorem provers open the possibility of formally validating properties of geometric constructions. For example: *Cinderella*¹ (Richter-Gebert & Kortenkamp, 1999) has a randomised theorem checker; *Geometry Constructions LaTeX Converter (GCLC)*² (Janičić, 2006) and *GeoGebra*³ (version 5) (Hohenwarter, 2002) incorporate a number of automated theorem provers that provide a formal answer to a given validation question (Botana et al., 2015; Janičić & Quaresma, 2007).

By means of the deductive database method (Ye, Chou, & Gao, 2010b), GATPs also enable students to explore new knowledge and discover new results and theorems (e.g., the algebraic formula of a loci (Abánades, Botana, Kovács, Recio, & Solyom-Gecse, 2016; Recio & Vélez, 2012)). An important addition to any learning environment would be a GATP with the ability to produce human readable formal proofs with, eventually, visual counterparts (Chou, Gao, & Zhang, 1996a, 1996b; Janičić, Narboux, & Quaresma, 2012; Kovács, 2015; Stojanović, Pavlović, & Janičić, 2011; Stojanović, Narboux, Bezem, & Janičić, 2014).

In this chapter, we focus on the subjects of verification and explanation. Section 2 describes the past, present, and foreseeable future of geometry automated theorem proving. Section 3 uses examples to introduce the use of GATPs in the formal validation of a conjecture. In Sect. 4, we explore GATPs that produce readable formal proofs. In Sect. 5, we extend this exploration to GATPs that produce formal proofs with the addition of natural language and visual rendering. In Sect. 6, we draw conclusions and anticipate future avenues of research.

¹<https://cinderella.de>.

²<http://poincare.matf.bg.ac.rs/~janicic/gclc/>.

³<https://www.geogebra.org/>.

2 Formal Geometric Proofs

Given the cognitive complexity of geometric activities, learning geometry is not always immediate or easy. However, visualisation can be used to understand proofs and help mathematical reasoning. Visual elements attract students' attention, raise their awareness, and help them make connections between subjects and concepts. Visualization contributes to a better understanding of the results obtained from a search for different approaches to solving a problem or proving a theorem. It promotes new forms of reasoning that can inspire simple and elegant solutions to similar problems, allowing students to respond to more complex problems or prove new theorems (de Villiers, 2006).

Introducing DGSs in the learning process enables greater experimentation than previous tools did. DGSs allow free geometric objects to be continuously modified, thereby keeping geometric properties unchanged. Although these manipulations are not formal proofs because only a finite set of positions is considered and because visualisation can be misleading (when the image, for example, refers to a particular case), they provide a first clue to the truthfulness of a given geometric conjecture (Hoyles & Jones, 1998; Nelsen, 1993; de Villiers, 2006). It can be said that DGSs provide an initial non-formal link between theories and models of geometry (Janičić & Quaresma, 2007; Ruthven, Hennessy, & Deaney, 2008).

Geometry automated theorem provers similarly enhance learning. First, they can be used to validate a geometric construction such as: are two given lines parallel? The answer—yes!/no!/cannot be established—is a “black box” approach (it does not answer the “why”), but can be useful in situations where formal proofs are not being considered. Second, some GATPs are able to provide formal proofs—not only the answer to a question, but also the “why.” As said above, this constitutes the “best proof.”

Unlike DGSs that are similar in their approach and capabilities, GATPs vary significantly in terms of the method they use (e.g., synthetic vs. algebraic), the implementation (e.g., area vs. full-angle), and the output (formal proof vs. only a yes/no answer) (Chou & Gao, 2001; Chou, Gao, & Zhang, 1994; Quaresma, 2017; Wang & Su, 2015; Wu, 1984). According to Jiang and Zhang (2012), consideration needs to be given to the somehow opposite goals of efficiency and readability. Efficiency is important because, in a learning situation, it is not viable to wait more than a couple of seconds to get an answer. Readability is important given that, without it, the “why” would be lost.

The first methods proposed in the 1950s adapted general reasoning approaches from the field of artificial intelligence to the automation of traditional geometric proving processes. In order to avoid combinatorial explosion while applying postulates, many suitable heuristics were used; for example, adding auxiliary elements to geometric configurations, and using the corresponding geometric constructions as a source of counterexamples to help the reasoning procedure choose the right path. Although these methods produced readable proofs that could be used in learning, they were very narrow in scope and inefficient (Wu, 1984). However, some recent

results using this approach in education focus on developing GATPs scoped for a given set of problems (Paneque, Cobo, Fortuny, & Richard, 2016; Richard, Oller Marcén, & Meavilla Seguí, 2016). Still, its usefulness in education has yet to be established.

The next step in creating more generic, powerful, and efficient provers was to integrate algebraic methods such as the characteristic set method (also known as Wu’s method) (Chou, 1985; Wu, 1984), the polynomial elimination method (Wang, 1995), the Gröbner based method (Kapur, 1986), and the Clifford algebra approach (Li, 2000). These methods reduced the complexity of logical inferences by computing relations between coordinates of geometric entities. The corresponding implementations could successfully solve many complicated geometric problems and discover new theorems. However, human readability of the proofs was lost, and the algebraic proofs were complex and not related to geometric reasoning (Chou et al., 1994; Wu, 1984). Nonetheless, there are many recent implementations of these methods (Botana et al., 2015; Janičić et al., 2012; Janičić & Quaresma, 2006; Kovács, 2015; Ye, Chou, & Gao, 2011) and, in Sect. 3, we explore the possible use of these GATPs in education.

In an effort to combine the readability of synthetic methods and the efficiency of algebraic methods, some approaches, such as the area method (Chou et al., 1996a; Janičić et al., 2012) and the full-angle method (Chou et al., 1996b), represent geometric knowledge with respect to geometric invariants. These methods and some of their implementations are capable of proving a large number of complex geometric theorems and, in many cases, rendering the proofs in readable natural language. In Sects. 4 and 5, we describe this innovation in length.

3 Verification of the Truth of a Geometric Statement

Dynamic geometry programs give users an initial visual validation of a geometric property. Instead of producing a fixed example, these programs produce a large set of examples that reinforce confidence in the truth of a statement. However, given that not all possible cases can be covered, this can be misleading.

GeoGebra has had, since an early version, a validation tool, the “ $a \stackrel{?}{=} b$ ” tool. This tool performs a numerical verification instead of a formal proof, and suffers from the fact that numerical errors may lead to erroneous conclusions. However, since version 5, *GeoGebra* has been enhanced with the support of GATPs. This makes it possible to ask for a formal validation of a given geometric statement (Botana et al., 2015) such as the Midpoint Theorem.

Theorem 1 (Midpoint Theorem) *Let ABC be a triangle, and let D and E be the midpoints of AB and BC respectively. Then the line DE is parallel to the base AC .*

Having made the geometric construction (see Fig. 1), a *GeoGebra* user can check if the conjecture is indeed true. Entering the command “Prove (AreParallel(b, f))” produces the answer “d=true” (see Fig. 1). This is a formal validation generated by *GeoGebra*’s built-in GATPs.

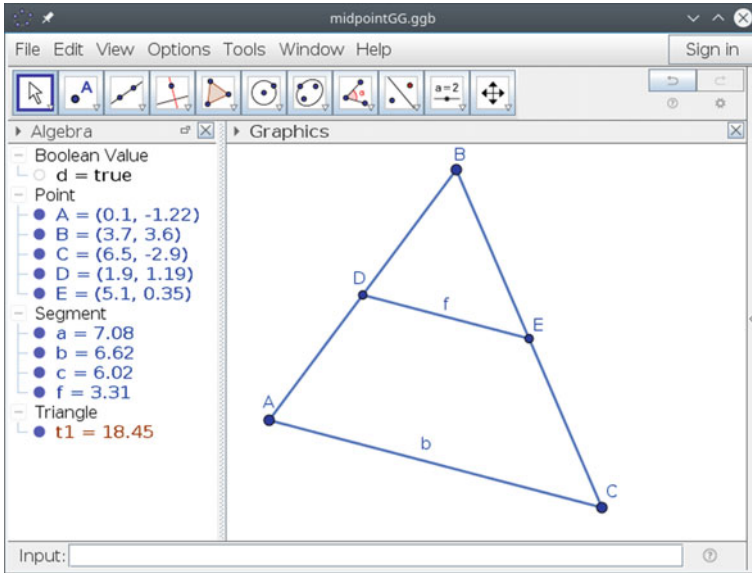


Fig. 1 GeoGebra construction validation

Other DGSs also have integrated GATPs. For example, *GCLC* incorporates *GCLCprover*, which gives the following response to a similar situation (Janičić & Quaresma, 2007):

Deduction check invoked: the property that led to the error will be tested for validity.

Once the conjecture is successfully proved, the critical property always holds. The prover output is written in the file `error-proof.tex`.

A similar approach is used by the DGS/GATP *Java Geometry Expert (JGEX)*⁴ (Ye et al., 2011). It calculates fix-points during the construction; that is, all the properties that can be inferred from the construction to a certain point. When a user tries to perform an illegal construction, the tool says why it is not possible to perform that construction (see Fig. 2).

If we jump to “explanation”—providing insight into why a given statement is true—we need to consider proofs. In the next two sections, we explore the capabilities of current GATPs to produce readable proofs with a natural language and even visual rendering.

⁴<http://www.cs.wichita.edu/~ye/>.

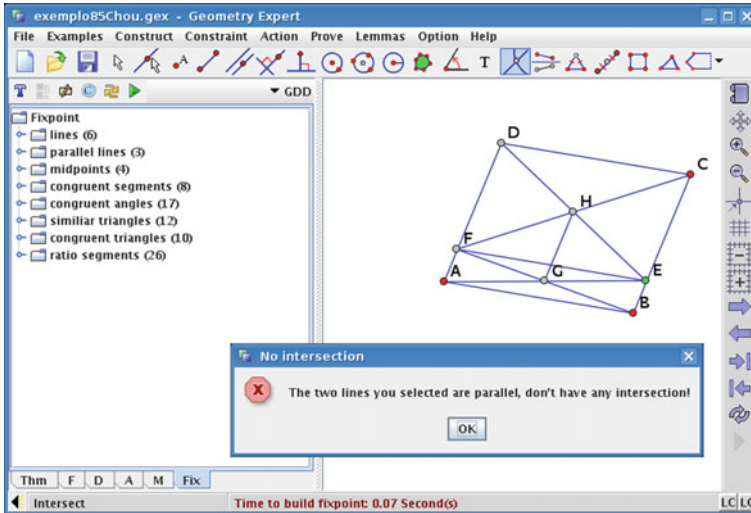


Fig. 2 *JGEX* construction validation, $\overline{FE} \parallel \overline{AB}$

4 Proofs with a Natural Language Rendering

Some of the methods used for geometric automated theorem proving, especially algebraic methods, are not useful in terms of the proofs produced. After a synthetic geometric representation is converted to an algebraic representation, the proofs are all done using complex algebraic reasoning without any connection to the geometric construction. The algorithms implemented in *GeoGebra* are of this type. They produce a time efficient answer to a verification problem, but are useless if the proof itself is the focus.

In (2010a), Ye et al. list the features they feel important to the dynamic visualisation of proofs in geometry. The first of these features is that “The proof text created by the program should be readable, similar to proofs in geometry textbooks or books.”

The semi-synthetic methods like the area method, the full-angle method or the deductive database method, along with the coherent logic based method, are able to produce readable geometric proofs. Implementations like the *GCLC* (area method) (Janičić et al., 2012; Quaresma, Janičić, Tomašević, Vujošević-Janičić, & Tošić, 2008), the *JGEX* (area method, full-angle method, and deductive databases) (Ye et al., 2010a, 2010b, 2011) and the *ArgoCLP*⁵ (coherent logic) (Stojanović et al., 2011) are examples of systems with such capabilities.

The Area Method The area method is a decision procedure for a fragment of Euclidean plane geometry. It deals with problems as sequences of specific geometric construction steps, using a set of specific *geometric quantities* to define relations (Janičić et al., 2012).

⁵<http://argo.matf.bg.ac.rs/?content=downloads>.

Table 1 Expressing geometry predicates in terms of area method geometric quantities

Property	Area method geometric quantities
Points A and B are identical	$\mathcal{P}_{ABA} = 0$
Points A, B, C are collinear	$\mathcal{S}_{ABC} = 0$
AB is perpendicular to CD	$\mathcal{P}_{ABA} \neq 0 \wedge \mathcal{P}_{CDC} \neq 0 \wedge \mathcal{P}_{ACD} = \mathcal{P}_{BCD}$
AB is parallel to CD	$\mathcal{P}_{ABA} \neq 0 \wedge \mathcal{P}_{CDC} \neq 0 \wedge \mathcal{S}_{ACD} = \mathcal{S}_{BCD}$
O is the midpoint of AB	$\mathcal{S}_{ABO} = 0 \wedge \mathcal{P}_{ABA} \neq 0 \wedge \frac{\overline{AO}}{\overline{AB}} = \frac{1}{2}$
AB has the same length as CD	$\mathcal{P}_{ABA} = \mathcal{P}_{CDC}$
Points A, B, C, D are harmonic	$\mathcal{S}_{ABC} = 0 \wedge \mathcal{S}_{ABD} = 0 \wedge \mathcal{P}_{BCB} \neq 0 \wedge \mathcal{P}_{BDB} \neq 0 \wedge \frac{\overline{AC}}{\overline{CB}} = \frac{\overline{DA}}{\overline{DB}}$
Angle ABC has the same measure as DEF	$\mathcal{P}_{ABA} \neq 0 \wedge \mathcal{P}_{ACA} \neq 0 \wedge \mathcal{P}_{BCB} \neq 0 \wedge \mathcal{P}_{DED} \neq 0 \wedge \mathcal{P}_{DFD} \neq 0 \wedge \mathcal{P}_{EFE} \neq 0 \wedge \mathcal{S}_{ABC} \cdot \mathcal{P}_{DEF} = \mathcal{S}_{DEF} \cdot \mathcal{P}_{ABC}$
A and B belong to the same circle arc CD	$\mathcal{S}_{ACD} \neq 0 \wedge \mathcal{S}_{BCD} \neq 0 \wedge \mathcal{S}_{CAD} \cdot \mathcal{P}_{CBD} = \mathcal{S}_{CBD} \cdot \mathcal{P}_{CAD}$

- *Ratio of parallel directed segments*, denoted $\overline{AB}/\overline{CD}$. If the points $A, B, C,$ and D are collinear, $\overline{AB}/\overline{CD}$ is the ratio between the lengths of directed segments AB and CD . If the points $A, B, C,$ and D are not collinear, and it holds $AB \parallel CD$, there is a parallelogram $ABPQ$ such that $P, Q, C,$ and D are collinear and then $\frac{\overline{AB}}{\overline{CD}} = \frac{\overline{QP}}{\overline{CD}}$.
- *Signed area* for a triangle ABC , denoted \mathcal{S}_{ABC} is the area of the triangle ABC , negated if ABC has a negative orientation.
- *Pythagoras difference*,⁶ denoted \mathcal{P}_{ABC} , for the points A, B, C , defined as $\mathcal{P}_{ABC} = \overline{AB}^2 + \overline{CB}^2 - \overline{AC}^2$.

These three geometric quantities allow for expressing (in the form of equalities) geometry properties such as the collinearity of three points, the parallelism of two lines, the equality of two points, the perpendicularity of two lines, and so forth (see Table 1).

The basic objects in an area method conjecture are the points—free points if they are freely placed in the Euclidean plane, and constructed points if they are obtained as the result of a given geometric construction. For example, in the Midpoint Theorem (see Theorem 1) the points A, B and C are free points not defined by construction steps. The constructed points D and E are the midpoints of \overline{AB} and \overline{BC} respectively.

The proof of a conjecture is based on eliminating all the constructed points in reverse order until equality is reached in only the free points. If the equality is provable, then the original conjecture is also a theorem. For Theorem 1, the proof automatically found by *GCLCprover* (see Fig. 3) shows not only the steps leading to

⁶The *Pythagoras difference* is a generalisation of the Pythagorean equality regarding the three sides of a right triangle, to an expression applicable to any triangle (for a triangle ABC with the right angle at B , it holds that $\mathcal{P}_{ABC} = 0$).

- (1) $S_{DEA} = S_{DEC}$, by the statement
- (2) $S_{ADE} = S_{CDE}$, by geometric simplifications
- (3) $\left(S_{ADC} + \left(\frac{1}{2} \cdot (S_{ADB} + (-1 \cdot S_{ADC}))\right)\right) = S_{CDE}$, by Lemma 29 (point E eliminated)
- (4) $\left(\left(\frac{1}{2} \cdot S_{ADC}\right) + \left(\frac{1}{2} \cdot S_{ADB}\right)\right) = S_{CDE}$, by algebraic simplifications
- (5) $\left(\left(\frac{1}{2} \cdot S_{ADC}\right) + \left(\frac{1}{2} \cdot S_{ADB}\right)\right) = \left(S_{CDC} + \left(\frac{1}{2} \cdot (S_{CDB} + (-1 \cdot S_{CDC}))\right)\right)$, by Lemma 29 (point E eliminated)
- (6) $\left(\left(\frac{1}{2} \cdot S_{CAD}\right) + \left(\frac{1}{2} \cdot S_{BAD}\right)\right) = \left(0 + \left(\frac{1}{2} \cdot (S_{BCD} + (-1 \cdot 0))\right)\right)$, by geometric simplifications
- (7) $(S_{CAD} + S_{BAD}) = S_{BCD}$, by algebraic simplifications
- (8) $\left(\left(\left(S_{CAA} + \left(\frac{1}{2} \cdot (S_{CAB} + (-1 \cdot S_{CAA}))\right)\right)\right) + S_{BAD}\right) = S_{BCD}$, by Lemma 29 (point D eliminated)
- (9) $\left(\left(0 + \left(\frac{1}{2} \cdot (S_{CAB} + (-1 \cdot 0))\right)\right) + S_{BAD}\right) = S_{BCD}$, by geometric simplifications
- (10) $\left(\left(\frac{1}{2} \cdot S_{CAB}\right) + S_{BAD}\right) = S_{BCD}$, by algebraic simplifications
- (11) $\left(\left(\frac{1}{2} \cdot S_{CAB}\right) + \left(S_{BAA} + \left(\frac{1}{2} \cdot (S_{BAB} + (-1 \cdot S_{BAA}))\right)\right)\right) = S_{BCD}$, by Lemma 29 (point D eliminated)
- (12) $\left(\left(\frac{1}{2} \cdot S_{CAB}\right) + \left(0 + \left(\frac{1}{2} \cdot (0 + (-1 \cdot 0))\right)\right)\right) = S_{BCD}$, by geometric simplifications
- (13) $\left(\frac{1}{2} \cdot S_{CAB}\right) = S_{BCD}$, by algebraic simplifications
- (14) $\left(\frac{1}{2} \cdot S_{CAB}\right) = \left(S_{BCA} + \left(\frac{1}{2} \cdot (S_{BCB} + (-1 \cdot S_{BCA}))\right)\right)$, by Lemma 29 (point D eliminated)
- (15) $\left(\frac{1}{2} \cdot S_{CAB}\right) = \left(S_{CAB} + \left(\frac{1}{2} \cdot (0 + (-1 \cdot S_{CAB}))\right)\right)$, by geometric simplifications
- (16) $0 = 0$, by algebraic simplifications

Fig. 3 *GCLCprover* output—Proof of Midpoint Theorem

the solution, but also the justification for those steps. *GCLCprover* uses the axioms and rules of inference of the area method to reach the conclusion in 16 steps, of which only 5 are applications of lemmas of the method (e.g., Lemma 29).

The area method implementation in *GCLC* took 0.001 s to prove the Midpoint Theorem. It also found that there are no non-degenerated conditions to this result; that is, it is true in any configuration.

Given that the area method does not follow the normal chain of geometric reasoning used in primary and secondary schools, the resulting proofs are not directly usable by students. However, with the help of teachers, the area method (the full-angle method is similar) could be used at secondary and college levels. This claim needs validation by case studies (see Sect. 6).

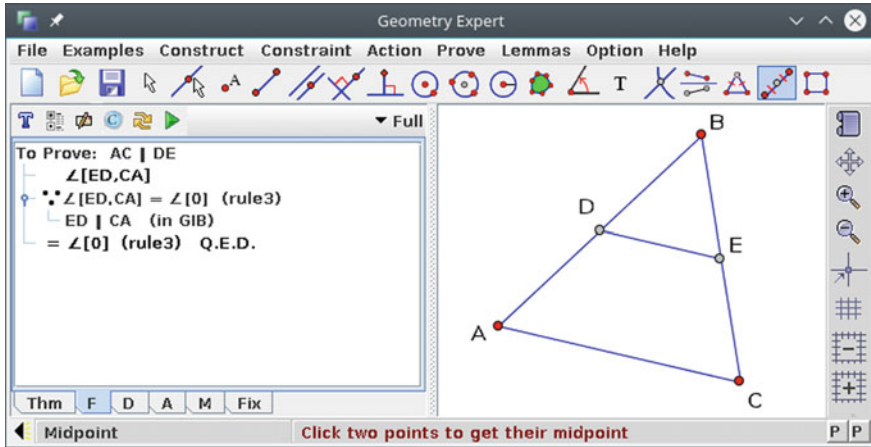


Fig. 4 JGEX screen capture—Proof of Midpoint Theorem

The Full-Angle Method The full-angle method is similar to the area method, but introduces the full-angle as another geometric quantity. Intuitively, a full-angle, $\angle[u, v]$, is the angle from line u to line v (not between segments).

Formally, full-angles can also be defined using the signed area and the Pythagoras difference. A full-angle is defined as an ordered pair of lines which satisfies a set of axioms and rules of inference that constitute the base of the method. The proofs are similar to those using the area method (Chou et al., 1996a).

The JGEX system implements the full-angle method and the deductive database method (based on full-angle rules). Using the full-angle prover, the Midpoint Theorem (see Theorem 1) can be proved (see Fig. 4) in a very concise form. However, the JGEX system does not produce a PDF output file.

Like the area method proofs, the use of full-angle method proofs in primary and secondary levels is limited by the axioms and rules of inference used. JGEX does not produce a proof with a natural language rendering. Unless the reader is familiar with the axiom system and corresponding inference rules, the proofs are somehow difficult to follow.

Coherent Logic Provers A different approach is given by the *ArgoCLP*, a coherent logic-based theorem prover. Coherent logic is a fragment of first-order logic where the conjectures can be proved directly (i.e., not by refutation). Proofs in coherent logic are natural and intuitive, and reasoning is constructive (Bezém & Coquand, 2005). Coherent logic is therefore a suitable framework for producing both readable and formal proofs. The *ArgoCLP* automatically and simultaneously generates traditional human readable proofs and formal proofs of geometry theorems (for various axiom systems). The generated step-by-step proofs are very similar to the proofs given in standard geometry textbooks. Unfortunately, however, efficiency issues prevent its use in education (Stojanović et al., 2011, pp. 13–14).

5 Proofs with Natural Language and Visual Rendering

Given the current implementations in geometry automated theorem proving, it is now possible to begin work on a new approach: proofs that interrelate geometric elements in the proof text and the construction.

As Ye et al. (2010a) show in their features of visually dynamic proofs in geometry, effective connection between the visual rendering and textual rendering is key:

The displays of the proof text and the geometry elements in the construction are separated, but should be internally related. By clicking a step or a part of a step of the proof’s text, the corresponding geometric elements in the construction should be highlighted using various dynamic visual effects.

They propose visual effects such as translations, rotations, reflections, and scaling to show congruence and similarities between geometric elements. Colours and blinking effects can be used to show the relations between the two renderings of a proof.

As discussed above (see Sect. 4), natural language rendering is possible and usable with some limitations in different contexts. Is it possible to connect natural language rendering with a formal proof and visual rendering? In the following, we examine the strengths and challenges of different approaches.

5.1 Area Method Visual Rendering of Proofs

As mentioned above (see Sect. 4), the area method deals with problems stated as sequences of specific geometric construction steps, using a set of specific *geometric quantities* (Chou et al., 1996a, 1996b; Janičić et al., 2012). In the following, we establish a connection between the axioms and inference rules of this method. Using this connection makes it possible to synchronize a formal proof, a natural language rendering, and a visual rendering.

Visual Counterparts of Geometric Quantities The *ratio of parallel directed segments* and the *signed area* have a clear visual counterpart. The *Pythagoras difference* needs the support of the algebraic quantity $\overline{AB}^2 + \overline{CB}^2 - \overline{AC}^2$ (see Fig. 5).

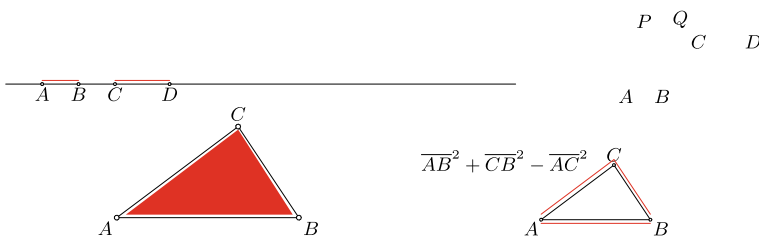
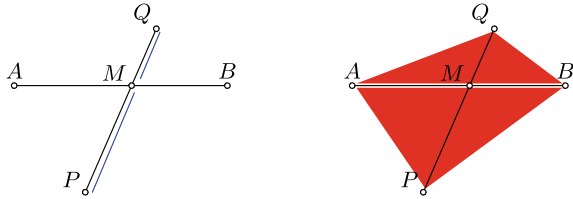


Fig. 5 Illustration for area method geometric quantities

Fig. 6 Illustration for elimination Lemma 1:
 $\frac{PM}{QM} = \frac{S_{PAB}}{S_{QAB}}$



Axioms and Lemmas of the Area Method The area method uses a large set of lemmas that characterise the geometric quantities and facilitate structuring the proofs (Chou et al., 1996a; Janičić et al., 2012; Quaresma & Janičić, 2009). Many of these lemmas are about technical issues related to the formal proofs; for example, how two triangles that differ only in the order of their vertices can be considered the same. The most important for area method proofs and also for their visualisation are the elimination lemmas.

In the following, we introduce two of these lemmas and present their visual counterparts.

Theorem 5.1 (Elimination Lemma 1—The Co-side Theorem) *Let M be the intersection of two non-parallel lines AB and PQ and Q ≠ M. Then it holds that $\frac{PM}{QM} = \frac{S_{PAB}}{S_{QAB}}$, $\frac{PM}{PQ} = \frac{S_{PAB}}{S_{PAQB}}$, $\frac{QM}{PQ} = \frac{S_{QAB}}{S_{PAQB}}$ (Fig. 6).*

Theorem 5.2 (Elimination Lemma 3 (reduced version)) *Let S_{ABY} be the ΔABY signed area where point Y is the intersection of lines UV and PQ. Then it holds that (Fig. 7)*

$$S_{ABY} = \frac{S_{UPQ}S_{ABV} - S_{VPQ}S_{ABU}}{S_{UPVQ}}$$

Area Method Visual Proof—Midpoint Theorem Using the visual counterparts of the lemmas of the area method we can revisit the Midpoint Theorem (see Theorem 1).

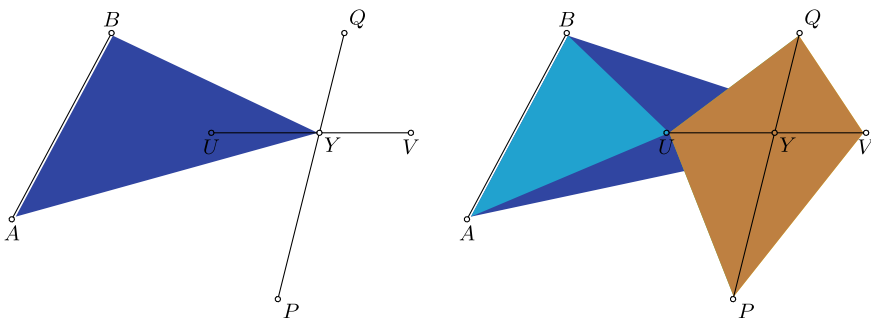


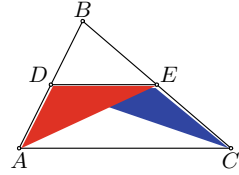
Fig. 7 Illustration for elimination Lemma 3: $S_{ABY} = \frac{S_{UPQ}S_{ABV} - S_{VPQ}S_{ABU}}{S_{UPVQ}}$

This time the *GCLCprover* area method proof has two parts: a natural language part and a visual counterpart.

(1–2) by the statement and geometric simplifications

$$S_{DEA} = S_{DEC}$$

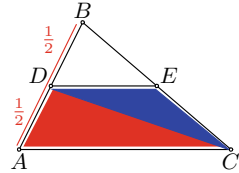
$$S_{ADE} = S_{CDE}$$



(3–4) by Lemma 29 (point *E* eliminated) and algebraic simplifications

$$S_{ADC} + \frac{1}{2}(S_{ADB} - S_{ADC}) = S_{CDE}$$

$$\frac{1}{2}S_{ADC} + \frac{1}{2}S_{ADB} = S_{CDE}$$

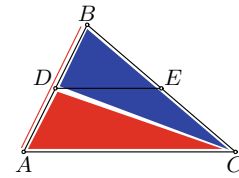


(5–7) by Lemma 29 (point *E* eliminated), geometric and algebraic simplifications

$$\frac{1}{2}S_{ADC} + \frac{1}{2}S_{ADB} = S_{CDC} + \frac{1}{2}(S_{CDB} - S_{CDC})$$

$$\frac{1}{2}S_{CAD} + \frac{1}{2}S_{BAD} = 0 + \frac{1}{2}(S_{BCD} - 0)$$

$$S_{CAD} + S_{BAD} = S_{BCD}$$

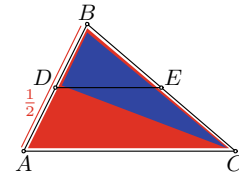


(8–10) by Lemma 29 (point *D* eliminated), geometric and algebraic simplifications

$$S_{CAA} + \frac{1}{2}(S_{CAB} - S_{CAA}) + S_{BAD} = S_{BCD}$$

$$0 + \frac{1}{2}(S_{CAB} - 0) + S_{BAD} = S_{BCD}$$

$$\frac{1}{2}S_{CAB} + S_{BAD} = S_{BCD}$$

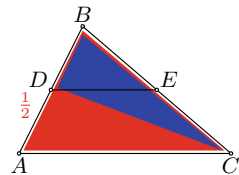


(11–13) by Lemma 29 (point *D* eliminated), geometric and algebraic simplifications

$$\frac{1}{2}S_{CAB} + S_{BAA} + \frac{1}{2}(S_{BAB} - S_{BAA}) = S_{BCD}$$

$$\frac{1}{2}S_{CAB} + 0 + \frac{1}{2}(0 - 0) = S_{BCD}$$

$$\frac{1}{2}S_{CAB} = S_{BCD}$$

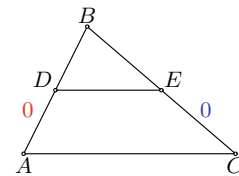


(14–16) by Lemma 29 (point *D* eliminated), geometric and algebraic simplifications

$$\frac{1}{2}S_{CAB} = S_{BCA} + \frac{1}{2}(S_{BCB} - S_{BCA})$$

$$\frac{1}{2}S_{CAB} = S_{CAB} + \frac{1}{2}(0 - S_{CAB})$$

$$0 = 0 \quad \text{Q.E.D.}$$



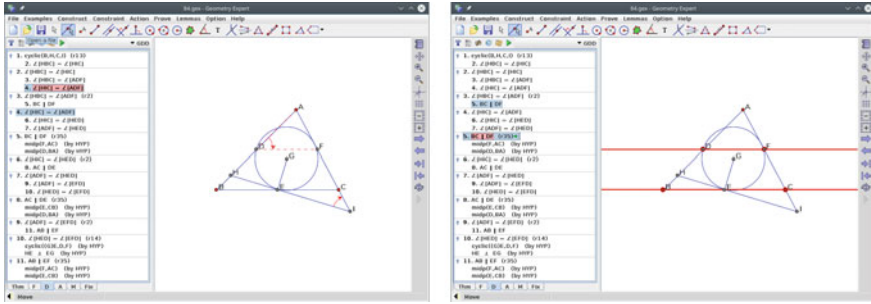


Fig. 8 JGEX—Example 84, Steps 2 and 5

The example illustrates how to express a problem using the given geometric quantities and how to prove it in a concise, easily understood way with a corresponding visual rendering.

As far as we know, there are no systems for the area method that provide such a connection. It is not yet a feature of *GCLC* (see Sect. 6). However, *JGEX* (Ye et al., 2010a, 2010b, 2011) does provide this connection for the full-angle method.

5.2 Full-Angle Method Visual Proofs

Using the *JGEX* system, we can build a given construction, state a conjecture about it, and then, using one of the built-in GATPs, prove it. Using the full-angle method based GATP we can produce examples where the formal proof has a visual counterpart.

Figure 8 was taken from the tool’s own set of examples. Clicking on a step of the formal proof produces a visual animation of the step on the construction. The related relations between objects on the construction—e.g. the angles between two lines in Fig. 8 (left)—initially “blink.” They then become fixed, but use colours to clearly show the corresponding relations in the formal proof.

In both situations, the drawback is that neither the area method nor the full-angle method use the usual set of axioms and rules of inference of primary and secondary school geometry. Instead, they use the geometric quantities *ratio of parallel directed segments*, *signed area*, *Pythagoras difference* and *full-angle*, and the axioms and rules of inference for these geometric quantities. Using these methods in secondary schooling could prove difficult.

6 Conclusions and Future Work

Geometry, with its very strong and appealing visual content and its formal axiomatic theory, is a privileged domain. It is an area where computational tools can significantly enhance the learning environment and make students active in building their knowledge.

Introducing dynamic geometry systems enhances exploration and proof by making it easy to posit and test conjectures. The ability to make, properties-preserving, manipulations enables students to explore “visual proofs” of geometric conjectures. DGSs significantly help students to acquire knowledge about geometric objects and, more generally, to acquire mathematical rigour.

Geometry automated theorem provers capable of construction validation and human readable proofs consolidate the knowledge acquired with the use of DGSs. If the GATP produces synthetic proofs, the proof of a conjecture or the proof of the soundness of a construction can be used as an object of study providing a logical explanation. With a DGS, students can visually explore constructions or check that certain conjectures are true. However, these systems do not provide mathematical arguments for the conclusions they produce. Instead of producing a “visual check,” a GATP can be used to draw accurate mathematical conclusions. Thus, we claim that GATPs can be used in the learning process (Janičić & Quaresma, 2007; Quaresma & Janičić, 2006; Santos & Quaresma, 2012, 2013).

The natural language rendering of a formal proof, especially if paired with a visual rendering, would allow for a wider application of GATPs in learning contexts. This is still an active area of research involving exploration of different methods, implementations, and renderings. Currently, we are exploring the construction of a controlled hybrid language for geometry; that is, a pair of controlled languages (natural and visual) with common semantics. By considering figures as sentences in a visual language sharing semantics with the natural language of geometric statements, we can achieve interaction between parts of text and corresponding figures. We can connect formal proofs to natural language and visual descriptions. This approach is more generic than the concrete cases described above. It is a line of research we are already pursuing.⁷

Given the availability of GATP technology and the work currently being done on the rendering of the proofs, it is important to begin integrating these advances in learning environments. As authors, we have already developed the *Web Geometry Laboratory* (WGL),⁸—an adaptive and collaborative blended-learning Web environment that integrates a dynamic geometry system (Quaresma, Santos, & Bouallegue, 2013; Quaresma, Santos, & Marić, 2018; Santos, Quaresma, Marić, & Campos, 2018). A short/medium term goal of the WGL is to connect with the *OpenGeoProver* (OGP)⁹ (Baeta & Quaresma, 2013), an open library of GATPs that we are currently developing alongside with others researchers. The connection between WGL and OGP will provide the lab with the automatic deduction capabilities discussed in this chapter; namely, checking the validation of a construction and getting formal proofs with a human readable and visual rendering (Quaresma & Santos, 2016). This and other projects like the integration of GATPs in *GeoGebra* (Botana et al., 2015) aim to

⁷Haralambous, Yannis and Quaresma, Pedro, *Geometric Statements as Controlled Hybrid Language Sentences, an Example* in preparation.

⁸<http://hilbert.mat.uc.pt/WebGeometryLab>.

⁹<https://github.com/ivan-z-petrovic/open-geo-prover>.

include the power of automatic deduction in the learning and teaching of geometry. Ultimately, our goal is to help teachers and students answer the question “why.”

References

- Abánades, M., Botana, F., Kovács, Z., Recio, T., & Sólyom-Gecse, C. (2016). Towards the automatic discovery of theorems in GeoGebra. In G. M. Greuel, T. Koch, P. Paule, & A. Sommese (Eds.), *Mathematical Software—ICMS 2016* (pp. 37–42). Cham: Springer International Publishing.
- Baeta, N., & Quaresma, P. (2013). The full angle method on the OpenGeoProver. In C. Lange, D. Aspinall, J. Carette, J. Davenport, A. Kohlhase, M. Kohlhase, P. Libbrecht, P. Quaresma, F. Rabe, P. Sojka, I. Whiteside, & W. Windsteiger (Eds.), *MathUI, OpenMath, PLMMS and ThEdu Workshops and Work in Progress at the Conference on Intelligent Computer Mathematics*, no. 1010 in CEUR Workshop Proceedings. Aachen. <http://ceur-ws.org/Vol-1010/paper-08.pdf>.
- Bezém, M., & Coquand, T. (2005). Automating coherent logic. In G. Sutcliffe & A. Voronkov (Eds.), *Logic for programming, artificial intelligence, and reasoning*. Lecture Notes in Computer Science (Vol. 3835, pp. 246–260). Berlin/Heidelberg: Springer. https://doi.org/10.1007/11591191_18.
- Botana, F., Hohenwarter, M., Janičić, P., Kovács, Z., Petrović, I., Recio, T., et al. (2015). Automated theorem proving in GeoGebra: Current achievements. *Journal of Automated Reasoning*, 55(1), 39–59. <https://doi.org/10.1007/s10817-015-9326-4>.
- Chou, S. (1985). Proving and discovering geometry theorems using Wu’s method. Ph.D. thesis, The University of Texas, Austin.
- Chou, S. C., & Gao, X. S. (2001). Automated reasoning in geometry. In J. A. Robinson & A. Voronkov (Eds.), *Handbook of automated reasoning* (pp. 707–749). Elsevier Science Publishers B.V.
- Chou, S. C., Gao, X. S., & Zhang, J. Z. (1994). *Machine proofs in geometry*. World Scientific.
- Chou, S. C., Gao, X. S., & Zhang, J. Z. (1996a). Automated generation of readable proofs with geometric invariants, I. Multiple and shortest proof generation. *Journal of Automated Reasoning*, 17(13), 325–347. <https://doi.org/10.1007/BF00283133>.
- Chou, S. C., Gao, X. S., & Zhang, J. Z. (1996b). Automated generation of readable proofs with geometric invariants, II. Theorem proving with full-angles. *Journal of Automated Reasoning*, 17(13), 349–370. <https://doi.org/10.1007/BF00283134>.
- Hanna, G. (2000). Proof, explanation and exploration: An overview. *Educational Studies in Mathematics*, 44(1–2), 5–23. <https://doi.org/10.1023/A:1012737223465>.
- Hanna, G., & Sidoli, N. (2007). Visualisation and proof: A brief survey of philosophical perspectives. *ZDM*, 39(1–2), 73–78. <https://doi.org/10.1007/s11858-006-0005-0>.
- Hohenwarter, M. (2002). Geogebra—A software system for dynamic geometry and algebra in the plane. Master’s thesis, University of Salzburg, Austria.
- Hoyle, C., & Jones, K. (1998). Proof in dynamic geometry contexts. In *Perspectives on the teaching of geometry for the 21st century* (pp. 121–128). Springer. <https://eprints.soton.ac.uk/41227/>.
- Jančić, P. (2006). GCLC—A tool for constructive Euclidean geometry and more than that. In A. Iglesias & N. Takayama (Eds.) *Mathematical Software—ICMS 2006*. Lecture Notes in Computer Science (Vol. 4151, pp. 58–73). Springer. https://doi.org/10.1007/11832225_6.
- Jančić, P., Narboux, J., & Quaresma, P. (2012). The area method: A recapitulation. *Journal of Automated Reasoning*, 48(4), 489–532. <https://doi.org/10.1007/s10817-010-9209-7>.
- Jančić, P., & Quaresma, P. (2006). System description: GCLCprover + GeoThms. In U. Furbach, N. Shankar (Eds.), *Automated reasoning*. Lecture Notes in Computer Science (Vol. 4130, pp. 145–150). Springer. https://doi.org/10.1007/11814771_13.
- Jančić, P., & Quaresma, P. (2007). Automatic verification of regular constructions in dynamic geometry systems. In F. Botana & T. Recio (Eds.), *Automated deduction in geometry*. Lecture

- Notes in Computer Science (Vol. 4869, pp. 39–51). Springer. https://doi.org/10.1007/978-3-540-77356-6_3.
- Jiang, J., & Zhang, J. (2012). A review and prospect of readable machine proofs for geometry theorems. *Journal of Systems Science and Complexity*, 25(4), 802–820. <https://doi.org/10.1007/s11424-012-2048-3>.
- Kapur, D. (1986). Using Gröbner bases to reason about geometry problems. *Journal of Symbolic Computation*, 2(4), 399–408. [https://doi.org/10.1016/S0747-7171\(86\)80007-4](https://doi.org/10.1016/S0747-7171(86)80007-4).
- Kovács, Z. (2015). Computer based conjectures and proofs in teaching Euclidean geometry. Ph.D. thesis, Universität Linz. urn:nbn:at:at-ubl:1-5034.
- Kovács, Z. (2015). *The relation tool in GeoGebra* (Vol. 5, pp. 53–71). Springer International Publishing. https://doi.org/10.1007/978-3-319-21362-0_4.
- Li, H. (2000). Clifford algebra approaches to mechanical geometry theorem proving. In X. S. Gao & D. Wang (Eds.), *Mathematics mechanization and applications* (pp. 205–299). San Diego, CA: Academic Press.
- Lin, F. L., Hsieh, F. J., Hanna, G., & de Villiers, M. (Eds.). (2009a). *Proceedings of the ICMI Study 19 Conference: Proof and Proving in Mathematics Education* (Vol. 1). The Department of Mathematics: National Taiwan Normal University.
- Lin, F. L., Hsieh, F. J., Hanna, G., & de Villiers, M. (Eds.). (2009b). *Proceedings of the ICMI Study 19 conference: Proof and Proving in Mathematics Education* (Vol. 2). The Department of Mathematics: National Taiwan Normal University.
- Nelsen, R. B. (1993). *Proofs without words: Exercises in visual thinking* (Vol. 1). MAA.
- Paneque, J., Cobo, P., Fortuny, J., & Richard, P. R. (2016). Argumentative effects of a geometric construction tutorial system in solving problems of proof. In: *Proceedings of the 4th International Workshop on Theorem Proving Components for Educational Software*, July 15, 2015, Washington, D.C., USA. CISUC Technical Reports (Vol. 2016-001, pp. 13–35). CISUC.
- Quaresma, P. (2017). Towards an intelligent and dynamic geometry book. *Mathematics in Computer Science*, 11(3), 427–437. <https://doi.org/10.1007/s11786-017-0302-8>.
- Quaresma, P., & Janičić, P. (2006). Integrating dynamic geometry software, deduction systems, and theorem repositories. In J. M. Borwein & W. M. Farmer (Eds.), *Mathematical knowledge management*. Lecture Notes in Computer Science (Vol. 4108, pp. 280–294). Berlin: Springer. https://doi.org/10.1007/11812289_22.
- Quaresma, P., & Janičić, P. (2009). The area method, rigorous proofs of lemmas in Hilbert's style axiom system. Tech. Rep. 2009/006, Centre for Informatics and Systems of the University of Coimbra.
- Quaresma, P., Janičić, P., Tomašević, J., Vujošević-Janičić, M., & Tošić, D. (2008). Communicating mathematics in the digital era. In *XML-bases format for descriptions of geometric constructions and proofs* (pp. 183–197). Wellesley, MA: A. K. Peters, Ltd.
- Quaresma, P., & Santos, V. (2016). Visual geometry proofs in a learning context. In W. Neuper & P. Quaresma (Eds.), *Proceedings of ThEdu'15, CISUC Technical Reports* (Vol. 2016001, pp. 1–6). CISUC. <https://www.cisuc.uc.pt/ckfinder/userfiles/files/TR2016-01.pdf>.
- Quaresma, P., Santos, V., & Bouallegue, S. (2013). The Web Geometry Laboratory project. In J. Carette, D. Aspinall, C. Lange, P. Sojka & W. Windsteiger (Eds.), *CICM 2013*. Lecture Notes in Computer Science (Vol. 7961, pp. 364–368). Springer. https://doi.org/10.1007/978-3-642-39320-4_30.
- Quaresma, P., Santos, V., & Marić, M. (2018). WGL, a web laboratory for geometry. *Education and Information Technologies*, 23(1), 237–252. <https://doi.org/10.1007/s10639-017-9597-y>.
- Recio, T., & Vélez, M. P. (2012). *An introduction to automated discovery in geometry through symbolic computation* (pp. 257–271). Vienna: Springer. https://doi.org/10.1007/978-3-7091-0794-2_12.
- Richard, P. R., Oller Marcén, A. M., & Meavilla Seguí, V. (2016). The concept of proof in the light of mathematical work. *ZDM*, 48(6), 843–859. <https://doi.org/10.1007/s11858-016-0805-9>.
- Richter-Gebert, J., & Kortenkamp, U. (1999). *The interactive geometry software Cinderella*. Springer.

- Ruthven, K., Hennessy, S., & Deaney, R. (2008). Constructions of dynamic geometry: A study of the interpretative flexibility of educational software in classroom practice. *Computers & Education*, 51(1), 297–317.
- Santos, V., & Quaresma, P. (2012). Integrating DGSs and GATPs in an adaptative and collaborative blended-learning Web-environment. In *First Workshop on CTP Components for Educational Software (THedu'11), EPTCS* (Vol. 79, pp. 111–123). <https://doi.org/10.4204/EPTCS.79.7>.
- Santos, V., & Quaresma, P. (2013). Collaborative aspects of the WGL project. *Electronic Journal of Mathematics & Technology*, 7(6). Mathematics and Technology, LLC.
- Santos, V., Quaresma, P., Marić, M., & Campos, H. (2018). Web geometry laboratory: Case studies in Portugal and Serbia. *Interactive Learning Environments*, 26(1), 3–21. <https://doi.org/10.1080/10494820.2016.1258715>.
- Stojanović, S., Narboux, J., Bezem, M., & Janičić, P. (2014). A vernacular for coherent logic. In S. M. Watt, J. Davenport, A. Sexton, P. Sojka, & J. Urban (Eds.), *Intelligent computer mathematics. Lecture Notes in Computer Science* (Vol. 8543, pp. 388–403). Springer International Publishing. https://doi.org/10.1007/978-3-319-08434-3_28.
- Stojanović, S., Pavlović, V., & Janičić, P. (2011). A coherent logic based geometry theorem prover capable of producing formal and readable proofs. In P. Schreck, J. Narboux, & J. Richter-Gebert (Eds.), *Automated deduction in geometry. Lecture Notes in Computer Science* (Vol. 6877, pp. 201–220). Berlin, Heidelberg: Springer. https://doi.org/10.1007/978-3-642-25070-5_12.
- de Villiers, M. (2006). Some pitfalls of dynamic geometry software. *Learning and Teaching Mathematics*, 2006(4), 46–52.
- Wang, D. (1995). Reasoning about geometric problems using an elimination method. In J. Pfalzgraf & D. Wang (Eds.), *Automated practical reasoning* (pp. 147–185). New York: Springer.
- Wang, K., & Su, Z. (2015). Automated geometry theorem proving for human-readable proofs. In *Proceedings of the 24th International Conference on Artificial Intelligence, IJCAI'15* (pp. 1193–1199). AAAI Press. <http://dl.acm.org/citation.cfm?id=2832249.2832414>.
- Wu, W. (1984). On the decision problem and the mechanization of theorem proving in elementary geometry. In *Automated theorem proving: After 25 years* (Vol. 29, pp. 213–234). American Mathematical Society.
- Ye, Z., Chou, S. C., & Gao, X. S. (2010a). Visually dynamic presentation of proofs in plane geometry, Part 1. *Journal of Automated Reasoning*, 45, 213–241. <https://doi.org/10.1007/s10817-009-9162-5>.
- Ye, Z., Chou, S. C., & Gao, X. S. (2010b). Visually dynamic presentation of proofs in plane geometry, Part 2. *Journal of Automated Reasoning*, 45, 243–266. <https://doi.org/10.1007/s10817-009-9163-4>.
- Ye, Z., Chou, S. C., & Gao, X. S. (2011). An introduction to java geometry expert. In T. Sturm & C. Zengler (Eds.), *Automated deduction in geometry. Lecture Notes in Computer Science* (Vol. 6301, pp. 189–195). Berlin, Heidelberg: Springer. https://doi.org/10.1007/978-3-642-21046-4_10.

Using 3D Geometry Systems to Find Theorems of Billiard Trajectories in Polyhedra



Heinz Schumann

Billiards in Euclidean spaces of dimension three or more are essentially in an infantile state. In brief: practically nothing is known in general.
–Berger (2010)

1 Introduction

The topic of this chapter belongs to the complex and developing theory of mathematical billiards. The use of 3D dynamic geometry systems (D3DGS) opens new topics in spatial geometry. One of these topics is billiards in convex polyhedra. Discovering distinctive billiards trajectories in a cube and its generalizations is suitable for spatial geometry activities beyond regular classroom lessons.

The basic literature comprises Tabachnikov's study of billiards on a plane in *Geometrie und Billard* (2013), and Galperin and Semliakov's work on spatial billiards in *Mathematische Billarde* (1990). The only sufficient and non necessary conditions known for billiards trajectories in polygons or polyhedra are that these trajectories exist in convex polygons or convex polyhedra whose sides with a common vertex or whose faces with a common edge enclose angles that are a rational multiple of π (180°).

The following general problem formulation can be used for a directed theorem discovery at school or undergraduate level:

For which special convex $2n$ -faced polyhedra is there a billiards trajectory traversing all faces that takes the shape of a regular $2n$ -gon. And vice versa: Which $2n$ -faced polyhedron can be constructed for a given regular spatial $2n$ -gon that represents a billiards trajectory?

H. Schumann (✉)

University of Education Weingarten, Kirchplatz 2, 88250 Weingarten, Germany
e-mail: schumann@ph-weingarten.de

© Springer Nature Switzerland AG 2019

G. Hanna et al. (eds.), *Proof Technology in Mathematics Research and Teaching*, Mathematics Education in the Digital Era 14,
https://doi.org/10.1007/978-3-030-28483-1_12

255

This problem formulation is equivalent to the following:

For which special convex $2n$ -faced polyhedra is there an inscribed $2n$ -gon of minimal perimeter that is regular. And vice versa: Which $2n$ -faced polyhedron can be constructed upon a given regular spatial $2n$ -gon as an inscribed polygon of minimal perimeter?

To provide an example: A regular spatial $2n$ -gon is the Petrie polygon of an n -gonal antiprism formed by regular congruent base areas, one of which is turned by $180^\circ/n$ (π/n). The convex hull of this regular spatial $2n$ -gon is the corresponding n -gonal antiprism. There can be no spatial n -gons with an odd number of vertices.

In the following sections, the given problem settings will be pursued for $n = 3$. The term cushion is used to describe the impact of a billiard with a reflection of the (point-shaped) billiards ball or an atomic particle or light corpuscle on the plane, and largely avoid the specialist terms of the theory of mathematical billiards.

By means of a suitable 3D dynamic geometry system (D3DGS) such as Cabri 3D (Bainville & Laborde, 2004–2015), billiards moves can be dynamically examined for simple polyhedra such as cubes, cuboids, parallelepipeds, and so forth. This creates elementary geometric access. Finding assertions about these billiards in a D3DGS is medially supported by the visualization, construction, measurement, and direct manipulation and variation of spatial objects (Schumann, 2007). Further, the use of a D3DGS reinforces the role of heuristic strategies in discovery; namely (see Winter, 1989; Borwein, 2012):

- the formation of analogies (especially from plane to spatial geometry)
- induction
- restructuring and variation
- generalization and specification
- analysis and synthesis
- composition
- reversibility.

Comment 1: These heuristic strategies can be expanded to include “case discrimination.”

Confidence in the validity of conjectures produced by 3D dynamic geometry software such as Cabri 3D diminishes the need for proof as a means of *verification*. However, utilizing students’ natural desire for explanation – to determine *causality* (Harel, 2013) – can instead, as argued by De Villiers (1990, 2010) and Hanna (2000), motivate proof as a means of *explanation* (or as systematization or discovery). In other words, a pedagogical paradigm shift from the traditional focus of proof only as a means of verification to a broader view of proof serving many other functions in mathematics is therefore required.


It is also conceivable that using dynamic 3D geometry software can help students progress through the Van Hiele-like levels of understanding 3D geometry identified by Gutierrez et al. (2004). To explore this conjecture and also the trajectories of 3D billiards outlined in this chapter, research is needed on the use of 3D geometry software with students. Such follow-up research would assist in a better understanding not only of students’ conceptualization of 3D, but also of their ability to construct

proofs for their observations. This focus, however, falls outside the scope of this chapter and is identified as an important area for future research.

The chapter will now examine the case of billiards in a square, then extending it to billiards in cubes by using the analogy between squares and cubes. Some properties and specific shapes of the hexagonal billiard trajectories (or paths) in cubes will be discussed. Next, these findings are generalized to equilateral parallelepipeds, and some point symmetric polygons that are formed. In the last two sections, the heuristic process is firstly reversed by using the hexagon to construct the hexahedron for which the hexagon is a billiards trajectory, and secondly, special tetrahedrons are identified along inscribed symmetrical 3D quadrangles results. Note that the polygonal billiards trajectories (paths) investigated here for polyhedra, are spatial ones.

2 From Square Billiards to Cube Billiards

Preliminary notes:

1. The dragging hand () symbolizes the potential to vary a shape from the position of a point (In the following, the property of a D3DGS to vary parameters of geometric figures by dragging corresponding points is not explicitly explained. However, the interactive dynamic Cabri 3D sketches for the investigations below can be downloaded from 'Springer Extra Materials' at <http://extras.springer.com/> A demo, trial version of Cabri 3D can be downloaded for free from <https://cabri.com/en/student/cabri-3d/-Trial>).
2. To provide a better visual perception of the shape properties that follow, the virtual space is displayed in parallel projection.
3. Print media can give only a poor impression of the interactive and dynamic work process, and the visualization possibilities.

2.1 Elementary Billiards Trajectories

Figure 1 shows the given positions S and T of two balls and the cushion g. *How does the ball in S need to be pushed in order to make it reflect on g in such a way that it hits the ball in T?*

The billiards trajectory hits g in R so that both trajectories enclose the same (non-obtuse) reflection angle (angle of incidence = angle of reflection). R is the intersection point of g with a straight line connecting T and the mirror image S' of S with respect to g. This elementary billiards trajectory is simultaneously the shortest polygonal chain connecting S and T that passes a point on g, as the triangle inequality ensures that $|SPT| > |SRT|$ for every point P different from R on g. This holds true analogously in space for the billiards trajectory from S to T with reflection on the plane e (Fig. 2). The straight lines RT and RS are symmetrically identical with regard

Fig. 1 Elementary billiards trajectory on the plane

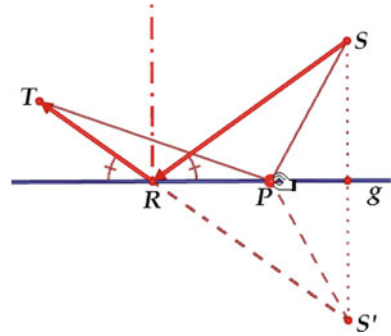
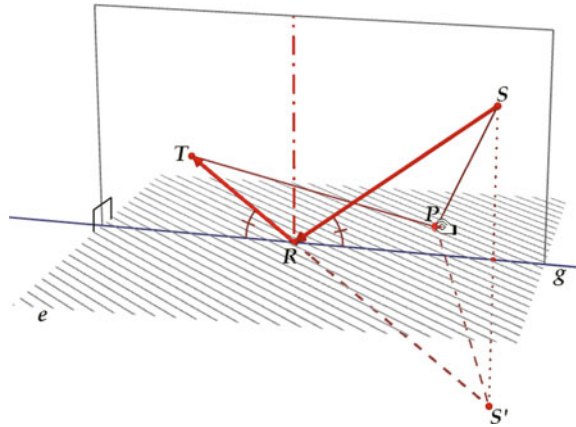


Fig. 2 Elementary billiards trajectory in space with reflection on plane



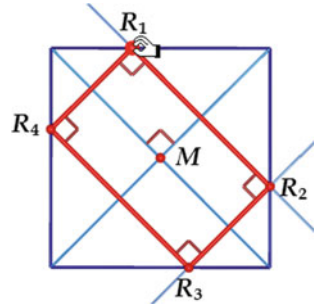
to the perpendicular in R on g and e respectively (Figs. 1 and 2). This property can be used, for example, to check if SRT is a billiards trajectory.

2.2 From Billiards Trajectories in a Square to the Construction of Billiards Trajectories in a Cube via Analogy Formation

It is well known that the simple closed billiards trajectory in a square consists of a rectangle $R_1R_2R_3R_4$ (Fig. 3) symmetrical to the square's centre M, whose opposing sides are each parallel to one of the square's diagonals and whose perimeter is double a square diagonal. The billiards rectangles are thus all of the same perimeter. The angle between two adjunct sides of a rectangle is equal to the angle between the square diagonals.

A billiards rectangle with the starting point R_1 is best constructed by drawing a parallel line to one of the square's diagonals through R_1 . It cuts an adjunct side

Fig. 3 Billiards quadrangle in the square



of the square in R_2 . Another parallel to the second diagonal of the square is drawn through this point, resulting in R_3 as an intersection point with another square side. Reflection of the polygonal chain $R_1R_2R_3$ to the square centre M completes the billiards rectangle $R_1R_2R_3R_4$.

It appears logical to construct billiards trajectories on the cube in analogy to the billiards rectangle. For that, we may, for example, start from a point R_1 on one face of the cube (Fig. 4) and construct a parallel to the space diagonal with the end point A . The parallel intersects one of the faces in the point R_2 . We construct a parallel to the space diagonal with end point B through this point, which intersects another face of the cube in R_3 . The parallel to the space diagonal with the end point C through R_3 intersects another cube face in R_4 . The reflection of the polygonal chain $R_1R_2R_3R_4$ on the cube centre M results in a spatial billiards hexagon $R_1R_2R_3R_4R_5R_6$. Likewise, there is another, different billiards hexagon for the starting point R_1 (Fig. 5). In space, these situations are generally more comprehensive than on a plane. A review of all potential construction cases shows that, aside from particular positions of the starting vertex R_1 , there are always only two different hexagons for the same starting vertex.

Fig. 4 Spatial billiards hexagon in a cube type 1

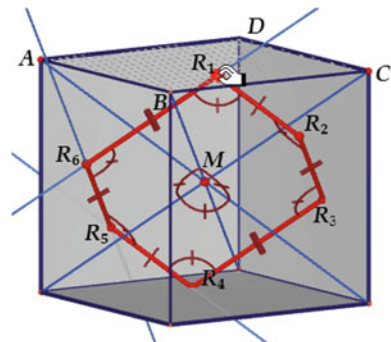
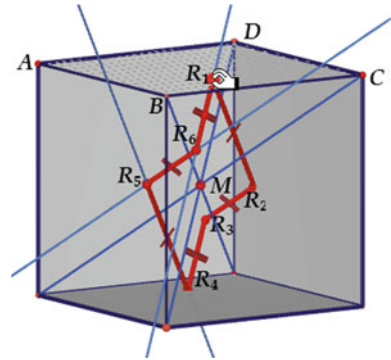


Fig. 5 Spatial billiards hexagon in a cube type 2



2.3 Properties of the Billiards Hexagons

The perimeters of the hexagons that are symmetrical to the cube centre equal twice the length of the space diagonal. This means that all billiards hexagons have the same perimeter. The hexagon has equal interior angles, which are equal to the obtuse angle between the diagonals. The billiards hexagon is thus an equiangular hexagon whose interior angle size remains constant regardless of form changes.

For determination: The given construction of the type 1 hexagon is valid for any starting point R_1 within the triangle ABC (Fig. 6). If the starting point R_1 of the construction lies within the diagonal AB, the hexagon shrinks to a parallelogram. The same applies to the billiards hexagon of type 2 with a starting point R_1 within the triangle BCD. A perpendicular 90° rotation through the centre of the square ABCD projects the generating space diagonals of the type 1 hexagon onto the generating space diagonals of the type 2 hexagon. Each type 1 hexagon ($R_1R_2R_3R_4R_5R_6$) thus transforms into a type 2 hexagon ($R_1'R_2'R_3'R_4'R_5'R_6'$) and vice versa (Fig. 7). A differentiation between hexagons of type 1 and type 2 is thus only relevant if a billiards trajectory is to be constructed from the same starting point. Billiards hexagons with starting points on the other five faces can be constructed by rotation and/or reflection

Fig. 6 Type 1 billiards hexagon with starting area

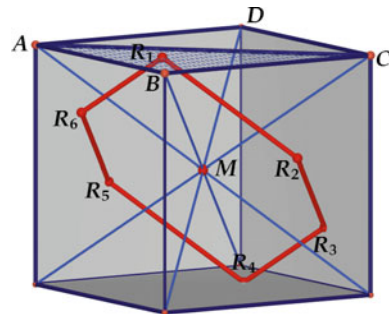
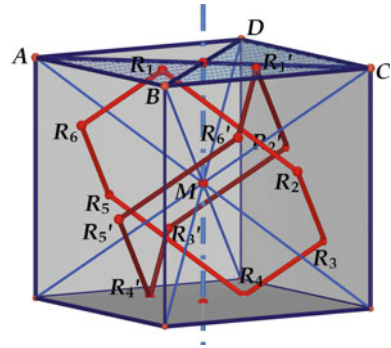


Fig. 7 Rotation of a type 1 billiards hexagon into a type 2 billiards hexagon



of a hexagon with a starting point inside the triangle ABC and the generating space diagonals through A, B, and C.

2.4 Special Shapes of the Billiards Hexagon

Special shapes of the type 1 billiards hexagon occur if R_1 is chosen so that it lies within one of the medians of the triangle ABC (Fig. 8, R_1 lies within BM_b). If this is the case, the hexagon will have two diametrical pairs of adjunct sides, all of which have the same length. The same holds for type 2 billiards hexagons in relation to the triangle BCD.

If R_1 lies on the centre of a square side, which is also the centre of mass of this side, the billiards rectangle is a square (Fig. 9). If R_1 coincides with the centre of mass of the triangle ABC, the spatial billiards hexagon is even equilateral (Fig. 10). The same is true for type 2 billiards hexagons with regard to the triangle BCD.

Comment 2: Gardner (1971) does not comment on the aforementioned properties of a billiards hexagon for special positions of starting point R_1 .

Fig. 8 Special shape of the billiards hexagon

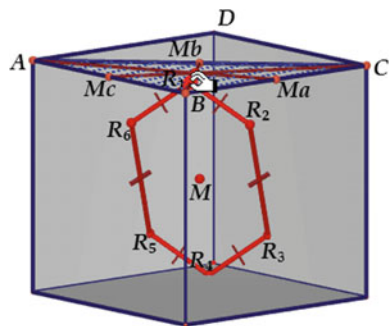


Fig. 9 Equilateral billiards rectangle (square)

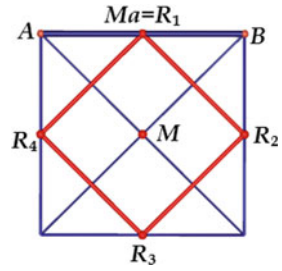
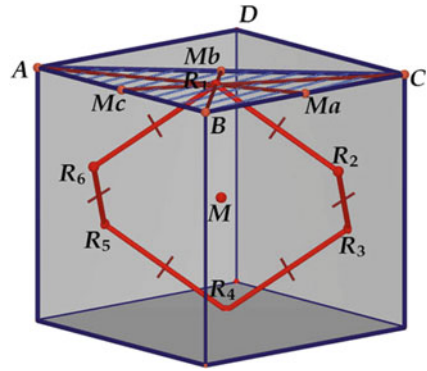


Fig. 10 Equilateral billiards hexagon, thus regular and point symmetrical



3 A First Generalization: Billiards in Equilateral Parallelepipeds

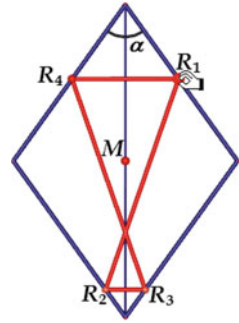
The cube can be generalized so that either its right-angled vertices or the equality of its edges or even the congruence of its six faces remain unchanged. Such generalizations of the cube are the cuboid (right-angled parallelepiped) and the equal-edged parallelepiped whose faces are rhombi. There is a special case for the latter in which the figure has congruent rhombi as faces, and so is an equilateral parallelepiped. I will limit my observations to billiards in equilateral parallelepipeds as their shape is determined by only one parameter: the size of the rhombus angle (α).

Comment 3: The generalizations of billiards to the cuboid and the parallelepiped leads to the problem of abstraction from the variety of shape parameters. Thus, findings about the billiards trajectories can be gained even more than before through inductive means. At the same time, this illustrates the limits of the inductive method.

3.1 Billiards Trajectories in Equilateral Parallelepipeds

The billiards trajectory in a rhombus is an intersecting quadrangle symmetrical to one of the rhombus's diagonals (Fig. 11). In an equilateral parallelepiped with acute

Fig. 11 Billiards trajectory in a rhombus



rhombus angle α , this trajectory corresponds to a hexagonal billiards trajectory with the following properties (Fig. 12): The angles with the vertices R_2, R_3, R_4 and R_5 are equal, as are the angles with the vertices R_1 and R_6 . The sides R_1R_2 and R_4R_5 are equal in length. For an equilateral parallelepiped and a reflection analogous to the one in the rhombus (Fig. 13), the special case of an intersecting, point-symmetrical billiards hexagon occurs (Fig. 14). In this hexagon, even the sides R_1R_6, R_6R_5, R_2R_3 and R_3R_4 are equal in length; however, its perimeter is not minimal. For the rhombus angle $\alpha = 90^\circ$, there are no billiards trajectories with reflection on opposite faces.

Fig. 12 Billiards trajectory in an equilateral parallelepiped

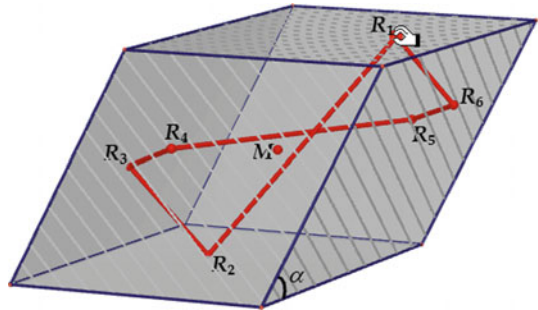


Fig. 13 Point symmetrical billiards-quadrangle in a rhomb

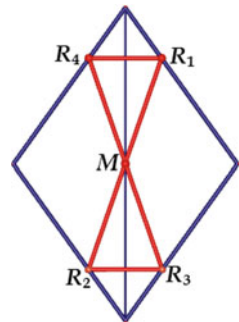
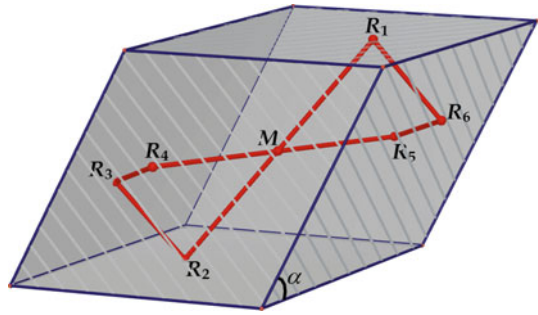


Fig. 14 Point symmetrical billiards hexagon in an equilateral parallelepiped



In contrast to the rhombus, there are, as in the cube, also billiards traces with a reflection on consecutive faces of the equilateral parallelepiped. There are also hexagonal billiards trajectories that are symmetrical to one of the diagonal planes of the parallelepiped. In this case, R_1 lies on a corresponding diagonal plane (Fig. 15). Whereas in a non-square rhombus, all billiards quadrangles are equiangular but no equilateral billiards quadrangle exists, it is natural to ask about the existence of an equiangular and equilateral billiards hexagon in an equilateral parallelepiped. As a special case of the plane-symmetrical billiards trajectory, there is a billiards hexagon symmetrical to the centre M of the equilateral parallelepiped, which is equiangular and equilateral (Fig. 16). Among all billiards hexagons within the equilateral parallelepiped this is the one with the smallest perimeter.

Comment 4: The convex hull of the regular point-symmetrical billiards hexagon in the equilateral parallelepiped is, as in the case of the cube, a triangular antiprism with congruent equilateral triangles as base areas.

Fig. 15 Plane-symmetrical billiards hexagon in an equilateral parallelepiped with reflection on adjunct faces

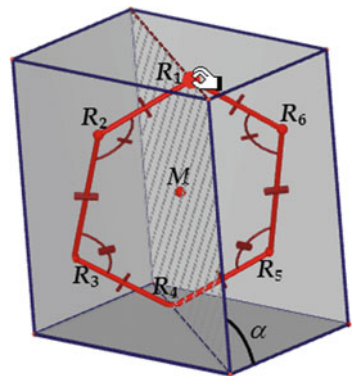
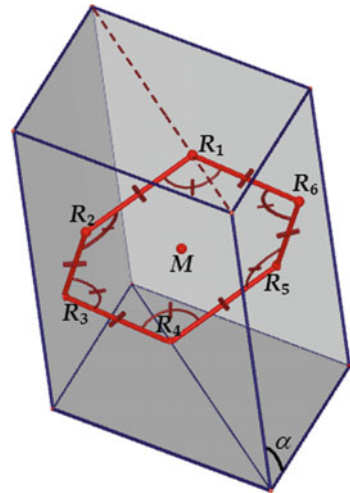


Fig. 16 Regular point-symmetrical billiards hexagon in an equilateral parallelepiped with reflection on adjunct faces

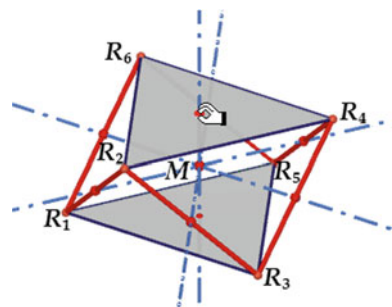


4 From the Spatial Regular Hexagon Towards the Equilateral Parallelepiped for Which This Hexagon Constitutes an Orbiting Billiards Trajectory

So far, I have constructed the respective billiards hexagons for the cube and a special generalization of the cube. In the following, I pursue the heuristic method of reversibility and approach the issue from the other side by using the hexagon to construct the hexahedron for which the hexagon is a billiards trajectory. In this way, types of hexahedra may be discovered which depend on the shape of the spatial hexagon.

Construction: Regular spatial hexagons $R_1R_2R_3R_4R_5R_6$ result, for example, from triangular antiprisms with an equilateral triangle as base area and another congruent triangle as the other base, but vertically shifted and rotated by 180° (a so-called antiprismatic hexagon or Petrie polygon of the antiprism—see Fig. 17). The

Fig. 17 Triangular antiprism with a regular point-symmetrical hexagon (Petrie polygon), symmetry axes, and symmetry centre



antiprism’s symmetrical properties transfer to the hexagon. The angle bisectors for such a hexagon are constructed (Fig. 18).

Perpendicular planes are then constructed in the vertices R_1, R_2, R_3, R_4, R_5 and R_6 of the angle bisectors (with perpendicular planes for R_1, R_3 and R_5 and their intersection point A —see Fig. 19). All three non-parallel perpendicular planes through the vertex intersect in a three-edged vertex of a polyhedron. Due to the properties of the regular point-symmetrical 3D hexagon, this polyhedron is an equilateral parallelepiped $ABCDEFGH$ (Fig. 20). First, the two-fold rotation axes of the antiprismic hexagon are projected onto the two-fold axes connecting the centres of opposing edges. Second, the three-fold symmetry axis of the hexagon is projected onto one of the three-fold symmetry axes of the parallelepiped. And third, the symmetry centre is projected onto the symmetry centre of the parallelepiped. Unfolding the parallelepiped to a net illustrates and confirms its equilateral property (Fig. 21). Thus, we arrive at the following assertion:

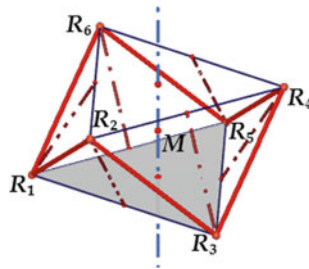


Fig. 18 Regular point-symmetrical hexagon with angle bisectors

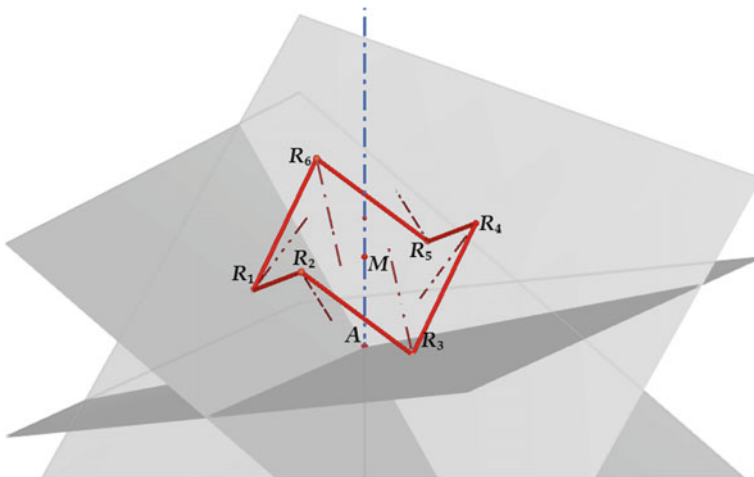


Fig. 19 Intersection of three perpendicular planes of three angle bisectors in a regular point-symmetrical hexagon

Fig. 20 Equilateral parallelepiped with generating regular point-symmetrical hexagon

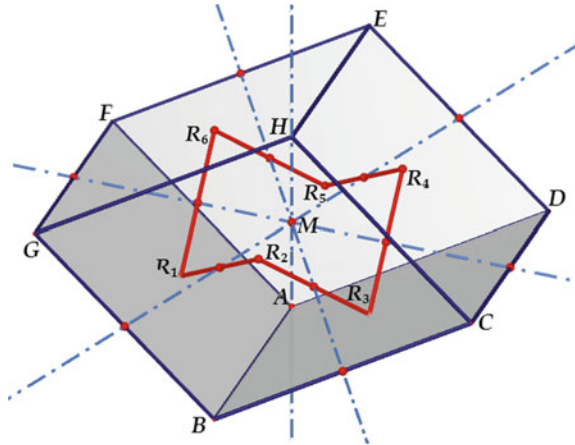
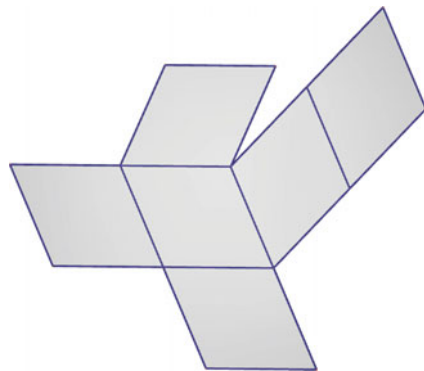


Fig. 21 A net of the equilateral parallelepiped



For a regular point-symmetrical spatial hexagon $R_1 R_2 R_3 R_4 R_5 R_6$ there is exactly one equilateral parallelepiped ABCDEFGH, for which the given hexagon $R_1 R_2 R_3 R_4 R_5 R_6$ constitutes an orbiting billiards trajectory.

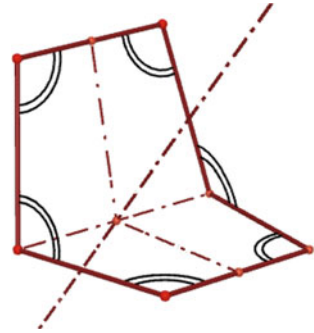
Comment 5: Besides the antiprismatic regular hexagons, there are other closed spatial hexagons that are regular (Fig. 22—example of a regular 3D hexagon that is axially symmetrical, but not point symmetrical).

Comment 6: If the angle of the regular hexagon is $2 \arccos \frac{2\sqrt{3}}{6}$, the equilateral parallelepiped is a cube (this angle equals the obtuse angle between two spatial diagonals of a cube).

Furthermore, by applying the result of the previous section (Fig. 16), we discover that the equilateral parallelepiped has a regular, point-symmetrical hexagon as its orbiting billiards trajectory. A property of the equilateral parallelepiped:

A hexahedron is an equilateral parallelepiped if and only if it has a single-orbit billiards trajectory which is a point-symmetrical 3D hexagon.

Fig. 22 Regular 3D hexagon, which has only one two-fold symmetry axis



Or:

A hexahedron is an equilateral parallelepiped if and only if it has an inscribed regular point-symmetrical 3D hexagon with minimal perimeter.

Comment 7: This can be followed by a study of simply closed regular point-symmetrical billiards trajectories with 8, 10, 12, ... vertices which lead to trapezohedrons consisting of a corresponding number of congruent kites. Symmetrical billiards quadrangles result in special tetrahedrons, amongst which the equilateral tetrahedron can be found (Schumann 2017).

Insertion: Besides the (simply closed) regular point-symmetrical hexagons, there are also, for example, self-intersecting regular hexagons with a three-fold symmetry axis that can be constructed easily from three-sided prisms with regular base areas (Fig. 23).

For regular hexagons of the self-intersecting type, there is a convex hull of the planes perpendicular to the angle bisectors in the vertices which forms a three-sided double pyramid, each consisting of congruent isosceles triangles (Fig. 24). The three-sided double pyramid from equilateral triangles is a special case.

This leads to the following statement:

Fig. 23 Regular symmetric self-intersecting 3D hexagon

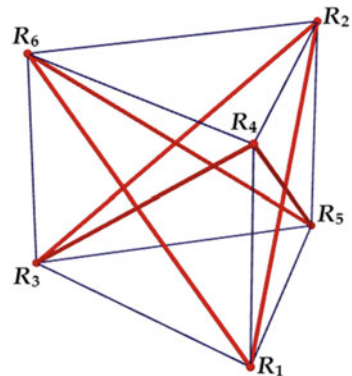
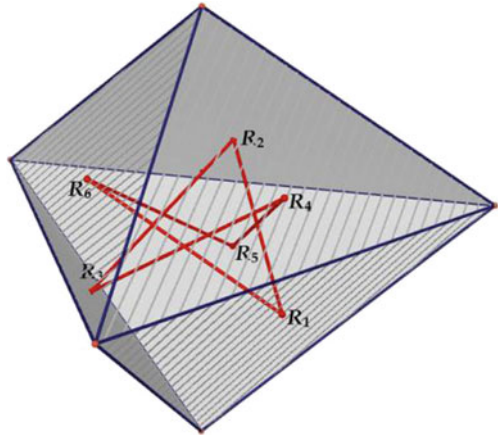


Fig. 24 Three-sided double pyramid with generating self-intersecting regular point-symmetrical 3D hexagon



A hexahedron is a three-sided double pyramid consisting of congruent isosceles triangles if and only if there is a simple orbiting billiards trajectory in the shape of a self-intersecting regular hexagon with a three-fold symmetry axis.

Or:

A hexahedron is a three-sided double pyramid consisting of congruent isosceles triangles if and only if there is an inscribed self-intersecting regular hexagon with three-fold symmetry axis whose perimeter is minimal.

Comment 8: An examination of such self-intersecting regular billiards trajectories with 10, 14, 18, ... vertices leads to trapezoids with corresponding numbers of congruent kites. Such polyhedra are also called deltoids.

Due to lack of space, I close my examination with identification of the parallelepiped.

Comment 9: It is questionable whether overtaking the definition of regular planar polygons by equilaterality and equiangularity for spatial polygons is sensible, as this definition also includes spatial polygons which do not “look” regular, as they are not vertex equivalent like the regular planar polygons. It is appropriate to require vertex transitivity for equilateral spatial polygons in order for them to be defined as regular; this includes equiangularity. According to this definition, the only remaining regular spatial polygons with more than five vertices are the $2n$ -sided polygons of the corresponding n -gonal antiprisms for $n = 3, 4, 5, \dots$ and the $2(2n+1)$ -sided polygons of the face self-intersecting diagonals of the corresponding $(2n+1)$ -sided prisms for $n = 1, 2, 3, \dots$. This was proven by Efremovitch and Il’jashenko (1962).

4.1 Identification of the parallelepiped

There are billiards trajectories within the parallelepiped that are (simply closed) point-symmetrical hexagons. Vice versa, a construction of the convex hull formed

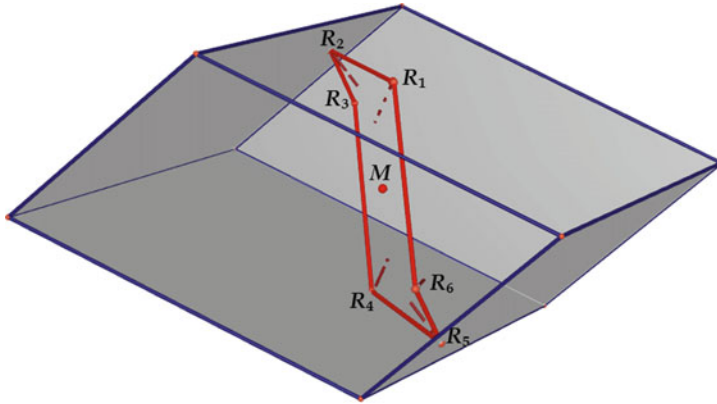


Fig. 25 Parallelepiped with generating point-symmetrical 3D hexagon

by the perpendicular planes to the angle bisectors in the vertices of such hexagons creates a parallelepiped (Fig. 25). This leads to the following statement:

A hexahedron is a parallelepiped if and only if it has a point-symmetrical billiards hexagon.

Or:

A hexahedron is a parallelepiped if and only if it has an inscribed point-symmetrical hexagon with minimal perimeter.

Comment 10: As far as I know, the theorems heuristically elaborated in this section have not yet been published.

5 An Identification of Special Tetrahedra

If symmetrical 3D quadrangles are those quadrangles emerging from unfolding the corresponding symmetrical plane quadrangles along one of their diagonals (Schumann, 2017), the following identification of special tetrahedrons along inscribed symmetrical 3D quadrangles results (Fig. 26):

- A tetrahedron consists of four congruent isosceles triangular faces if and only if its billiards quadrangle is an equiangular 3D rhombus. This means that the tetrahedron is isosceles and equilateral.
- A tetrahedron consists of four congruent triangular faces if and only if its billiards quadrangle is an equiangular 3D parallelogram. This means the tetrahedron is equilateral.
- A tetrahedron consists of two pairs of triangular faces, each of which consists of two congruent isosceles triangles, if and only if its billiards quadrangle is a 3D rhombus.

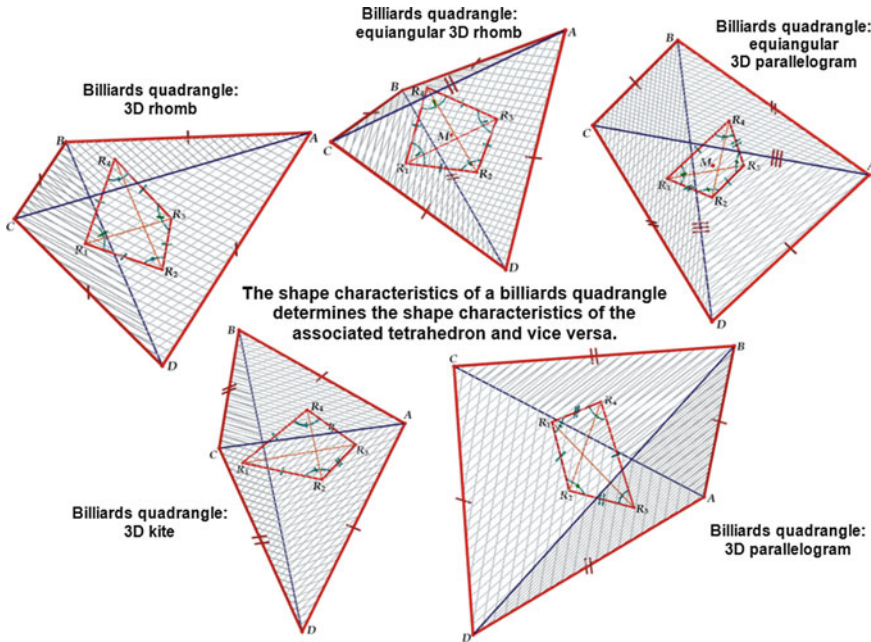


Fig. 26 Symmetrical 3D quadrangles as billiards quadrangles

- A tetrahedron consists of two pairs of triangular faces, each of which consists of two congruent triangles, if and only if its billiards quadrangle is a 3D parallelogram.
- A tetrahedron consists of two pairs of triangular faces, one of which consists of two congruent isosceles triangles and the other of congruent triangles, if and only if its billiards quadrangle is a 3D kite.

Comment 11: Not all 3D kites will reveal a tetrahedron whose simply closed billiards trajectory is this kite. This sets the kite apart from the other symmetrical 3D quadrangles.

6 Concluding Remarks

The investigations presented here can be transferred to omnihedral orbiting billiards trajectories in other convex polyhedra, such as the other platonic solids (Hudelsohn, n.d.), Archimedean solids, Johnson solids (convex polyhedra with regular polygon faces), or Catalan solids (Archimedean dual solids). This relation between convex polyhedra and special 3D polygons can be denoted as a type of “orthogonal-duality.” The more faces these polyhedra have, the more difficult the interactive construction of corresponding billiards polygons becomes. Indeed, such investigations are not always successful. If the investigation is based on special 3D polygons, different convex

polyhedra can be found as convex hulls of the perpendicular planes to the angle bisectors in the vertices of the polygon, which can be identified by their polygonal properties.

Several open questions remain regarding the existence of polyhedra for given billiards polygons. When regarding 3D polygons as polygonal billiards trajectories, the lack of a thorough theoretical investigation concerning such polygons becomes obvious. However, with the aid of dynamic 3D-software as described and used in this chapter, such investigations are now a feasible reality.

Lastly, with available software, students at both high school and undergraduate level, can easily conduct such investigations. With the confidence provided by such empirical investigations, students can be encouraged to explain (prove) the results in a variety of ways ranging from the use of symmetry, synthetic geometry, and vectors. Such investigations can be valuable extensions of plane geometry to 3D geometry.

References

- Bainville, E., & Laborde, J.-M. (2004–2015) Cabri 3D 2.1. Grenoble: Cabrilog (www.cabri.com).
- Berger, M. (2010). *Geometry revealed. A Jacob's ladder to modern higher geometry* (p. 728). Heidelberg: Springer.
- Borwein, J. (2012). Exploratory experimentation: digitally-assisted discovery and proof. In G. Hanna & M. de Villiers (Eds.), *Proof and proving in mathematics education*. New ICMI Study Series 15 (pp. 69–95). Springer. https://doi.org/10.1007/978-94-007-2129-6_4.
- De Villiers, M. (1990). The role and function of proof. *Pythagoras*, 24, 17–24.
- De Villiers, M. (2010). Experimentation and proof in mathematics. In G. Hanna, H. Jahnke & H. Pulte (Eds.), *Explanation and proof in mathematics: Philosophical and educational perspectives* (pp. 205–221). Springer. https://doi.org/10.1007/978-1-4419-0576-5_14.
- Efremovitch, V. A., & Il'jashenko, J. S. (1962). *Regular Polygons in En (Russian)*. Bulletin Moscow University, Series I 17, No. 5, pp. 18–23.
- Galperin, G. A., & Semliakov, A. N. (1990). *Mathematical billiards (Russian)*. Moskau: Nauka.
- Gardner, M. (1971). *Martin Gardener's sixth book of mathematical games from scientific American* (pp. 29–38). San Francisco: W. H. Freeman.
- Gutierrez, A., Pegg, J., & Lawrie, C. (2004). Characterization of students' reasoning and proof abilities in 3-dimensional geometry. *Proceedings of the 28th Conference of the International Group for the Psychology of Mathematics Education* (Vol. 2, pp. 511–518).
- Hanna, G. (2000). Proof, explanation and exploration: An overview. *Educational Studies in Mathematics*, 44, 5–23.
- Harel, G. (2013). Intellectual need. In K. Leatham (Ed.), *Vital directions for mathematics education research* (pp. 119–151). New York (NY): Springer.
- Hudelson, M. (not specifiable). Periodic omnihedral billiards in regular polyhedra and polytopes.
- Schumann, H. (2007). Schulgeometrie im virtuellen Handlungsraum (School geometry in the virtual action space). Hildesheim: Verlag Franzbecker.
- Schumann, H. (2017). Das räumliche Viereck – eine Einführung (The spatial quadrangle—An introduction). *MNU Journal* 6(70), 382–389.
- Tabachnikov, S. (2013). *Geometrie und Billard (Geometry and Billiards)*. Berlin, Heidelberg: Springer.
- Winter, H. (1989). *Entdeckendes Lernen im Mathematikunterricht (Discovery learning in the teaching of mathematics)*. Braunschweig: Verlag Vieweg.

Links

<http://stanwagon.com/public/hudelsonbilliards.pdf>. Accessed 13 Aug 2019.

<http://mathworld.wolfram.com/Billiards.html>. Accessed 13 Aug 2019.

Classroom Experience with Proof Software

Learning Logic and Proof with an Interactive Theorem Prover



Jeremy Avigad

1 Introduction

Although mathematical language and the canons of proof have been evolving for centuries, the fundamentals have remained remarkably stable over time. High school students can still profitably read Euclid's *Elements* and follow the deductive structure of the arguments. They can similarly make sense of Cardano's solution to the cubic equation and read Euler's proof of the theorem that if p is a prime number and a is not divisible by p , then a^{p-1} is congruent to 1 modulo p .

In contemporary terms, the definition of a prime number might be rendered as follows:

Definition 1.1 A natural number p is prime if and only if $p \geq 2$ and for every n , if $n \mid p$, then $n = 1$ or $n = p$.

Symbolic logic provides idealized languages to model informal mathematical vernacular, and in a language in which variables are assumed to range over the natural numbers, the definition of a prime number might be rendered as follows:

$$\text{Prime}(p) \equiv p \geq 2 \wedge \forall n (n \mid p \rightarrow n = 1 \vee n = p).$$

Logicians like to think that symbolic logic helps students understand mathematical language and proof, since it is specifically designed to codify the rules and norms that govern mathematical reasoning. This chapter provides at least anecdotal evidence that this is the case, describing a course that teaches symbolic logic and ordinary mathematical proof together. The course asks students to recognize that they

J. Avigad (✉)

Department of Philosophy, Carnegie Mellon University, Baker Hall 161, Pittsburgh,
PA 15213, USA

e-mail: avigad@cmu.edu

© Springer Nature Switzerland AG 2019

G. Hanna et al. (eds.), *Proof Technology in Mathematics Research and Teaching*, Mathematics Education in the Digital Era 14,
https://doi.org/10.1007/978-3-030-28483-1_13

277

are dealing with two different languages and keep them distinct, donning a mathematician’s hat and a logician’s hat at different times. This provides complementary perspectives on mathematical reasoning and encourages students to think about the relationships between the two.

A novel feature of the course is that students also learn to use a third language, namely, the language of the *Lean* interactive theorem prover (de Moura, Kong, Avigad, van Doorn, & von Raumer, 2015). In this system, the definition of a prime number can be rendered as follows:

```
def prime (p : ℕ) := p ≥ 2 ∧ ∀ n, n | p → n = 1 ∨ n = p
```

Such definitions are parsed and checked for grammatical correctness by the system. Students are asked to write proofs that can be checked by the system as well, in a proof language that looks much like a programming language. This requires them to don yet a third hat, namely, that of the computer scientist, and put up with the trials and tribulations of formal specification.

The goal of the course is to teach students to read, write, and understand ordinary mathematical proofs. There are independent reasons to teach students symbolic logic, which is an interesting branch of mathematics in its own right, with applications to philosophy, computer science, linguistics, and other disciplines. There are also independent reasons to teach students how to use an interactive theorem prover: not only are such systems becoming important tools for the verification of complex hardware and software systems, but, more generally, facility with logical formalisms and specification languages is important in a number of branches of computer science, ranging from programming languages and database theory to artificial intelligence. But these are not explicitly addressed in the course, and we take these benefits to be ancillary. The course remains focused on mathematics.

Robert Y. Lewis, Floris van Doorn, and I have taught the course to undergraduates at Carnegie Mellon University over the past three years, based on course materials that we are still developing (Avigad, Lewis, & van Doorn, 2019). These are freely available online:

http://leanprover.github.io/logic_and_proof

The textbook can be read in a browser, and clicking on examples and exercises in Lean launches a version of the program that also runs in the browser. A PDF version of the textbook is also available for download, and Lean can be installed and run independently. Running Lean in the browser is slower than running a native version, but it is more than adequate for the purposes of the course.

Here I report on our experiences teaching the course. When I say “we tell students ...” or “we discuss ...,” I am describing our general classroom practice. But since we developed the textbook specifically to support the lectures, statements like these generally describe the contents of the textbook as well.

2 Course Outline

The course was taught with three weekly 50-minute meetings over the course of a 14-week semester. The textbook’s introductory chapter proclaims to students that the goals are as follows:

- to teach you to write clear, “literate,” mathematical proofs
- to introduce you to symbolic logic and the formal modeling of deductive proof
- to introduce you to interactive theorem proving
- to teach you to understand how to use logic as a precise language for making claims about systems of objects and the relationships between them, and specifying certain states of affairs.

We tell students that they will learn three separate languages, and that they should take care not to conflate them. We also tell them that the course can be viewed simultaneously as an introduction to mathematical proof, an introduction to symbolic logic, and an introduction to interactive theorem proving. We emphasize, however, that the latter two components are carried out in service of the first.

The first third of the course focuses on symbolic logic. It begins with a logic puzzle by George Sumner, “Malice and Alice,” that presents a list of assertions about a cast of characters consisting of Alice, her husband, her brother, her son, and her daughter, and the murder of one by another. Students are asked to work out a solution to the problem on their own, but then we turn to an analysis of the language and form of Sumner’s published solution. We ask students to reflect on the role of words and phrases like “and,” “or,” “if ...then,” and “this is impossible,” the last of which signals a contradiction. We then introduce symbols corresponding to these verbal constructions, and begin to analyze the rules governing their use.

Our introduction to symbolic logic is fairly conventional. We first present propositional logic and introduce natural deduction as our system of formal proof. We describe truth-table semantics, and explain the notions of soundness and completeness informally. We then continue with the language of first-order logic, including function and relation symbols, quantifiers, and equality. Once again, we present the relevant proof rules in natural deduction, followed by an informal discussion of the semantics. We get students to recognize that a first-order sentence can be interpreted in various structures, and that the role of the proof system is to certify certain entailments as *valid*, which means that the conclusion holds in any structure that satisfies the hypotheses.

The main novelty is that the conventional presentation is accompanied by an introduction to the use of logic in Lean. As we go, we show students how to write formal expressions and proofs that can be checked automatically. This material is treated in separate chapters in the textbook, with a clearly delineated shift in the exposition. Roughly every third class, we set aside conventional lectures to project a laptop display to the front of the classroom and work through examples and proofs together. Weekly homework assignments reflect a similar balance: roughly one third of the exercises are carried out formally in Lean, with the remainder consisting of conventional pen-and-paper exercises.

The remaining two thirds of the course provide a similarly conventional introduction to the fundamental concepts of mathematics and mathematical proof. We present elementary set theory, the conventional operations on sets, and the usual set identities. We talk about relations, including order relations and equivalence relations. We deal with functions and their general properties. We then turn to the natural numbers and proof by induction. With these fundamentals in place, we illustrate their use with quick tours of key mathematical topics: elementary number theory, finite combinatorics and counting principles, a construction of the real numbers via Cauchy sequences, and the theory of the infinite. A final chapter brings the logical foundations and the informal mathematics together by presenting and exploring the axioms of set theory.

In this latter part of the course, symbolic logic plays a more limited role. Initially, we show students how informal proofs of set identities can be carried out in natural deduction, but while natural deduction is useful for modeling the building blocks of proof, once those building blocks have been established, the utility of the system diminishes quickly. Natural deduction does not scale to writing real mathematical arguments, and laboriously representing long mathematical arguments in those terms would produce no new insights. This is not to say that logic has no role in clarifying concepts: later in the course, we return to logic to cast the principle of induction in symbolic terms and observe that it quantifies over predicates. Similarly, we ask students to recognize that the statement “a function is surjective if and only if it has a right inverse” involves an implicit quantification over functions. The set-theoretic axioms, at the very end of the course, are presented explicitly in the first-order language of set theory. This return to symbolic logic brings the presentation full circle.

When it comes to modeling ordinary mathematical proof in formal terms, Lean allows us to get further than natural deduction. The chapters on sets, relation, functions, and the natural numbers are also tracked by chapters that show how to reason about them in Lean. Lean’s logic is very expressive, so that, in principle, *any* ordinary mathematical theorem can be formalized and proved in the system. But the course is not a course in interactive theorem proving, and teaching students to use the range of Lean’s features and libraries falls well outside the scope. Rather, we focus on small examples: set theoretic identities, elementary properties of relations and functions, and simple proofs by induction. As with logical reasoning, the point is primarily to solidify students’ understanding of the patterns of reasoning by showing them the formal analogues. Thus we provide students with just enough information to carry out basic, illustrative homework examples, and ask them to carry out more complicated pen-and-paper proofs in ordinary mathematical language.

Over time, we expect Lean to evolve to the point where we will be able to use automation to support writing more complicated mathematical proofs in a style that is close to the informal ones we expect students to write. As a result, we ultimately hope to extend the textbook with Lean-based chapters on combinatorics, number theory, the real numbers, and so on. In the meanwhile, the approach we have adopted is decidedly pragmatic: we use Lean as long as it is a useful tool for helping students to understand the logical structure of mathematical concepts and proofs, and otherwise rely on conventional mathematical presentations.

3 Formal Perspectives on Proof

In this section, I will discuss the ways that using an interactive theorem prover can help students understand the logical structure of mathematical statements and proofs. Representing natural language assertions in symbolic terms takes some getting used to. We explain to students that, in the language of propositional logic, we might assign propositional variables to basic assertions as follows:

- *A*: Alice’s husband was in the bar
- *B*: Alice was on the beach
- *C*: Alice was in the bar
- *D*: Alice’s husband was on the beach.

With that assignment, the statement “either Alice’s husband was in the bar and Alice was on the beach, or Alice was in the bar and Alice’s husband was on the beach” is rendered as

$$(A \wedge B) \vee (C \wedge D).$$

In Lean, we can declare propositional variables and write the expression as follows:

```
variables A B C D : Prop
#check (A ∧ B) ∨ (C ∧ D)
```

The `#check` command provides immediate feedback to the student and confirms that the expression is a well formed proposition. Lean is designed as a research tool and not for pedagogical purposes, and error messages can be cryptic. But they are generally good enough for students to locate problems and fix them, with some trial and error. For that purpose, Lean’s real-time feedback is essential.

For another example, after we introduce relations and quantifiers, students learn that they can express the fact that a binary relation $<$ is dense by writing

$$\forall x, y (x < y \rightarrow \exists z (x < z \wedge z < y)).$$

In Lean, they can declare a binary relation on a datatype, A , assign the $<$ notation, and express the assertion as follows:

```
variables (A : Type) (R : A → A → Prop)
local infix ` < ` := R
#check ∀ x y, x < y → ∃ z, x < z ∧ z < y
```

The correspondence between the usual notation of first-order logic and Lean’s syntax is not exact. For example, in first-order logic one usually fixes the underlying domain of all the variables and functions, whereas in Lean one declares the underlying data type A explicitly and models the relation R as a function which takes two elements of A and returns a proposition. (In computer science and interactive theorem proving, it is common to take the binary arrow operation in $R : A \rightarrow A \rightarrow \text{Prop}$ to associate to the right, to make declarations like this convenient.) Another difference

is that in mathematical logic it is common to take quantifiers to bind tightly, which means that the parentheses in the first rendering above are needed to make the scope of the universal quantifier over x and y bind the rest of the formula. In contrast, in computer science, it is common to give quantifiers the widest scope possible, which explains why no parentheses are needed in Lean’s formulation. We do not shy away from explaining this to students: learning to use a symbolic formal language requires learning the relevant conventions. To avoid overwhelming them, we do our best to limit the information they need to understand the examples and carry out the homework assignments. To our pleasant surprise, we have found that minor syntactic differences between the three languages—informal mathematical language, symbolic logic, and Lean—do not cause problems. Students are capable of moving between the different linguistic contexts while at the same time recognizing them as alternate representations of a common logical structure.

In the course, we use both natural deduction and Lean to convey to students the common patterns and idioms of mathematical proof. For example, informally, we tell students that in order to prove a statement of the form “if A then B ,” they should assume A hypothetically and use that assumption to argue that B holds. As a natural deduction proof rule, this pattern is represented as follows:

$$\frac{\begin{array}{c} \overline{A} \\ \vdots \\ B \end{array}}{A \rightarrow B}$$

The line over the hypothesis A indicates that the hypothesis is *canceled* when the inference is complete. The vertical ellipsis represents the argument for B . In Lean, the pattern is rendered as follows:

```
variables A B : Prop

example : A → B :=
  assume h : A,
  show B, from ...
```

Once again, the ellipsis represents the argument for B using A . Notice that the hypothesis A is labeled with the letter h . We will see below how such labels can be used in a proof.

In a similar way, we teach students the standard patterns for dealing with conjunction, disjunction, negation, and universal and existential quantifiers. In each case, we use informal examples to motivate the corresponding rules in natural deduction, and later show students how to implement the patterns in Lean. The rule for eliminating a disjunction, corresponding to a proof by cases in ordinary mathematics, is a nice example. In natural deduction, the rule is represented as follows:

$$\frac{\begin{array}{ccc} \overline{A} & & \overline{B} \\ & \vdots & \vdots \\ A \vee B & C & C \end{array}}{C}$$

The rule is used to establish that a statement C follows from the assumption “ A or B ,” denoted $A \vee B$ in symbolic logic. The rule has three premises: assuming, first, that we know that A or B holds, second, that we can prove that C follows from A , and, third, that we can prove that C follows from B , we can conclude that C holds outright. Once again the lines over the local hypotheses A and B signify that these are only temporary assumptions, and that the resulting proof of C does not depend on them. In Lean, proof by cases is carried out as follows:

```
example : C :=
have h : A ∨ B, from ...,
or.elim h
  (assume h1 : A,
   show C, from ...)
  (assume h1 : B,
   show C, from ...)
```

This example is only schematic; in actual use, A , B , and C would likely be compound formulas, and ambient assumptions or previously established facts would allow us to fill in the ellipses with actual proofs. Notice that h labels the fact that $A \vee B$ is available in the current context, and `or.elim h` tells Lean what we wish to use the or-elimination rule described in symbolic terms above. The terminology matches the conventions used to name rules in natural deduction, solidifying the relationship in students’ minds.

The following natural deduction proof establishes the implication $A \wedge (B \vee C) \rightarrow (A \wedge B) \vee (A \wedge C)$:

$$\frac{\frac{\frac{A \wedge (B \vee C)}{B \vee C}^1}{\frac{\frac{\frac{A \wedge (B \vee C)}{A}^1}{A \wedge B} \quad \frac{B}{B}^2}{(A \wedge B) \vee (A \wedge C)} \quad \frac{\frac{\frac{A \wedge (B \vee C)}{A \wedge C}^1}{A \wedge C} \quad \frac{C}{C}^2}{(A \wedge B) \vee (A \wedge C)}^2}{\frac{A \wedge (B \vee C) \rightarrow (A \wedge B) \vee (A \wedge C)}{A \wedge (B \vee C) \rightarrow (A \wedge B) \vee (A \wedge C)}^1}^1$$

Students were expected to be able to carry out proofs like this for homework or during an exam. The labels on the hypotheses are used to tag the places in the proof where those hypotheses are canceled. Here is a corresponding proof in Lean:

```
variables A B C : Prop

example : A ∧ (B ∨ C) → (A ∧ B) ∨ (A ∧ C) :=
assume h : A ∧ (B ∨ C),
have h1 : A, from and.left h,
have h2 : B ∨ C, from and.right h,
or.elim h2
  (assume h3 : B,
   have h4 : A ∧ B, from and.intro h1 h3,
   show (A ∧ B) ∨ (A ∧ C), from or.inl h4)
  (assume h3 : C,
   have h4 : A ∧ C, from and.intro h1 h3,
   show (A ∧ B) ∨ (A ∧ C), from or.inr h4)
```


Students were also expected to be able to carry out proofs like this on their homework. (We gave ordinary pen-and-paper exams, so their ability to use Lean was only assessed on homework assignments.) The lines that begin with `have` support forward reasoning, establishing facts that can be used later on. The keyword `show` is usually unnecessary, but allows the user to display the conclusion of the proof that follows, making the proof more readable and robust.

As with assertions, the correspondence between natural deduction proofs and proofs in Lean is not exact, and we have to teach students how to translate their intuitions from one to the other. Thus, once again, the two representations provide alternative perspectives on informal mathematical proof and reinforce one another. The syntax is precise and unforgiving, and students often struggle with it. But most come to enjoy it in the end. Using a proof assistant offers instant gratification: when a proof is correct, students know it right away, and feel good about it.

4 Formal Perspectives on Mathematics

It may not seem surprising that an interactive theorem prover can be useful for teaching the syntax and rules of logic. But after a few weeks in our course, logic is relegated to the background, and we turn to the fundamental building blocks of mathematics. We found that Lean continues to be a helpful tool in that respect as well.

We begin with a conventional treatment of sets, covering not just binary operations like union and intersection, but indexed unions and intersections as well. When we initially designed the course, an aspect of Lean’s underlying foundational framework gave us pause. In ordinary mathematical discourse, it is permissible to say “let A , B , and C be any three sets,” without specifying what sort of elements they contain. In practice, mathematicians generally deal with sets of objects of some domain—for example, sets of numbers, sets of points, or sets of elements of some group. In Lean, every object is assumed to live in some such domain, its *type*, and it is most natural to introduce the concept of a set of elements of a particular type. In other words, in a Lean formulation, one can say “let A , B , and C be sets of natural numbers,” or, more abstractly, “let U be some type, and let A , B , and C be sets of elements of U .”

On the informal side, we were committed to following conventional mathematical terminology, making the treatment in our text seem as run-of-the-mill as possible. When we first designed the course, we worried a good deal about how to smooth over the mismatches between our ordinary mathematical presentations and the formal representations in Lean. As it turns out, we needn’t have been concerned. It is easy to explain to students that whereas the broadest conception of set allows mathematicians to consider the three-element set consisting of the number seven, the function which maps any real number to its square, and the president of France, the representation in Lean is more restrictive. Students know that informal language is not the same as a formal language, and can appreciate the design decisions that are made when deciding how to represent informal proof in formal terms.

An ordinary mathematics textbook might offer the following example of a proof of a set theoretic identity:

Theorem 4.1 *Let A , B , and C be any three sets. Then $(A \setminus B) \setminus C = A \setminus (B \cup C)$.*

Proof. Suppose x is any element of $(A \setminus B) \setminus C$. Then x is in $A \setminus B$ but not C , and hence x is in A but not B . But this means that x is in A but not B or C , and hence not in $B \cup C$. So x is in $A \setminus (B \cup C)$, as required.

Conversely, suppose x is in $A \setminus (B \cup C)$, ...

A corresponding proof in Lean is not as concise. Below we break out the implicit inference used above that if x is not an element either B or C then it is not an element of $B \cup C$, and make it an independent lemma. Modulo that lemma, the formal proof follows the structure of the informal one closely.

```
import data.set
open set

variables {U : Type}

lemma ex41a {A B : set U} {x : U} (h1 : x ∉ A) (h2 : x ∉ B) :
  x ∉ A ∪ B :=
assume h3 : x ∈ A ∪ B,
or.elim h3
  (assume h4 : x ∈ A,
   show false, from h1 h4)
  (assume h4 : x ∈ B,
   show false, from h2 h4)

theorem ex41 (A B C : set U) : (A \ B) \ C = A \ (B ∪ C) :=
eq_of_subset_of_subset
  (assume x : U,
   assume h1 : x ∈ (A \ B) \ C,
   have h2 : x ∈ A \ B, from h1.left,
   have h3 : x ∉ C, from h1.right,
   have h4 : x ∈ A, from h2.left,
   have h5 : x ∉ B, from h2.right,
   have h6 : x ∉ B ∪ C, from ex41a h5 h3,
   show x ∈ A \ (B ∪ C), from ⟨h4, h6⟩)
  ...
```

In the current implementation of the system, students are required to `import` the relevant theory and `open` the `set` namespace. The latter allows users to write `eq_of_subset_of_subset` rather than `set.eq_of_subset_of_subset` to name the relevant lemma. We try to shield students from implementation details like these by providing them with templates on the homework assignments that provide such information. For example, we might carry out all the relevant imports, declare some variables, offers some examples of the background theorems we expect them to use, state a theorem, and then leave them the task of filling in the proof. This keeps the focus of the course on learning how to construct the proofs rather than learning how to use the theorem prover and its libraries.

In cases like this, automated proof procedures could be used to decrease the length of the proof. The level of detail above therefore represents an extreme form of mathematical rigor, in which every inference is spelled out explicitly. Though tedious, the practice helps students get used to the low-level mechanics of a proof. Over time, they learn how to moderate the level of detail when writing ordinary mathematics.

Formal languages are equally useful in clarifying the logical structure of common properties of relations and functions. For example, after introducing the notions of injectivity, surjectivity, and bijectivity, we ask students how they can be represented in terms of symbolic logic. By this point, students are generally comfortable making the translation, and so the corresponding definitions in Lean come as no surprise:

```
variables {X Y Z : Type}

def injective (f : X → Y) : Prop := ∀ {x₁ x₂}, f x₁ = f x₂ →
  x₁ = x₂

def surjective (f : X → Y) : Prop := ∀ y, ∃ x, f x = y

def bijective (f : X → Y) := injective f ∧ surjective f
```

In Lean, curly braces around arguments like x_1 and x_2 indicate that they are to be left implicit when writing expressions, because they can typically be inferred from the context of the expression and other arguments. With these definitions, it is not hard to prove that the composition of two injective functions is injective, and similarly for surjective functions:

```
theorem injective_comp {g : Y → Z} {f : X → Y}
  (Hg : injective g) (Hf : injective f) :
  injective (g ∘ f) :=
  assume x₁ x₂,
  assume : (g ∘ f) x₁ = (g ∘ f) x₂,
  have f x₁ = f x₂, from Hg this,
  show x₁ = x₂, from Hf this

theorem surjective_comp {g : Y → Z} {f : X → Y}
  (hg : surjective g) (hf : surjective f) :
  surjective (g ∘ f) :=
  assume z,
  exists.elim (hg z) $
  assume y (hy : g y = z),
  exists.elim (hf y) $
  assume x (hx : f x = y),
  have g (f x) = z, from eq.subst (eq.symm hx) hy,
  show ∃ x, g (f x) = z, from exists.intro x this
```

The use of the keyword `this` is another syntactic gadget in Lean: users can omit the label on a hypothesis, in which case `this` refers to the most recent hypothesis that has been left unlabeled. We introduce tricks like these casually, as side remarks. Some students enjoy using them, but they are by no means essential to carrying out the tasks that we assign for homework.

By this point, the gap between the kinds of facts we ask students to establish with pen-and-paper proofs and the kinds of facts we ask students to establish in Lean has grown. Full-blown interactive theorem proving can be a tedious and difficult affair, and it is a subject in and of itself. In our course we are committed instead to making sure that students are capable of writing the kinds of proofs that they would be expected to write in any course on the fundamentals of mathematical proof, and so we use only small examples in Lean that illuminate the main concepts and patterns of argument. For example, to illustrate proof by induction, we show students how to write a recursive foundational specification of addition on the natural numbers in terms of zero and the successor operation. We can then prove things like the commutativity of addition in Lean:

```
theorem add_comm (m n : ℕ) : m + n = n + m :=
nat.rec_on n
  (show m + 0 = 0 + m, by rewrite [add_zero, zero_add])
  (assume n,
    assume ih : m + n = n + m,
    show m + succ n = succ n + m, from calc
      m + succ n = succ (m + n) : by rewrite add_succ
        ... = succ (n + m) : by rewrite ih
          ... = succ n + m : by rewrite succ_add)
```

In contrast, when treating induction informally, we provide more interesting examples of combinatorial and number-theoretic identities, and discuss more general forms of induction. Ultimately, we would like to include some examples of these in Lean as well. But, at the moment, the material we have is more than sufficient to fill a one-semester course, and getting students used to the formal overhead would be too much of a distraction from our main goals.

5 Results

We have taught this course for three years now in the Dietrich College of Humanities and Social Sciences at Carnegie Mellon University, developing the materials along the way. The course is listed as a 200-level course, signaling that it is appropriate for first- and second-year undergraduate students but more advanced than courses at the 100 level. It fills a mathematical modeling breadth requirement for the college that can also be filled with a calculus or statistics course. The course draws students from a wide range of backgrounds, from majors in mathematics, computer science, or physics to majors in philosophy or even business administration. Students at Carnegie Mellon are generally strong and not averse to mathematics and computer science, but the course does not presuppose any background beyond ordinary high-school mathematics.

Students rated the course highly in their evaluations. Each year we also solicited informal feedback verbally and with short questionnaires, and in the second year we enlisted the aid of Carnegie Mellon's Eberly Center for Teaching Excellence to

carry out a more formal mid-semester evaluation. The classes were small and we have not accumulated precise data or developed quantitative measures of student improvement, so the data we report here is only anecdotal.

Students generally enjoyed the course, and some were quite enthusiastic. (“This was my favorite class to attend this semester—I always looked forward to it, and left happy.”) On questionnaires we identified the various components of the course—symbolic logic, ordinary mathematics, and Lean—and asked students to compare them in terms of difficulty and interest. Students generally felt that the material was appropriately balanced. Some expressed a slight preference for one component over another, but among these students the specific preferences were fairly evenly distributed. Most importantly to us, students found the combination to be natural and helpful, and did not find it confusing when we asked them to switch between the different perspectives on proof.

Lean’s syntax takes getting used to, and students were sometimes frustrated when the error messages were not sufficient for them to figure out what they were doing wrong. But all of them got the hang of it eventually, and many really enjoyed it. Using an interactive theorem prover offers instant gratification when a proof is accepted: Freek Wiedijk once described the feeling as being “like clearing a screen in a video game, but better.” One student told us that she worked on unassigned problems in Lean in order to procrastinate writing English essays. Many students reported that, if anything, writing ordinary mathematical proofs was harder than writing formal proofs in Lean: at least with Lean they knew what the rules were, whereas they could not anticipate how their informal proofs would be received by the person grading them.

The juxtaposition of formal and informal language led to interesting classroom discussion. Students were interested to discover that from a logical perspective the locution “ A , but B ” functions the same as “ A and B ” and the locution “ A unless B ” functions the same as “ A or B .” This led them to reflect as to what information is conveyed by the natural-language variants. Some of our discussions focused on the ability of symbolic logic to clear up ambiguity, for example, by distinguishing between an inclusive and exclusive “or.” We also discussed the fact that the phrase “everybody loves somebody” can be interpreted ambiguously as asserting that for every person there is another person they love, or that there is a single person that everybody loves. It is then interesting to see how the symbolic use of quantifiers resolves the ambiguity. Students got used to relativizing quantifiers, that is, translating phrases like “every car is red” and “some car is red” into the language of first-order logic.

Because we asked students to prove ordinary set-theoretic identities right after we completed the chapters on first-order logic, their natural tendency was to use symbols like quantifiers and arrows in their informal proofs. We explicitly told them not to do that, on the grounds that ordinary mathematical language favors using the words “every,” “some,” and “if...then” instead. Students were at first surprised and even skeptical of the claim that books and papers in ordinary mathematics, outside of formal logic, rarely use the logical symbols. After all, they are acutely aware that symbolic notation is central to mathematics. This led to interesting (and speculative)

discussions as to why natural language is used for the logical connectives. Perhaps it can be attributed to the vagaries of linguistic convention, but perhaps it is because we find it easier to follow arguments verbally, as we recite them to ourselves. We also discussed the ways that the use of natural language can signal nuances, such as the use of the word “but” in a proof to signal either something unexpected, a contradiction, a fact that somehow complements a statement that has come right before, or the end of a proof.

The juxtaposition of formal proof and informal proof also facilitates another important discussion. In a formal proof it is clear that every step has to follow one of the given rules, but how much detail is needed in an informal proof? This question highlights the fact that the purpose to writing an informal proof is not just to be correct, but also to convince someone else of the validity of a result and help them understand *why* it is true. Of course, these issues can also be discussed in an ordinary mathematics class, but the contrast with formal proof helps make the differences salient and meaningful.

We plan to continue developing the textbook, for example, with a chapter on algebraic structures and another on probability. More interestingly, as Lean evolves and gains new libraries and functionality, we plan to extend its use as well. We will use our pedagogical goals as a guideline: anything that provides useful insight into the nature of mathematical proof is worthwhile, whereas things that serve as a distraction from that goal are best avoided.

As we have taught it, the course covers a lot of ground. It requires commitment from the students, a general aptitude for mathematics, and comfort with computer languages and their precise syntactic requirements. It could easily be expanded to a longer course, which would allow for a more leisurely pace, greater depth, and the inclusion of additional topics.

Our course aims primarily to prepare students for higher-level mathematics courses, introducing them to the definition-theorem-proof style of presentation, and making them consciously aware of mathematical norms and expectations of rigor. At the same time, the course prepares students for more advanced courses in logic: by the time they are done, they are comfortable speaking in the language of first-order logic, which is to say, reading, writing, and interpreting formal expressions. This puts them in a good position to appreciate metatheoretic results about syntax and semantics, including results regarding provability, definability, and completeness. Finally, the course offers good preparation for branches of computer science that invoke logical notions, including database theory, automated reasoning, and formal verification. In a number of branches of that discipline, facility with formal languages, formal rules, and semantic notions is essential.

We are by no means the only ones to use software to teach mathematical proof. Jon Barwise and John Etchemendy’s *Hyperproof* (Barwise and Etchemendy, 1994) is an early and notable example of using software to teach proof and general logical reasoning. Daniel Velleman offers an online Java applet for writing proofs in elementary set theory to accompany his popular introductory text, *How to Prove It* (Velleman, 2006). The *AProS* project (Sieg, 2007), also at Carnegie Mellon, uses an interactive proof tutor and allows students to write proofs in first-order logic and

elementary set theory. Nathan C. Carter and Kenneth G. Monks are developing a mathematical word processor, *Lurch* (Carter and Monks, 2016), that can track the logical structure of a proof and verify inferences using back-end automation tools. They have used *Lurch* successfully in proof-based mathematics courses. In all the examples just cited, the approaches and technologies are slightly different from ours, but we expect that the pedagogical benefits are largely the same as the ones reported here. The goal of this chapter is just to share our own experience and observations, and to add our voices to the mounting chorus in support of using such tools to teach students how to read and write mathematical proofs.¹

Although this chapter provides anecdotal evidence to support the claim that interactive theorem proving software helps teach students mathematics, it does not constitute hard data. Work is needed to develop suitable quantitative measures and careful means of assessment. It is also an interesting question as to the extent to which mathematical reasoning skills transfer to other domains (Inglis and Attridge, 2017). Our experiences have shown that using formal methods to teach mathematics is at least possible, and, in the right circumstances, enjoyable. We hope they encourage others to give it a try.

Acknowledgements I am grateful to Mateja Jamnik and Keith Jones for helpful comments, corrections, and suggestions.

References

- Avigad, J., Lewis, R. Y., & van Doorn, F. (2019). *Logic and Proof*. http://leanprover.github.io/logic_and_proof/.
- Barwise, J., & Etchemendy, J. (1994). *Hyperproof*. Stanford, CA: CSLI Publications.
- Carter, N. C., & Monks, K. G. (2016). From formal to expository: using the proof-checking word processor *Lurch* to teach proof writing. In Schwell, Steurer & Vasquez (Eds.), *Beyond lecture: Techniques to improve student proof-writing across the curriculum*. MAA Press. <http://lurchmath.org/>.
- de Moura, L., Kong, S., Avigad, J., van Doorn, F., & von Raumer, J. (2015) The Lean theorem prover. In A. P. Felty & A. Middeldorp (Eds.), *Proceedings of the 25th International Conference on Automated Deduction (CADE-25)* (pp. 378–388). Berlin: Springer. <https://leanprover.github.io/>.
- Inglis, M., & Attridge, N. (2017). Does mathematical study develop logical thinking. *Testing the Theory of Formal Discipline*. London: World Scientific.
- Sieg, W. (2007). AProS project: Strategic thinking and computational logic. *Logic Journal of the IGPL* 15(4), 359–368. <http://www.phil.cmu.edu/projects/apros>.
- Velleman, D. (2006) *How to prove it: A structured approach* (2nd ed.). Cambridge: Cambridge University Press. <https://app.cs.amherst.edu/~djvelleman/pd/pd.html>.

¹Logic-based software is also often used to teach other topics in logic and computer science. The web page https://avigad.github.io/formal_methods_in_education/ provides links to some resources.

Web-Based Task Design Supporting Students' Construction of Alternative Proofs



Mikio Miyazaki , Taro Fujita  and Keith Jones 

1 Improvement of Proof Construction by Using Technology

The teaching and learning of proof and proving is acknowledged globally as a crucial part of mathematics education (Hanna & de Villiers, 2012) not only for echoing the nature of mathematics, but also for cultivating generic competencies of authentic explorative thinking (Miyazaki & Fujita, 2015). Yet students at the secondary school level (and beyond) suffer serious difficulties related to constructing and evaluating proofs in mathematics in general, and in geometry in particular (e.g. McCrone & Martin, 2004).

In order to improve on these difficulties, learning environments with technology for proof learning has been developed in two directions. One direction relates to developments in Artificial Intelligence (AI) by offering environments that focus on what might be considered more formal aspects of proving and how learners might be guided to construct correct proofs by appropriate feedback on their proofs and proving (from early systems, e.g. Anderson, Boyle, & Yost, 1986, to current initiatives, e.g. Wang & Su, 2017). The second direction is characterized by dynamic geometry environments (DGEs, such as *Cabri Express*, *Sketchpad Explorer* and *GeoGebra*). This direction has contributed to stimulating the use of conjecturing and the dialectical relationship between proofs and refutations in mathematics classrooms (e.g. González & Herbst, 2009; Komatsu & Jones, 2019). A slightly different approach to

M. Miyazaki (✉)
Shinshu University, Nagano, Japan
e-mail: miikun.miikun@gmail.com; mmiyaza@shinshu-u.ac.jp

T. Fujita
University of Exeter, Exeter, UK
e-mail: T.Fujita@exeter.ac.uk

K. Jones
University of Southampton, Southampton, UK
e-mail: d.k.jones@soton.ac.uk

© Springer Nature Switzerland AG 2019
G. Hanna et al. (eds.), *Proof Technology in Mathematics Research and Teaching*, Mathematics Education in the Digital Era 14,
https://doi.org/10.1007/978-3-030-28483-1_14

the two directions is to create a domain-specific learning space or environment for students (e.g. Cabri-Eulid (Luengo, 2005)). In our study we work on integrating technologies into daily mathematics lessons so teachers and learners use technologies effectively to advance their learning (for our technology, see Sect. 3.2.3). We argue that this approach might improve the status quo of proof learning at the secondary school level, especially in terms of constructing proofs as this is not yet satisfactory.

One way to improve students' capabilities related to constructing proofs by means of technology is to focus on the strategic knowledge needed by learners during not only proof constructions (Weber, 2001) but also during proof 'reconstructions' where students need to consider and apply this knowledge in order to change their proofs into alternatives. As such, and given that proving tasks with technology may contribute to developing strategic knowledge required for constructing alternative proofs, the purpose of this chapter is to explore how a proving task with technology can be designed to encourage students' emerging strategic knowledge of how to construct alternative proofs to the same problem, and how the designed task might enrich learners' strategic knowledge in proving in the context of geometrical proof that is commonly used to teach deductive proofs and proving in lower secondary schools (Fujita & Jones, 2014; also see Sect. 4.1).

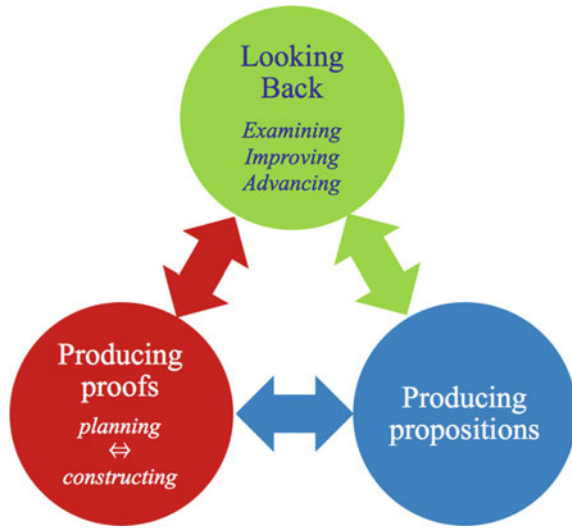
2 Strategic Knowledge of How to Construct Alternative Proofs to the Same Problem

2.1 *Constructing Alternative Proofs in the Process of Explorative Proving*

Luengo (2005, p. 13) views the construction of a mathematical proof as "a problem solving activity". In line with this, we take proving in mathematics is a fallibilistic activity (e.g. Lakatos, 1976) that involves producing statements inductively/deductively/analogically, planning and constructing proofs, looking back over proving processes and overcoming global/local counter-examples or errors, as well as utilizing already-proved statements in the context of working on further proofs (see Fig. 1). We define *explorative proving* as having the following three phases that inter-relate: (1) producing propositions, (2) producing proofs (planning and construction), and (3) looking back (examining, improving and advancing) (Miyazaki & Fujita, 2015).

In the process of explorative proving, students may need/want to be challenged to construct alternative proofs. When confronting mistakes and errors in their proving, they try to overcome them if they need to. When noticing more sophisticated ideas, they are willing to improve their previous proofs. At all these times, they may need to change the used theorems, assumptions or conclusions, rephrase their statements more clearly and so on. More globally, they sometimes need to switch from a direct proof to an indirect proof or vice versa, or overcome circular arguments (Fujita, Jones, & Miyazaki, 2018), and so on. We call such activities of producing different

Fig. 1 Explorative proving



proofs based on already constructed proofs, ‘reconstructing (either correct/incorrect) proofs’.

2.2 Forms of Strategic Knowledge of How to Reconstruct Proofs

Strategic knowledge, the ‘knowing-how’ in problem solving (Greeno, 1978), enables students to “recall actions that are likely to be useful or to choose which action to apply among several alternatives” (Weber, 2001, p. 111). Such knowledge plays an important role in explorative proving.

Strategic knowledge of how to reconstruct proofs has a domain-specific aspect related to achieving successful resolutions in the concerned domain. Especially, proof problems usually request students to show the logical connection between premises and conclusions. In the solving process, students are engaged in the activity, including constructing and reconstructing proofs as appropriate to the proof problem. As such, we propose the following three types of strategic knowledge of how to reconstruct proofs; *strategic knowledge for constructing proofs*, *strategic knowledge for reconstructing proofs*, *strategic knowledge appropriate for solving the proof problem*. We consider each in turn.

Strategic knowledge for constructing proofs (SKC) is used to connect premises and conclusions in order to form an original proof even if the proof contains some errors. This can involve, at a minimum, distinguishing conclusions and premises, deciding which theorems and/or definitions can be applied, and arranging the ways of thinking backward from conclusions to premises and thinking forward in the opposite

direction. Knowledge of thinking backward/forward, especially, can fundamentally inform students' proof construction (e.g. Heinze, Cheng, Ufer, Lin, & Reiss, 2008). This can work successfully, particularly when supported by more general strategic knowledge such as 'planning' (Schoenfeld, 1985), 'working backward' (Anderson, Corbett, Koedinger, & Pelletier, 1995), metacognition, and so on. For example, in constructing a geometrical proof, students often need to choose an appropriate triangle congruent condition before choosing which singular propositions can be used as premises. In this case, they use SKC related to deciding which theorems and/or definitions can be applied to the particular proof problem.

Strategic knowledge for reconstructing proofs (SKRc) is used to evaluate the previous proofs and improve/advance them. This can involve checking premises (definitions, axioms, etc.) and the validity of the used theorems during reconstructing proofs. Moreover, from a local point of view, it also involves organizing the elements of proofs to overcome any errors. For example, in the case of geometrical proofs, students often need to switch between three types of congruent triangle conditions in accordance with the problem conditions. In this case, they use SKRc related to validating the used theorems that can be applied.

Finally, strategic knowledge appropriate for solving the proof problem (SKSP) is used to supplement the parts of the logical connection in constructing and reconstructing proofs by using SKC and SKRc. This can involve considering the conditions of the problem to be solved and using theorems appropriate to them. For example, even if students can choose or switch an appropriate triangle congruent condition in constructing a geometrical proof, they also need to find/replace pairs of sides/angles, or triangles correctly. In this case, they use SKSP related to using theorems according to the problem conditions.

3 Task Design to Develop Strategic Knowledge of How to Reconstruct Proofs

In developing students' strategic knowledge of how to reconstruct proofs, it is necessary to organise activities for students that cultivate their strategic knowledge in proving. In this section we discuss key ideas for our task design and how technology is used to support students' learning.

3.1 Necessary Components of a Task for Problem Solving

A 'task' generally means an activity that needs to be undertaken and accomplished. In education, a task is expected to have distinct roles to guide learners' attention and interests towards desirable aspects of the concerned content (Doyle, 1983) that, in the context of mathematics education, encompasses thinking about, developing,

using, and making sense of mathematics (Stein, Grover, & Henningsen, 1996). For more on task design in mathematics education, see the various chapters in Watson and Ohtani (2015).

In order to undertake and accomplish a task as an intended activity in education, the task should be designed with essential components that lead learners toward the targets of the activity. Given that an intended activity is related to problem solving, the two questions ‘What should be solved?’ and ‘How should it be solved’ should be addressed. The first can be answered by the characteristics of a problem to be tackled. The second question has two aspects; a process to solve the problem and an environment that encourages the process. The former can be designed for managing and directing learners’ solving processes toward appropriate ones. The latter can be designed for supporting learners’ solving processes. Therefore, in this study, three components are adopted to design a task to develop strategic knowledge of how to reconstruct proofs: a *problem* to be tackled, a *learning environment* where to solve the problem, and a *process* to solve the problem.

3.2 Three Components of a Task Designed to Develop Strategic Knowledge of How to Reconstruct Proofs

Here we consider “what to design; which tools are necessary, or beneficial, for task design and under what conditions” (Jones & Pepin, 2016, p. 115). In our task design, we use ‘open problems with flow-chart proofs’ (what to design), web-based proof support system (hereinafter ‘the system’) that provides automatic translations of figural to symbolic objects and feedback in accordance with the type of errors (which tools are necessary, or beneficial) and specially-designed worksheets for students (under what conditions). We explain the detail below.

3.2.1 Open Problems with Flowchart Proofs

To develop strategic knowledge of how to reconstruct proofs, domain-specific strategic knowledge is applied in the problem-solving process consisting of a series of propositions and their transformations. In order to actualize, and be conscious of, this domain-specific strategic knowledge, open problems with a flow-chart proof format can be adopted. Open problems encourage students to construct multiple solutions by deciding the assumptions and intermediate propositions necessary to deduce a given conclusion. A flow-chart proof format can demonstrate the logical chains of propositions, and support students to construct proofs systematically.

Based on a scaffolding analysis (Sherin, Reiser, & Edelson, 2004), the use of open problems with flowchart proofs can enhance the structural understanding of proofs because it is expected that flow-chart format provides a visualization of the structural aspects of proofs in geometry, and encourages thinking backward/forward interac-

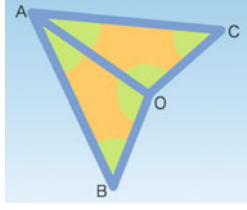
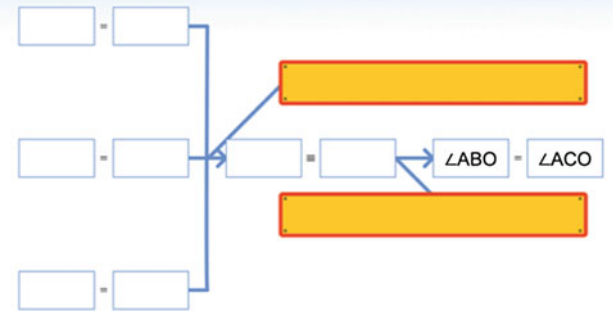
<p>In the right-hand side diagram, you can prove $\angle ABO = \angle ACO$ by showing that these triangles are congruent. What else do you need to add to prove this? What type of condition of congruence and what property of congruent figures do you use in there? Let's complete the flow-chart!</p>	
	

Fig. 2 Example of open problems with flow-chart proofs

tively because learners freely choose assumptions to prove the conclusion (Miyazaki, Fujita, & Jones, 2015). Open problems can function successfully in developing strategic knowledge of how to reconstruct proofs because students are encouraged to switch from an existing proof to an alternative proof by appropriate use of the viewpoint of proof structure; that is, “the relational network via deductive reasoning that combines singular and universal propositions” (Miyazaki, Fujita, & Jones, 2017, p. 226).

For example, the problem in Fig. 2 is intentionally designed so that students can freely choose which assumptions they use to draw the conclusion $\angle ABO = \angle ACO$. One solution is to show $\angle ABO = \angle ACO$ by using SSS condition. Nevertheless, other solutions can be found. One alternative solution might be to use $AO = AO$ as the same line and hence $\triangle ABO \cong \triangle ACO$ can be shown by assuming $AB = AC$ and $\angle OAB = \angle OAC$ using the SAS condition. During the time to find these solutions, by using flow-chart proofs format students can distinguish theorems and pairs of sides/angles/triangles, and then think backward/forward flexibly.

3.2.2 Learning Environment with Web-Based Proof Learning Support System

In order to support students to tackle open problems, our web-based system has three characteristics; translating automatically figural to symbolic objects, reviewing a learner’s answer, providing systematic feedback during/after constructing proofs (Miyazaki, Fujita, Jones, & Iwanaga, 2017; Fujita, Jones, & Miyazaki, 2018).

Our system has some similarities to the Cabri-Euclid system designed by Luengo (2005). The Cabri-Euclid system supports learners' proof construction process with workspaces dedicated to dynamic representations of geometrical figures, graphs to represent algorithms, and text spaces for exploiting direct manipulation and a set of operators to express statements and organise them under a precise "proof" "format" (p. 4). At the same time, Cabri-Euclid gives feedback to learners to support their proof construction process, e.g. "if the student attempts a deduction without giving a deduction rule, the software will ask the student for a theorem or a definition." (p. 11).

Cabri-Euclid is a powerful tool to support learners' proof construction process, but one issue is that students need to learn specific ways of inputting their texts as "The textual component of Cabri-Euclide has a limited linguistic capability." (ibid., p. 5). As we have described above, our system adopts a flow-chart format as we consider this is more intuitive for novice learners so that they do not have to learn specific commands and language to use the system.

Unlike Cabri-Euclid, the diagrams in our system for representing geometrical figures are static, but this is because of a technological reason—the interface of the system translates automatically from figural to symbolic objects. For example, to insert the symbol $\triangle ABO$ into the target box of flow-chart proof, by clicking the triangle on the diagram, the character gives message " $\triangle ABO$ closed", and red circles appear on the possible boxes (Fig. 3). Next, by clicking the proper box, the symbol $\triangle ABO$ appears inside the box. The opposite way (firstly click the box, then click the triangle) can also be executed. This automatic transformation of symbols is intended to reduce the representational barrier that students encounter when proof writing.

In order to complete the open problem, students are expected to seek multiple solutions, and, in such processes, are expected to find it useful to review the solutions that they have already found. Similarly, in the case of open problems with flow-chart proofs, students might find difficulties in distinguishing whether the on-going proof is different from their previous proofs. Our system supports students to review their correct proofs. For example, the problem shown in Fig. 2 has four kinds of solutions indicated with the number of stars '★'. If learners find correct solutions, the stars corresponding to these solutions are filled in yellow with an icon that shows the used congruent triangle condition. In the case illustrated (Fig. 4), the learners have already found two solutions with different conditions. As part of their search for the two remaining solutions, they can review their previous answers by clicking each yellow star.

In the process of proving, there can occur different kinds of errors. Students may notice some errors, but not others, and it is necessary to give support for this, e.g. by providing feedback. However, it is not an easy task to organize automated feedback from the computer. In fact, this was an issue for Cabri-Euclid – Luengo wrote that "during experimentations, ... the erroneous message is not understood by the students because there are several errors and the system give the first that it found." (p. 26). In order to deal with such situations, our system provides systematic, user-friendly feedback during/after constructing proofs and this is perhaps one of the most important technological features of our system (for the examples of learners' use of the

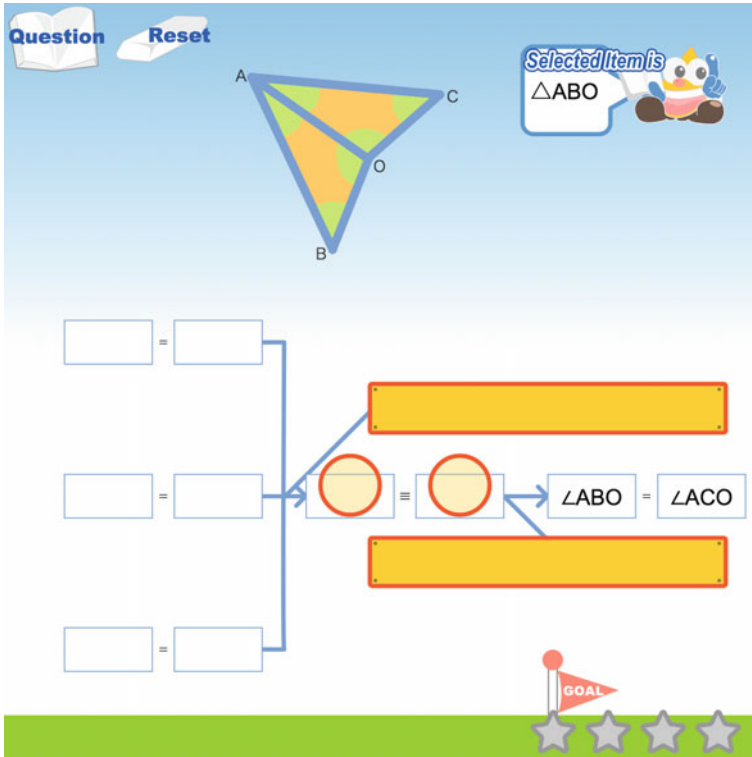


Fig. 3 Interface of web-based proof learning support system

system’s feedback, see Fujita, Jones, & Miyazaki, 2018). There are four categories of feedback; Category A: errors related to hypothetical syllogism, Category B: errors related to universal instantiation, Category C: errors related to singular propositions, Category D: errors related to proof-format. Category A, B, C are based on a viewpoint of proof structure. For example, Category A includes a logical circulation that use conclusions as assumptions (Fig. 4), Category B includes an error of choosing theorems, and Category C includes an error of choosing pairs of sides/angles. Category D includes proof-format errors, usually rather trivial ones related to singular propositions. Particularly in geometrical proof, a singular proposition concerning angles or sides should correspond to a singular proposition concerning triangle congruency.

3.2.3 Process of Expressing Strategic Knowledge of How to Reconstruct Proofs

In reconstructing proofs, the applied strategic knowledge is fundamentally implicit for students. Therefore, there needs an intervention that includes strategic knowledge of how to reconstruct proofs to enable them to express their ideas on changing

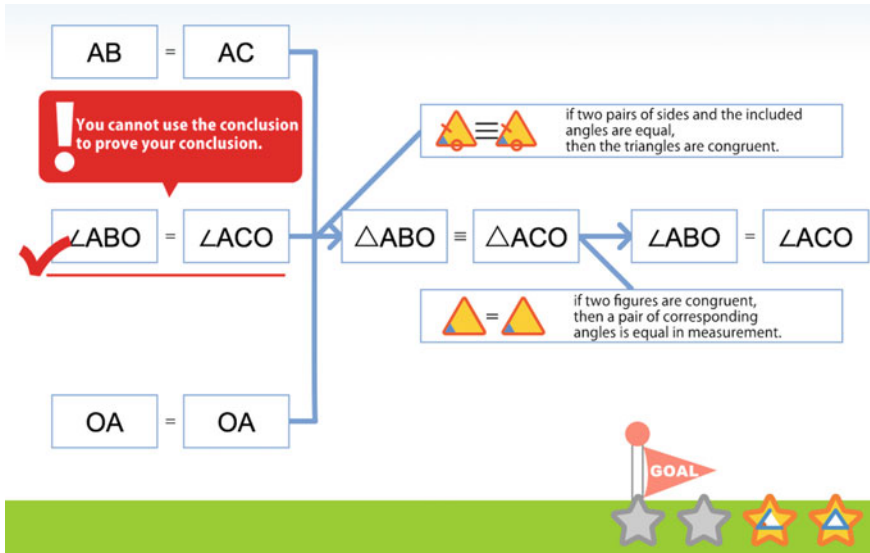


Fig. 4 Example of feedback of category A

their proofs into alternatives. Expressing the applied strategic knowledge encourages students to make explicit, and notice, their own ways of reconstructing proofs. This recognition enables them to apply their own strategic knowledge intentionally, and to expand the range of its application.

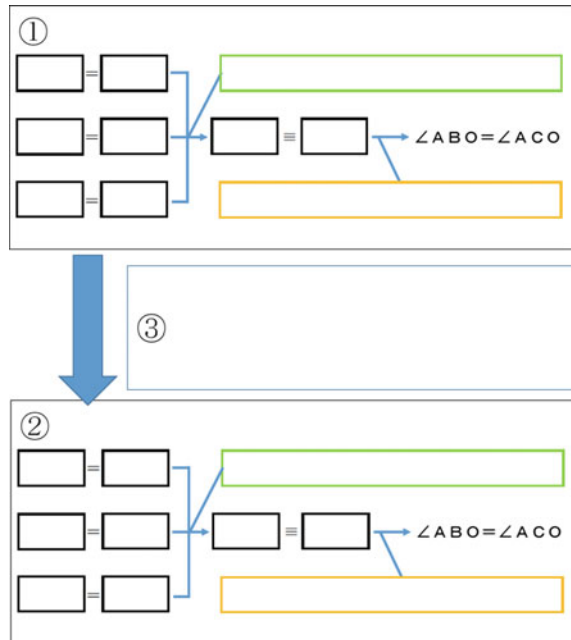
In order to realize the intervention, our approach is to devise a worksheet comprising some boxes that facilitate students expressing their ideas on changing their proofs into alternatives; see Fig. 5. Using the worksheet, students first write a proof checked by our system in ①. After finding an alternative proof by using our system, they write this proof in ②, and then express their ideas on changing their proof into an alternative in ③. The worksheet consists of a series of boxes. The number of boxes depends on how many correct proofs there are for the open problem. For example, the problem in Fig. 3 has four correct proofs so the worksheet for the problem consists of three boxes connected successively; as shown in Appendix 1.

4 Experimental Lessons

4.1 Learning Deductive Proving in Geometry in Japan

In Japan, deductive proof is explicitly taught in ‘Geometry’ in Grade 8. Although the Japanese national ‘Course of Study’ prescribes no official teaching sequence, some kinds of progressions can be found in the seven authorized textbooks (Fujita &

Fig. 5 Worksheet used by students



Jones, 2014). Most schools follow the progressions in the textbooks. Building on geometrical reasoning in earlier grades, students in Grade 8 gradually access deductive proofs through studying properties of angles and lines, triangles, and quadrilaterals. After learning congruent triangle conditions, they reach deductive proofs through learning the structure of deductive proofs and how to construct the proofs, and then explore and prove properties of triangles and properties of quadrilaterals. Recently, explorative proving (see Sect. 2.1) is also emphasized in teaching and assessment (Miyazaki & Fujita, 2015).

4.2 General Information of Classes and the Plan of the Experimental Lessons

The experimental lessons were carried out in a grade 8 class of an attached junior high school of a national university in Japan. The lessons took place after the students had learnt properties of angles and lines, triangles, and quadrilaterals, and used these in related proofs. The class had forty students, and was taught by a teacher with more than 20 years of teaching experience. The classes was relatively homogeneous with the most recent mathematics test scores being closely equal. Each lesson had the following instructional flow; understanding a problem (checking the conditions and the goal, reminding of what they learned, etc.), planning how to solve it (finding the

ways of solving and guessing the solution), solving it based on the plan, discussing ways to solve this type of problem, summarizing ideas of how to solve this type of problem.

Four lessons (each 50 min long) were planned, as below. The 1st and 2nd lessons were conventional and aligned with the authorized textbooks. In contrast, the 3rd and 4th lesson were designed to develop strategic knowledge of how to reconstruct proof shown as set out above (Sect. 3.2).

- 1st: Understanding the meaning of congruent figures and their properties.
Students learn that congruent figures can put on top of each other, and then solve a problem to find the sides of corresponding sides/angles of congruent figures by using the properties of congruent figures.
- 2nd: Understanding three conditions of congruent triangles.
Students explore the way of constructing congruent triangles, and then find three conditions of congruent triangles. Finally, students apply these conditions and to state them they used to find by solving a problem that requires to find congruent pairs among a lot of triangles by using.
- 3rd: Constructing flow-chart proof with conditions of congruent triangles.
Students solve an open problem with flow-chart proofs (Sect. 3.2.1) with conditions of congruent triangles that requires one step of deductive reasoning by using web-based proof learning support system (Sect. 3.2.2) as they express strategic knowledge of how to reconstruct proofs on their worksheet (Sect. 3.2.3).
- 4th: Constructing flow-chart proof with conditions of congruent triangles and properties of congruent figures.
Students solve an open problem with flow-chart proofs with conditions of congruent triangles and properties of congruent figures that requires two steps of deductive reasoning by using the system as they express strategic knowledge of how to reconstruct proofs on their worksheet.

4.3 Data and Analysis

Data from 3rd and 4th lessons that adopted our task design principles were collected by three video cameras. One recorded the whole class activity, the others recorded two students' activity individually. These students were chosen by the mathematics teacher based on daily performances and behaviors. For the purposes of this paper, we select one student, Tatsumi (all names are pseudonyms), to exemplify emerging strategic knowledge of how to reconstruct proofs by using the web-based proof learning support system effectively. His strategic knowledge was captured by his interactions with the system and writings on his worksheet concerning his ideas of how he changed his proofs into alternatives (we refer to such ideas as 'tips').

In what follows, we analyze the fourth lesson as this aimed at students' reconstructing a flow-chart proof with conditions of congruent triangles and properties of

congruent figures by using the designed task explained in Sect. 3. The lesson devoted time to each phase of its instructional flow as follows (time in minutes); understanding a problem (checking the conditions and the goal, reminding of what they learned, etc., 1:25), planning how to solve it (finding the ways of solving and guessing the solution, 2:30), solving it individually based on the plan (finding flow-chart proof by using the system, and writing their proofs and strategic knowledge on their worksheet, 28:21), discussing in the class how to solve it (presenting some students' strategic knowledge and sharing/improving them, 10:42), summarizing ideas of how to solve this type of problem (organizing their strategic knowledge and applying the other problem, 7:33). In our analysis, we focus on the student activity related to the emergence of strategic knowledge of how to reconstruct proofs. As well as student Tsunami, we also refer to the work of other students during the 'discussion' phase of the lesson.

5 Analysis of the Teaching Experiment

5.1 *Emerging Strategic Knowledge of How to Reconstruct Proofs*

In the lesson, student Tsunami easily constructed his proof with SSS condition by filling out all the cells of the flow-chart proof displayed in our proof learning support system, and wrote it down on his worksheet. Next, he changed " $BO = CO$ " to " $\angle ABO = \angle ACO$ ", and then changed the condition SSS to SAS. After checking this proof by the system, he also transcribed it to his worksheet, and after a while wrote his "Tips" for reconstructing proofs in ③ (see Fig. 6): "I noticed that a congruent triangle condition ought to be inserted in the green box, and also a property of congruent figures be inserted in the yellow box. So, I put a different condition for congruent triangles.". In this "Tips", he mentioned the order of the theorems used in the first proof, and the change of congruent triangle conditions. Thus, the former was triggered by the knowledge of keeping the order of theorems embedded in the previous proofs, and the latter by the knowledge of changing a theorem into another that can deduce the same property. These belong to the strategic knowledge for reconstructing proofs (SKRc).

In reconstructing the second proof to a third proof, and in keeping with SAS, Tsunami changed " $AB = AC$ " to " $BO = CO$ " and then changed " $\angle ABO = \angle ACO$ " to " $\angle BOA = \angle COA$ ". After checking this proof using the web-based system, he also transcribed it to his worksheet, and after a while wrote his "Tips" as follows: "I made the second proof with SAS. In a pair of triangles, I noticed there were two way of using SAS, then I kept the shared side OA and chose the different sides.". In this "Tips", he mentioned how to change a pair of sides with keeping up SAS. Thus, this was triggered by the knowledge of changing the elements of proofs with remaining the theorem used in previous proofs. This belongs to the strategic knowledge proper

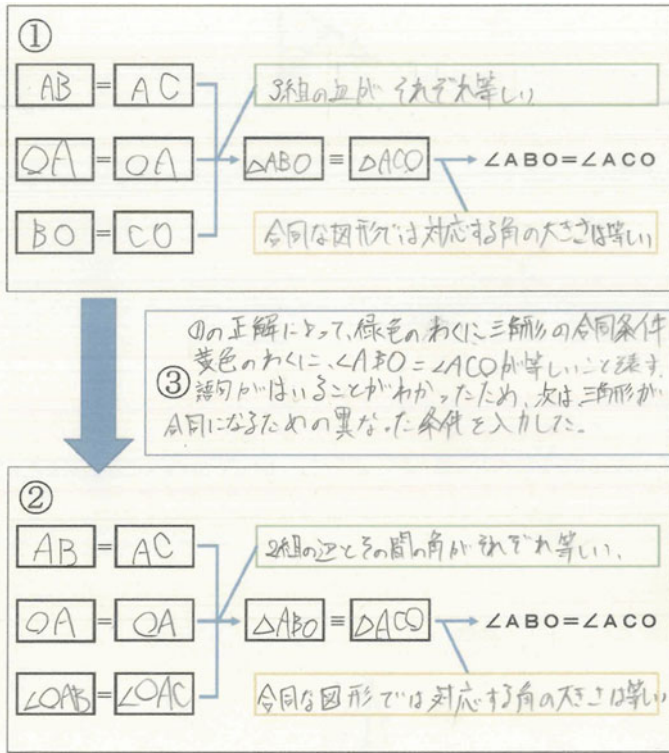


Fig. 6 Tatsumi's description in the worksheet

to reconstructing proofs (SKRc). Notably, he pointed out there were two ways of using SAS. As he avoided using this problem's conclusion, " $\angle ABO = \angle ACO$ ", in his proofs, this is likely to mean that he functioned with knowledge of avoiding logical circularity in his strategic knowledge for constructing proofs (SKC).

In reconstructing from his third to his fourth proof, he further changed " $BO = CO$ " to " $\angle OAB = \angle OAC$ ", and checked his proof, then received from the system the feedback "Is this theorem OK? If you want to use it, what do you need to use?". Based on this feedback, he realized his error, changed SAS to ASA, and then he completed all four correct proofs (Fig. 7).

After writing his fourth proof on his worksheet, he wrote his "Tips": "I chose the condition that I didn't use yet. The shared side OA is necessary to make proofs, then I chose the pairs of angles that included the side OA.". In this "Tips", he mentioned changing congruent triangle conditions and changing elements of proofs based on the givens of the problem. Thus, the former was triggered by the knowledge of changing a theorem into another that can deduce the same conclusion. This belongs to the strategic knowledge for reconstructing proofs (SKRc). Moreover, the latter was triggered by the knowledge of using theorems according to the conditions of the

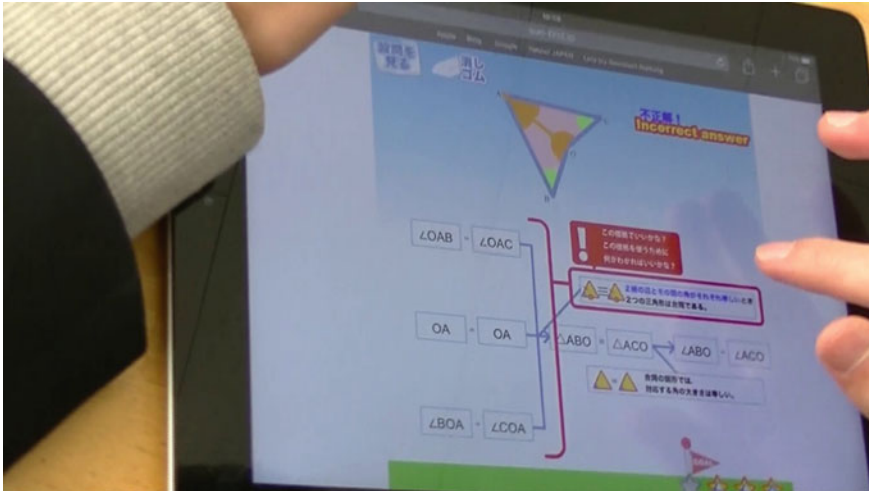


Fig. 7 System’s feedback for using congruent triangle conditions appropriately

problem to be solved. This belongs to the strategic knowledge for solving the proof problem (SKSP).

5.2 *Sharing Strategic Knowledge of How to Reconstruct Proofs in Class*

At the start of the phase of the lesson ‘discussing ways to solve this type of problem’, the class teacher asked four students to write on the blackboard their proof to the flow-chart problem. The teacher explained that these four proofs, shown in Table 1, were typical among the students in the class, and encouraged the class to focus on the differences between the proofs (minutes 35:01–36:14).

For example, concerning the change from the first to the second proof, the teacher pointed out the mysterious picture drawn by student Shinnosuke on the blackboard to explain his idea of changing proofs (see Fig. 8), and guided him to explain the meaning of this picture as follows (minutes 36:34–37:32).

- T61: Shinnosuke, so I asked you to draw the ‘mystery’ pictures below but what are these pictures about? What were you thinking when you drew this?
- S105: Ah, yes, uh, I thought [my proofs] in that order.
- T62: Ah, OK, that order (the four pictures).
- S106: When I was working with the iPad, um, I thought this would be the least number of ways...
- T63: The least number of ways... What do you mean, the least number of ways?

Table 1 Proofs on the blackboard chose by Teacher

Proof	Assumption				Conclusion
1st	OA = OA	SSS	$\triangle ABO \equiv \triangle ACO$	CPCTC (Angle)	$\angle ABO = \angle ACO$
	AB = AC				
	BO = CO				
2nd	OA = OA	SAS	$\triangle ABO \equiv \triangle ACO$	CPCTC (Angle)	$\angle ABO = \angle ACO$
	AB = AC				
	$\angle OAB = \angle OAC$				
3rd	OA = OA	SAS	$\triangle ABO \equiv \triangle ACO$	CPCTC (Angle)	$\angle ABO = \angle ACO$
	BO = CO				
	$\angle BOA = \angle COA$				
4th	OA = OA	ASA	$\triangle ABO \equiv \triangle ACO$	CPCTC (Angle)	$\angle ABO = \angle ACO$
	$\angle OAB = \angle OAC$				
	$\angle BOA = \angle COA$				

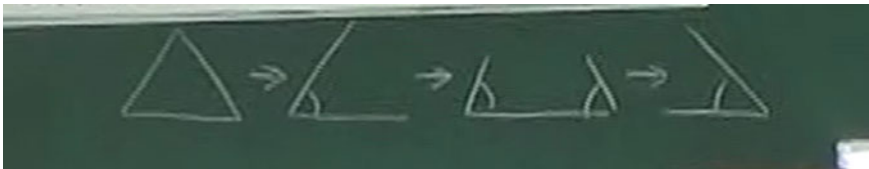


Fig. 8 Shinnosuke's mysterious picture

S107: Ah, well, that [the left SSS triangle], if I change one of the sides to angles, then I can have the second proof. Then change the side to angle, um, the third proof, and then angle to side, so the fourth one.

T64: That angles or that sides... Can you see? [saying to the other students in the class] Can you see? If you follow that [order], then we can complete with the least number of ways. The most efficient method, I would say. This way of thinking, it is interesting, isn't it? Yes? Three small claps!

As can be seen from this extract, Shinnosuke explained not only how to change the congruent triangle conditions, but also the reason why this order of change was very efficient. This suggests Shinnosuke applied his strategic knowledge of switching between theorems that can deduce the same property and of judging its efficiency. These belong to the strategic knowledge for reconstructing proofs (SKRc). After his explanation, the teacher wrote Shinnosuke's ideas on blackboard in order to record his strategic thinking, and praised him and his classmate.

Next, concerning the change from the second to the third proof, both of which use the SAS condition, the teacher asked student Yuki to explain the way and the reason of this change. Yuki explained that he chose the alternative pairs of side/angle different from the second proof because the common side OA had not been changed.

S118: OK, I have done ②, so, um, the sides OA are shared so I thought I cannot change. So I exchanged the rest of the side and angles.

This suggests that Yuki applied his strategic knowledge of changing the elements of proofs while retaining the theorem used in the previous proof. This belongs to the strategic knowledge for reconstructing proofs (SKRc). The teacher also wrote his idea on blackboard to share it in class.

In the last section of the lesson, on the change from the third to the fourth proof, the teacher asked student Narumi to explain the change, and the reason of this change. She showed her ideas as follows.

S119: Yes, um, um, as usual, OA, OA are always equal, so OA will be there. And the last one we need to prove angles $ABO = ACO$, so I used the other two angles.

T77: OK, I think you understood well. Do you see this (saying to the other students in the class)? Can you say it again? OK? He is going to say very important thing so please listen carefully!

S120: Um...

T78: These ($OA = OA$). We really need these ($OA = OA$).

S121: And, uh, the last thing I want to say is, uh, angles $ABO = ACO$, so, uh, of these three pairs of angles, I thought the other pairs than $ABO = ACO$.

T79: You thought about. That sounds very good (the others started clapping their hands). OK, angles ABO and ACO , and then... And then what? I forgot (laugh). What was it? What was the last thing you wanted to say?

S122: Equal.

T80: Yes, we want to say they are equal.

As above, Naomi pointed out that she did not choose " $\angle ABO = \angle ACO$ " because this is a conclusion to be proven and cannot be used as an assumption. This suggests that Naomi applied her strategic knowledge of avoiding logical circularity that belongs to strategic knowledge for constructing proofs (SKC). The teacher wrote Naomi's ideas that "We want to conclude angles $ABO = ACO$, so we cannot use this. Therefore, we have to find the other pairs of angles." on the blackboard, and praised her strategic thinking. Figure 9 shows how the blackboard was used during the 4th lesson.

By analyzing the data from our teaching experiment, we identified the students' strategic knowledge of how to construct alternative proofs to the same problem. In the case of student Tatsumi, as in the analysis above, we identified the strategic knowledge that emerged during the classroom activity.

In doing so, we found that we could exemplify the three categories of strategic knowledge that we set out in Sect. 2.2, viz:

- Strategic knowledge for constructing proofs (SKC)
 - Avoiding logical circularity (Tatsumi, Naomi)
- Strategic knowledge for reconstructing proofs (SKRc)

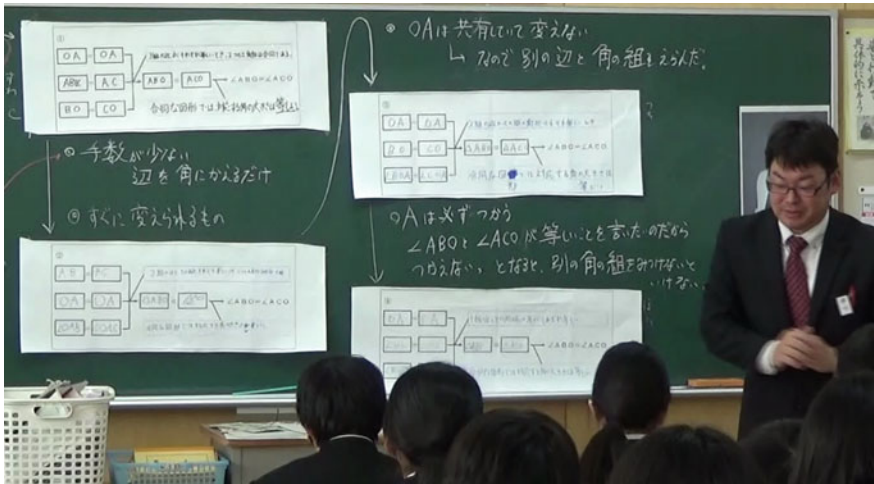


Fig. 9 Use of the blackboard in the 4th lesson (photo used with permission)

- Keeping the order of theorems embedded in the previous proofs (Tatsumi)
- Changing a theorem into another that can deduce the same property (Tatsumi)
- Changing the elements of proofs with remaining the theorem used in previous proofs (Tatsumi, Yuki)
- Switching between theorems that can deduce the same property (Shinnosuke)
- Judging the efficiency of switching between theorems (Shinnosuke)
- Strategic knowledge appropriate for solving the proof problem (SKSP)
 - Using theorems according to the conditions of the problem to be solved (Tatsumi).

6 Discussion

Most of the examples of strategic knowledge listed immediately above echo the characteristics of geometrical proof based on triangle congruency. For example, switching between theorems that can be used to deduce the same property strongly relates to the three conditions of congruent triangles. This type of knowledge belongs to the “knowledge of the domain’s proof techniques” that Weber (2001, p. 111) hypothesized as the type of strategic knowledge that undergraduates appear to lack.

In order for this strategic knowledge to emerge, it is necessary for students to notice it, and apply it, in the context of proof problem solving. Additionally, by expressing the knowledge students clarify and enrich their own thinking as well as noticing and applying it objectively. Moreover, the designed task to develop strategic knowledge of how to reconstruct proofs consists of three components described in Sect. 3. These

components and their interactions could contribute to students noticing, applying, and expressing their strategic knowledge.

In fact, Tatsumi firstly reconstructed his proof, and then could use his strategic knowledge that he had applied to reconstructing it as shown in Sect. 5.1. This activity was especially encouraged by open problem with flow-chart proofs and technologies of web-based proof learning support system. As shown above (Sect. 3.2.1), open problem with flow-chart proofs encourage students to construct multiple solutions by supporting them when deciding the assumptions and intermediate propositions. Similar to the activities illustrated in our previous studies (e.g. Miyazaki, Fujita, & Jones, 2015), the open problems with flow-chart proofs encouraged Tatsumi, on the one hand, to construct his proof, and reconstruct another based on the previous by switching the congruent triangle conditions and the properties of congruent figures, and changing the pairs of sides/angles. On the other hand, in every trail of reconstructing his proofs, he checked his proof by using the system and gained confidence to progress his proof solving. Additionally, he could take into account suggestions of how to resolve his error informed by the feedback of the system (e.g. “Is this theorem OK? If you want to use it, what do you need to use?”(see Sect. 5.2). Hence, the technologies of our system also encouraged him to reconstruct his proofs by providing systematic feedback according to his errors or mistakes, accompanied by offering a useful interface to switch between the elements of proof and suggesting whether his proof was correct or not.

Moreover, Tatsumi was encouraged to express his strategic knowledge by writing his ideas on the worksheet composed of three boxes (see Fig. 6) as shown in Sect. 5.1. Similarly, in analysis of the part of the 4th lesson on discussing how to solve the problem in the class (as shown in Sect. 5.2), students were also inspired to pay attention to the ideas of how to reconstruct proofs, rather than the four proofs whose correctness was certificated by the system. Thus, the process of expressing strategic knowledge of how to reconstruct proofs contributed to formulating these students’ strategic knowledge.

Concerning using technology, the system enables students to tackle open problem with flow-chart proofs efficiently and enhance their understanding of proof structure (Miyazaki, Fujita, & Jones, 2017), informed by the systematic feedback (Fujita, Jones & Miyazaki, 2018) that is more user-friendly than earlier systems such as Cabri-Euclid (Luengo, 2005). As a result, students could begin to build their strategic knowledge of ‘changing the elements of proofs whilst retaining the theorem used in previous proofs’ by understanding universal instantiation as well as the knowledge related to the elements of proof. Also in terms of proof structure, students encountered the strategic knowledge ‘avoiding logical circularity’ on the whole structure of proof. Moreover, the system enabled students to recognize a proof as a group of ‘modules’ (Weber & Mejia-Ramos, 2011; Mejia-Ramos, et al., 2012). In the case of geometrical proofs by using triangle congruency, most proofs have the same groups of ‘modules’ that combines a congruent triangle condition with a property of congruent figures. Thus, the strategic knowledge of ‘keeping the order of theorems embedded in the previous proofs’ is certainly a fundamental key to reconstructing proofs, and most students are implicitly getting to apply this knowledge by their experience of proof

construction. By using these technological advantages, open problems with flow-chart proofs came to be more effective for students to reconstruct proofs.

However, a teacher's instruction is another important factor (Parero & Aldon, 2016). In the 4th lesson, the teacher's instructions for students on how to find their proofs by using the system, and how to write their ideas clearly, took a role of organizing the process of expressing the strategic knowledge. Particularly, the instruction about looking back on how to think in changing their proofs was essential to expressing this knowledge. Moreover, the teacher praised Shinnosuke's idea concerning judging the efficiency of switching between theorems and Naomi's idea concerning avoiding logical circularity as shown in Sect. 5.2. These teacher's actions suggest what to be expressed as the strategic knowledge to reconstruct proofs.

7 Conclusion

This study explored how a proving task with technology can be designed to develop strategic knowledge of how to construct alternative proofs to the same problem, and how the designed task enriched learners' strategic knowledge in proving in the context of geometrical proof. For the former, the task we designed had three components; open problem with flow-chart proofs, learning environment with the web-based support system, and process of expressing strategic knowledge of how to reconstruct proofs. For the latter, through the observation of individual student activity, and the part of the 4th lesson on 'discussing ways to solve this type of problem', our analysis shows how these components, and their interactions encouraged students to notice, apply, and express this strategic knowledge.

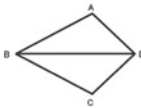
The adoption of this designed task in the proof lessons was found to encourage students to carry out the process of explorative proving accompanied with reconstructing proofs as necessary. What is more, the way that the tasks were designed can contribute to advancing research on task design. Especially, concerning the role of technology, the support system can make open problems with flow-chart proofs more effective for students to reconstruct proofs, and promote the process of expressing strategic knowledge of how to reconstruct proofs. Nevertheless, a teacher's appropriate involvement and participation can help to ensure that students acquire this strategic knowledge. It shows the necessity of refining this design task in cooperation with teachers (Jones & Pepin, 2016) and designing teaching to organize students' activities in the most productive way.

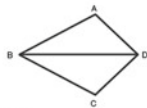
Acknowledgements This research was supported by the Grant-in-Aid for Scientific Research, Ministry of Education, Culture, Sports, Science, and Technology, Japan. Special thanks to Mr. Yasuyuki Matsunaga for data collection, and Mr. Daisuke Ichikawa for practicing the lessons.

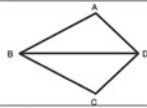
Appendix 2: Post-test After the 4th Lesson

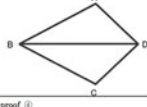
证明 相 等 的 角

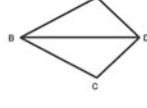
In the below diagram, you will prove $\angle BAD = \angle BCD$ by showing that these triangles are congruent. What else do you need to add to prove this? What type of condition of congruence and what property of congruent figures do you use in there? Let's complete the flow-chart, and write down your tips!



①  Tips to find proof ①

②  Tips to find proof ②

③  Tips to find proof ③

④  Tips to find proof ④

<input type="text"/>	<input type="text"/>	<input type="text"/>	
<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>
<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>

证明条件 Congruent triangle conditions
 证明后图形中的性质 CPCTC
 $\angle BAD = \angle BCD$

References

Anderson, J. R., Boyle, C. F., & Yost, G. (1986). Using computers to teach: The geometry tutor. *Journal of Mathematical Behavior*, 5(1), 5–19.

Anderson, J. R., Corbett, A. T., Koedinger, K. R., & Pelletier, R. (1995). Cognitive tutors: Lessons learned. *Journal of the Learning Sciences*, 4(2), 167–207.

Doyle, W. (1983). Academic work. *Review of Educational Research*, 53(2), 159–199.

Fujita, T., Jones, K., & Miyazaki, M. (2018). Learning to avoid logical circularity in deductive proofs through computer-based feedback: Learners' use of domain-specific feedback. *ZDM Mathematics Education*, 50(4), 699–713.

Fujita, T., & Jones, K. (2014). Reasoning-and-proving in geometry in school mathematics textbooks in Japan. *International Journal of Educational Research*, 64, 81–91.

González, G., & Herbst, P. G. (2009). Students' conceptions of congruency through the use of dynamic geometry software. *International Journal of Computers for Mathematical Learning*, 14(2), 153–182.

Greeno, J. G. (1978). A study of problem solving. In R. Glaser (Ed.), *Advances in instructional psychology* (Vol. 1). Hillsdale NJ: Lawrence Erlbaum Associates.

Hanna, G., & de Villiers, M. (2012). Aspects of proof in mathematics education. In G. Hanna & M. de Villiers (Eds.), *Proof and proving in mathematics education: the ICMI Study* (pp. 1–10). New York: Springer.

- Heinze, A., Cheng, Y.-H., Ufer, S., Lin, F.-L., & Reiss, K. (2008). Strategies to foster students' competencies in constructing multi-steps geometric proofs: Teaching experiments in Taiwan and Germany. *Zentralblatt für Didaktik der Mathematik*, 40(3), 443–453.
- Jones, K., & Pepin, B. (2016). Research on mathematics teachers as partners in task design. *Journal of Mathematics Teacher Education*, 19(2–3), 105–121.
- Komatsu, K. & Jones, K. (2019). Task design principles for heuristic refutation in dynamic geometry environments. *International Journal of Science and Mathematics Education*, 17(4), 801–824. <https://doi.org/10.1007/s10763-018-9892-0>.
- Lakatos, I. (1976). *Proofs and refutations: The logic of mathematical discovery*. Cambridge, UK: Cambridge University Press.
- Luengo, V. (2005). Some didactical and epistemological considerations in the design of educational software: The Cabri-Euclide example. *International Journal of Computers for Mathematical Learning*, 10(1), 1–29.
- McCrone, S. S., & Martin, T. S. (2004). Assessing high school students' understanding of geometric proof. *Canadian Journal for Science, Mathematics, and Technology Education*, 4(2), 223–242.
- Mejia-Ramos, J. P., Fuller, E., Weber, K., Rhoads, K., & Samkoff, A. (2012). An assessment model for proof comprehension in undergraduate mathematics. *Educational Studies in Mathematics*, 79(1), 3–18.
- Miyazaki, M., & Fujita, T. (2015). Proving as an explorative activity in mathematics education: New trends in Japanese research into proof. In B. Sriraman, et al. (Eds.), *First sourcebook on Asian research in mathematics education: China, Korea, Singapore, Japan, Malaysia and India* (pp. 1375–1407). Charlotte, NC: Information Age Publishing.
- Miyazaki, M., Fujita, T., & Jones, K. (2015). Flow-chart proofs with open problems as scaffolds for learning about geometrical proofs. *ZDM Mathematics Education*, 47(7), 1–14.
- Miyazaki, M., Fujita, T., & Jones, K. (2017a). Students' understanding of the structure of deductive proof. *Educational Studies in Mathematics*, 94(2), 223–239.
- Miyazaki, M., Fujita, T., Jones, K., & Iwanaga, Y. (2017b). Designing a web-based learning support system for flow-chart proving in school geometry. *Digital Experiences in Mathematics Education*, 3(3), 233–256.
- Panero, M., & Aldon, G. (2016). How teachers evolve their formative assessment practices when digital tools are involved in the classroom. *Digital Experiences in Mathematics Education*, 2(1), 70–86.
- Schoenfeld, A. H. (1985). *Mathematical problem solving*. Orlando: Academic Press.
- Sherin, B., Reiser, B. J., & Edelson, D. (2004). Scaffolding analysis: Extending the scaffolding metaphor to learning artifacts. *The Journal of the Learning Sciences*, 13(3), 387–421.
- Stein, M. K., Grover, B. W., & Henningsen, M. (1996). Building student capacity for mathematical thinking and reasoning: An analysis of mathematical tasks used in reform classrooms. *American Educational Research Journal*, 33(2), 455–488.
- Weber, K. (2001). Student difficulty in constructing proofs: The need for strategic knowledge. *Educational Studies in Mathematics*, 48(1), 101–119.
- Wang, K., & Su, Z. (2017). Interactive, intelligent tutoring for auxiliary constructions in geometry proofs. <https://arxiv.org/abs/1711.07154v1>.
- Watson, A., & Ohtani, M. (Eds.). (2015). *Task design in mathematics education: ICMI study 22*. Cham, Switzerland: Springer.
- Weber, K., & Mejia-Ramos, J. (2011). Why and how mathematicians read proofs: An exploratory study. *Educational Studies in Mathematics*, 76(3), 329–344.

Reasoning by Equivalence: The Potential Contribution of an Automatic Proof Checker



Christopher Sangwin

1 Introduction

The heart of mathematics is setting up abstract problems and solving them, and the outcome of this process is a “proof”. Polya (1962) identified four patterns of thought to help structure thinking about solving mathematical problems. His “*pattern of two loci*” is highly geometric. To use the pattern of two loci keep only one of the conditions needed and solve this simpler problem, expecting to get a set (locus) of solutions in this less constrained situation. Keep each condition in turn, and the overall solution is where the loci intersect. For example, when finding the tangents to a circle centred at O through an external point A , the tangent line must be at right angles to the radius. The set of points P for which OP is perpendicular to AP is the circle centered at the midpoint of OA through O . This is one locus, and the circle itself is the second. The “*superposition*” pattern also breaks constraints into separate parts in a more algebraic way. Examples include Lagrange interpolation, and solving linear differential equations. The “*recursion*” pattern solves a problem by using a smaller (or simpler) case, for example finding the binomial coefficients using Pascal’s triangle.

Legitimate patterns of thought directly translate into what is considered to be an acceptable proof. For example, recursion solves the problem and proof by induction is the resulting formal justification.

The “*Cartesian*” pattern is where a problem is turned into a system of equations, before the equations are solved using algebra. Note that the algebraic manipulation is the technical middle step in the process: setting up the equations and interpreting the solutions are essential parts to complete this pattern. My previous work Sangwin and Köcher (2016) examined questions set in school-level examination papers and found that line-by-line algebraic reasoning, termed by Nicaud, Bouhineau, and Chaachoua (2004) as *reasoning by equivalence*, is the most important single form of reasoning in school mathematics. This is closest to Polya’s “*Cartesian pattern*” and this predominates normative answers to school examination questions.

C. Sangwin (✉)

School of Mathematics, University of Edinburgh, Edinburgh, UK

e-mail: c.j.sangwin@ed.ac.uk

© Springer Nature Switzerland AG 2019

G. Hanna et al. (eds.), *Proof Technology in Mathematics Research*

and Teaching, Mathematics Education in the Digital Era 14,

https://doi.org/10.1007/978-3-030-28483-1_15

Reasoning by equivalence is essentially a symbol-pushing technique, which reduces algebraic reasoning to a mechanical calculation. There is nothing pejorative in describing this activity as symbol-pushing or as mechanical: in some senses this automation is liberating. Indeed Leibniz (1966) sought a “universal calculus” and Boole (1847) explicitly sought to replace syllogisms in language with symbolic calculus. This work continues with attempts to automate proof more generally, see Beeson (2004) for a survey.

This research is based on the following epistemological position: to successfully automate a process it is necessary to understand it profoundly. It follows that automation of a process necessitates the development of a certain kind of understanding. I am trying to automate the assessment of students’ line by line algebraic reasoning, including provision of effective feedback on their progress. As I hope to show, the attempt to implement a line by line assessment system reveals ambiguities, inconsistencies and outright errors in elementary algebra as currently taught and learned. For example, using algebraic rules uncritically outside the domains of definition such as $\sqrt{ab} = \sqrt{a}\sqrt{b}$ leads to contradictions such as $-1 = 1$, which have been examined elsewhere, e.g. Bernardo and Carmen (2009), Tirosh and Evan (1997) and Levenson (2012). While professionals might dismiss such errors as trivial they have been made by the very best mathematicians in the past (e.g. see Euler, 1822, p. 43, §148) and students regularly continue to do so (Kirshner and Awtry, 2004).

In this chapter I examine the role of reasoning by equivalence in mathematical proof. This chapter contains a number of sections. First I define the mathematics of reasoning by equivalence. By looking at what students are asked to do in high-stakes national examinations I examine the extent to which students are currently asked to “prove”, “show” or “justify” and the mathematics this involves. I then report research into how students go about solving such problems on paper. This evidence has been used to inform the design of software which assesses students’ responses, and I report on the use of such software with students. I then discuss the implications for what constitutes mathematical “proof” at school level, and how this might be taught and learned online.

2 Reasoning by Equivalence

In educational terms reasoning by equivalence is a loose collection of rules, such as “doing the same thing” to both sides of an equation. In Sangwin and Köcher (2016) we did not define the process more formally, indeed the rules of elementary algebra are not normally articulated carefully. When seeking to develop technology, a loose collection of rules is simply not sufficient or satisfactory and so in this section I define reasoning by equivalence.

The phrase “doing the same thing” to both sides of an equation has a focus on the individual steps, and the legitimacy of working in steps. Reasoning by equivalence does not follow a prescribed routine, with fixed size steps in working. Instead, the move between adjacent lines from one expression to the next is legitimate if and

Solve $x^2 - 6x - 16 = 0$, by factoring and working line by line.
 Leave your answer in the form $x = \dots$ or $x = \dots$ in fully simplified form.

$$x^2 - 6x - 16 = 0$$

$$(x-3)^2 - 9 = 16$$

$$(x-3)^2 = 25$$

$$x-3 = +5$$

$$x = -2 \text{ or } x = 8$$

$$x^2 - 6x - 16 = 0$$

$$\Leftrightarrow (x-3)^2 - 9 = 16$$

$$\Leftrightarrow (x-3)^2 = 25$$

$$\Leftrightarrow x-3 = \pm 5$$

$$\Leftrightarrow x = -2 \text{ or } x = 8$$

The variables found in your answer were: $[x]$

Fig. 1 Reasoning by equivalence in the STACK system

only if adjacent expressions are equivalent. In most situations many algebraic “steps” are combined, including associativity, commutativity and basic integer arithmetic. Hence, defining what is and is not a legitimate step turns out to be problematic in an educational context.

Let $X \subset \mathbb{K}$ where \mathbb{K} is a set of numbers given by the context, such as \mathbb{Q} , \mathbb{R} or \mathbb{C} . For the purposes of this chapter, I define “a correct argument” as an ordered list of mathematical expressions E_1, \dots, E_n so that all the E_j are equivalent. For example, in the case of equations in a single variable x this means the solution set of E_j is precisely the same for all $j = 1, \dots, n$. The role of \mathbb{K} is in deciding what is and is not a solution, e.g. $x^4 - 9 \equiv x^2 - 3$ in \mathbb{R} , but not in \mathbb{Q} or \mathbb{C} . Reasoning by equivalence is the process by which an individual takes successive representatives from a particular equivalence class.

The advantage of defining the correctness of an argument in terms of equivalence is that the student is free to take any route they choose. In an educational context, overall correctness will additionally require that E_1 is some specific expression (i.e. the problem to be solved) and that E_n is given as a specific form, such as $x = ?$ in the case of a linear equation. A teacher might also require some other properties from a student for an answer to be fully correct. An example of a student solving a quadratic equation using reasoning by equivalence, as implemented in the STACK system (see Sangwin, 2013), is shown in Fig. 1. In this case, the student has correctly solved the quadratic specified and the (automatically generated) \Leftrightarrow symbols¹ indicate the equivalence of adjacent lines. However, the final system-generated feedback (omitted in the figure) actually says “*The question asked you to solve the equation by factoring. The factored form should appear as one line in your working.*” and no numerical marks are awarded: the student has not followed the instructions in the question. This

¹The symbol \Leftrightarrow should be read “if and only if” just as $=$ is read “is equals to”. The symbol $:=$ is read as “is defined to be”.

example is typical of many educational situations. “Correctness” is a combination of properties, and correct line by line reasoning is only one of them.

Two expressions are equivalent if they take the same value when evaluated at each of the points in the domain of the variables. For example, $x^2 - 1$ is equivalent to $(x - 1)(x + 1)$ on the domain of real numbers. The difference between equations and expressions is illustrated by the following example.

$$|x - 1/2| + |x + 1/2| - 2 = 0 \Leftrightarrow |x| - 1 = 0$$

but

$$|x| - 1 \not\equiv |x - 1/2| + |x + 1/2| - 2 \text{ for all } x \in \mathbb{R}.$$

I shall write \equiv to indicate equality of expressions on their domain of definition, the third bar indicating the stronger condition. That is

$$p \equiv q \Leftrightarrow p(x) = q(x), \quad \forall x \in X.$$

Equivalence of equations can be defined in two useful ways, and for educational purposes both are needed. $V(p) = \{x \in X \mid p(x) = 0\}$ is called the *variety* defined by p . This is a geometric notion of the set of solutions, but the variety lacks any way to distinguish between repeated roots. Formally, and for completeness, for systems of polynomial equations the multiplicity of the roots can be distinguished by calculating the reduced Gröbner basis, with respect to a specified order of the variables which gives a canonical representation of the equivalence class, see Adams and Loustaunau (1994). For a set of polynomials in a single variable the Gröbner basis is given by the highest common factor, and hence provides the mutual solutions including multiplicity. For a set of linear equations in many variables the Gröbner basis is equivalent to the reduced echelon form, with respect to an ordering of the variables.

These abstract definitions do not help someone learn *how to solve* algebraic problems. The issue then is the algebraic operations applied to the expressions on both sides of an equation that provide an equivalent equation. For the purposes of this discussion, I restrict to single variable expressions over the set $X \subset \mathbb{K}$, the natural domain of the expressions. Solving the equation $p = q$ necessitates finding those values of $x \in X$ so that $p(x) = q(x)$. To legitimately do the same thing to both sides of an equation $p = q$, it is necessary to ensure that

$$p = q \Leftrightarrow f(p) = f(q).$$

Since f is a function

$$p = q \Rightarrow f(p) = f(q).$$

A function $f : X \rightarrow Y$ is *injective* if

$$\forall p, q \in X, f(p) = f(q) \Rightarrow p = q.$$

Hence “doing the same thing to both sides” is legitimate if and only if f is an injective function.

Note that multiplication by a term a is an injection if and only if $a \neq 0$. Cases when a is an algebraic term which might be zero give rise to various false arguments, see e.g. Maxwell (1959) and Northrop (1945). This problem can be entirely avoided by *auditing* to track the side condition $a \neq 0$, see Sangwin (2015).

A substitution² is a syntactic transformation of a formal expression in which a variable, sub-expression, or term, is consistently replaced by other expressions. For substitution assume $s: X \rightarrow X$ is a function, then substitution involves replacing a term x by $s(x)$. For this to be legitimate it is necessary to ensure that

$$p(x) = q(x) \Leftrightarrow p(s(x)) = q(s(x)), \quad \forall x \in X.$$

Since s is a function it follows immediately that

$$p(x) = q(x) \Rightarrow p(s(x)) = q(s(x)), \quad \forall x \in X.$$

However in general the converse is false. For example, let $X = \mathbb{R}$, $p(x) := x$ and $q(x) := |x|$ and $s(x) := x^2$. Then $x^2 = |x^2|$ but $x \neq |x|$ for all $x \in \mathbb{R}$. If $s: X \rightarrow X$ is surjective then

$$\forall x' \in X, \exists x \in X \text{ such that } x' = s(x).$$

The hypothesis $p \circ s \equiv q \circ s$ and the assumption of surjectivity of s prevents a counter example of the form $\exists x' \in X \ p(x') \neq q(x')$.

In summary, “doing the same thing to both sides” retains equivalence if and only if f is injective, and substitution retains equivalence if and only if s is surjective. In both these definitions the domain X is crucial.

I should acknowledge that since there is no conscious “direction of travel” or sense of “progress”, in educational terms, reasoning by equivalence looks very odd. It is entirely possible for correct reasoning to have repetitive steps, unnecessary loops or digressions. Aesthetic judgements are a separate matter from correctness, as is what might be considered an appropriate “jump” or level of detail. Only further experience and development will allow us to decide if additional aesthetic measures can be automated. For example, an aesthetic measures might include “distance from a model answer”.

A more serious, and unfortunate, consequence of this definition is that some correct equivalence reasoning arguments do not correspond to correct mathematical steps.

²Note, by “substitution” I mean capture-avoiding substitution. A substitution is said to be a *capture-avoiding substitution* when the process avoids accidentally allowing free variables in the substitution to be captured inside the original expression. For example, in the function $x \mapsto xy$ if I replace y with x , the function would become $x \mapsto x \times x$ which is different. Both x s now refer to the argument of the function. The second x in $x \mapsto x \times x$ which was originally “free” has been “captured”.

Table 1 Descriptive statistics of marks available on Higher Mathematics questions from 2015

	M	A	R	N
# of marks	65	50	3	12
%	50	38	2	9
IB %	31	50	4	15

[M] Marks awarded for attempting to use a correct Method; working must be seen

[A] Marks awarded for an Answer or for Accuracy: often dependent on preceding M marks

[R] Marks awarded for clear Reasoning

[N] Marks awarded for correct answers if no working shown

$$\begin{aligned}
 x^2 - 6x + 9 &= 0 \\
 \Leftrightarrow (x - 5)(x - 1) &= -2 \times 2 \\
 \Leftrightarrow x - 5 &= -2 \text{ or } x - 1 = 2 \\
 \Leftrightarrow x &= 3
 \end{aligned}$$

Fortunately good nonsense of this kind is surprisingly hard to find.

3 Reasoning and School Examinations

My previous joint work, Sangwin and Köcher (2016), examined questions set in school-level examination papers and found that a third of the marks for school examinations were awarded for reasoning by equivalence. In Sangwin and Köcher (2016) our methodology was to take a corpus of published examination questions, together with the official mark scheme. We examined the extent to which we could automatically mark answers to these questions using the STACK software in a way which was faithful to the published mark scheme. In that research we selected the specimen questions on paper 1 and paper 2 for International Baccalaureate³ (IB) Mathematics Higher level, for first examinations in 2008. In this section I repeat the analysis, using different questions and with the benefit of a number of years of software development. The reasoning by equivalence engine now exists as a working prototype, and this research provides an opportunity to test my previous claims regarding the potential automation of assessment of students' equivalence reasoning.

The IB specification has now been superseded and so for this chapter I took the two Mathematics papers from the 2015 Scottish Higher Mathematics examinations. In Scotland school students typically take five Higher subjects aged 16–17, which form the basis of university entrance criteria. Universities outwith Scotland, e.g. in the rest of the UK, may require students to study to Advanced Higher level aged 17–18, so that the Highers do not perfectly align with the IB. Paper 1 is a non-calculator paper with a 70 min time duration, and with 60 marks available. Paper 2 permitted the use

³International Baccalaureate is a registered trademark of the International Baccalaureate Organization.

Table 2 The extent to which Higher Mathematics questions can be automatically assessed

	# marks	%
(i) Awarded by STACK <i>exactly</i>	47	36
(ii) Of which reasoning by equivalence	35	27

of a calculator, with a 90 min time duration, and with 70 marks available. Students sat both papers on the morning of Wed 20 May 2015. The examination papers, together with the official mark scheme, are publicly available from the Scottish Qualifications Authority website.

For the purposes of this chapter I repeated the following methodology from Sangwin and Köcher (2016). The IB and Highers examination systems are similar, but not identical. However the similarity and the specificity of the supplied mark scheme made it is straightforward to map the Higher marks onto the IB classifications, providing comparability with the previous work, see (Table 1).

My analysis of these questions sought to determine the extent to which answers could be automatically assessed by STACK exactly as in the mark scheme. I noted the extent to which this assessment included reasoning by equivalence as the method marks. The results are shown in Table 2. For the Highers, I implemented 27% of the marks as reasoning by equivalence. I previously suggested that for the IB papers 36% of the marks could be awarded. This discrepancy can be accounted for, since there is still reasoning by equivalence which is not yet accessible because calculus operations are not currently included. Typically, students either algebraically rearrange an expression before integrating/differentiating (e.g. see Highers paper 1 Q7, and paper 2, Q8b) or they perform calculus and rearrange the result, or form an equation. Inclusion of calculus operations in STACK would substantially increase the range of answers which could be fully assessed. These results, therefore, represent a lower bound on the extent to which school examinations could ultimately be automatically assessed.

As with the IB, the “reasoning” marks are an almost insignificant component of the examination marks. In the case of the Highers questions I decided that only 3 marks (from 130) were available for reasoning. Higher paper 1 Q3 asked students to use the remainder theorem to show $(x + 3)$ was a factor of $x^3 - 3x^2 - 10x + 24$; Higher paper 1 Q9 required students to establish if points were colinear; and paper 2 Q3b asked students to reason about the model based on derived inequalities. Most questions asked students to calculate, rather than reason based on calculations.

The results reported in this section support the hypothesis that the most important single form of reasoning in school mathematics is, and remains, algebraic reasoning by equivalence. In many cases students’ first, very simple, proofs are entirely algebraic. Even where additional reasoning is used, reasoning by equivalence is a central component of most current school-level proofs. In the next section I describe students’ traditional paper and pencil attempts at solving algebraic problems, before reporting on their attempts at reasoning by equivalence with the STACK software.

4 Students' Attempts at Algebraic Reasoning

Following from the research reported in Sangwin and Köcher (2016) and curious to pursue the potential of reasoning by equivalence as a starting point for assessment of complete mathematical arguments, albeit of the most elementary kind of proof, I started to consider how software might automatically assess the correctness of a student's algebraic derivation. To inform this design process I investigated students' attempts at algebraic reasoning. My prior experience as a teacher strongly suggests that students' written algebraic work typically contains no logical connectives or other justification and entirely ignores natural domain conventions. When solving an equation students appear to work line-by-line, but each line is apparently disconnected from the previous lines. There is some literature on this issue, e.g. Bernardo and Carmen (2009), Levenson (2012), Tirosh and Evan (1997), but none of this directly relates to university mathematics students. I therefore set out to investigate the following, working on paper in the traditional way.

1. To what extent do students use (and correctly use) logical connectives between lines of algebraic working? What other justification is evident other than "implied equivalence"?
2. To what extent do students acknowledge the natural domains of definition? For example in the expression $\frac{1}{x-4}$ the value $x = 4$ is excluded from the natural domain.
3. To what extent do students show evidence they have checked a particular answer is actually correct?

In Sangwin (2016) my experimental instrument to address the above research questions consisted of the following two algebraic problems.

$$\text{Q1. Please solve } \frac{x+5}{x-7} - 5 = \frac{4x-40}{13-x}. \quad (1)$$

$$\text{Q2. Please solve } \sqrt{3x+4} = 2 + \sqrt{x+2}. \quad (2)$$

These two questions were chosen because they require line by line reasoning but there are opportunities for applying rules outside the domain of applicability. For example, an erroneous solution to the first question, taken from (Northrop, 1945, p. 81), is shown in Fig. 2. The "rule" $\frac{a}{b} = \frac{a}{c} \Leftrightarrow b = c$ obscures the requirement that $a \neq 0$. The purpose of including this question was to see how many students cancel a term such as $4x - 40$, and evidence for how students dealt with the subsequent contradiction.

The typical algebraic solution to the second question, taken from Newman (1957, p. 8), involves squaring both sides and collecting terms to get $x - 1 = 2\sqrt{x+2}$ and squaring again to obtain a quadratic with real roots $x = 7$ or $x = -1$. However, there is only one real solution as shown in the graphical solution of Fig. 3. It is relatively straightforward to construct similar examples with no real solutions (e.g. $\sqrt{x+2} = 2 + \sqrt{3x+4}$), see Bonnycastle (1836, p. 88), Durell (1930, p. 46) or Lund (1852,

$$\begin{aligned} \frac{x+5}{x-7} - 5 &= \frac{4x-40}{13-x} \\ \frac{x+5-5(x-7)}{x-7} &= \frac{4x-40}{13-x} \\ \frac{4x-40}{7-x} &= \frac{4x-40}{13-x} \\ 7-x &= 13-x \\ 7 &= 13. \end{aligned}$$

Fig. 2 An erroneous solution to question (1)

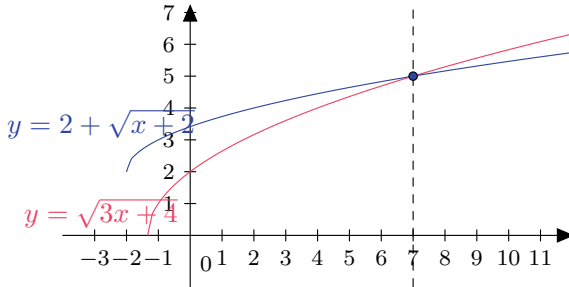


Fig. 3 A graphical solution to question (2)

p. 130). In some problems of this kind the natural domain can be used to eliminate one of the solutions, but $x = -1$ does not violate the domain constraints of $\sqrt{3x + 4}$ or $\sqrt{x + 2}$, i.e. $x \geq -4/3$. In this case it is the reversibility of squaring both sides of an equation which introduces a spurious solution, rather than domain constraints.

As reported in Sangwin (2016), students solved these two equations writing answers on paper. The cohort were a group of 175 students taking an engineering programme at a good United Kingdom university. The detailed methodology and results are reported in Sangwin (2016). For question 1, of the 113 correct responses, 53 (46.9%) cross multiplied, expanded out all brackets and solved the resulting equation correctly to get the unique answer $x = 10$. Therefore, these students missed the opportunity to cancel the term $4x - 40$ on both sides. In this question 25 (22%) students started by writing the left hand side as a rational expression. While 22 of these students had the opportunity to cancel a factor none of them did so. For question 1, only 14 (9.5%) of students showed any evidence of logical connectives between algebraic statements. Only 2 students wrote any evidence of having performed a check that their answer satisfied the original equation, and only 1 student explicitly considered domains of definition of the rational expression by excluding $x = 7$ and $x = 13$ from the domain of definition for the original equation. Overall, only 17 (11.6%) of students wrote any evidence of more than algebraic symbolic manipulation.

For question 2, the most popular answer consisted of squaring both sides, rearranging and squaring again before solving the resulting quadratic to get roots $x = 7$, $x = -1$. 85 students took this approach, of which 24 students also checked that

their answers satisfied the original equation, giving complete and correct solutions by only 16% of the cohort. For question 2, only 4 students showed any evidence of checking domains of definition and only a further 3 students used any logical connectives. The most common mistake was squaring a binomial incorrectly, e.g. $(\sqrt{a} + \sqrt{b})^2 = a + b$. In particular, 18 students wrote

$$\sqrt{3x+4} = 2 + \sqrt{x+2}, \quad 3x+4 = 4 + x + 2, \quad x = 1. \quad (3)$$

What was perhaps surprising was that even for comparatively elementary problems, students took on average 10 or 14 lines to achieve a correct solution to questions (1) and (2) respectively. This strongly suggests that online assessment systems, such as STACK, need to assess more than the final answer, particularly in a formative setting.

5 Developing Algebraic Reasoning by Equivalence in STACK

Based on the analysis of examination questions, and research associated with students' algebraic reasoning, I started to develop an extension to the STACK online assessment system to assess students' reasoning by equivalence. Initially I decided that the minimum worthwhile functionality should include (i) rearranging a simple equation to make a particular variable the subject, and (ii) solving linear and quadratic equations in a single real variable, over the real numbers. However, ultimately the first version included sufficient additional functionality to allow automatic assessment of the problems in questions (1) and (2).

In developing a reasoning by equivalence system I adopted the following design principles.

- P1 The system should be mathematically and logically correct.
- P2 Students should provide a complete line by line solution. A student's answer constitutes a single mathematical object, which is to be subject to verification.
- P3 The system should mirror current practice as closely as possible, within the constraints of a liberal typed linear syntax, see Sangwin and Ramsden (2007).

A distinctive aspect of this design is the focus on the whole argument as a single object which can be subject to formal verification. That is to say, the student's lines of working are assumed to be connected by equivalence symbols and this becomes the single mathematical object. Similarly, I decided that students should not need to explicitly add in the natural domain of expressions. However, in line with (P1), the system would indicate natural domain constraints automatically.

There is an important distinction between reasoning and argumentation.

Reasoning is [...] the line of thought adopted to produce assertions and reach conclusions. *Argumentation* is the substantiation, the part of the reasoning that aims at convincing oneself or someone else that the reasoning is appropriate. (Boesen, Lithner, & Palm, 2010, p. 92)

Therefore reasoning can be valid or invalid. Similarly, argumentation may or may not correctly justify a particular step. Teachers often use a phrase such as “right method, wrong reason” to describe various types of mistake. In line with (P3) I decided that, in the first version, students would only be required to provide their reasoning (i.e. the algebraic expressions) but *not* also supply the argumentation to justify what they have done or why.

The decision not to require students to provide details of (i) argumentation and (ii) natural domains significantly simplified the input mechanism. STACK already has a well-developed input mechanism for algebraic expressions, as described in Sangwin and Ramsden (2007) and later in Sangwin (2013). A distinctive feature of this interface is a separation of “validity” of input from “correctness” of an answer. Feedback relating to the validity of input includes syntactic problems such as missing brackets, and this feedback is intended to always be available even during an examination or other high-stakes situation. The provision of validity feedback mediates the difference between a typed expression (e.g. $1 / (1+x^2)$) and traditional two dimensional displayed forms (e.g. $\frac{1}{1+x^2}$), helping students to ensure the expression they typed matches what they intended. Rejecting expressions as “invalid” rather than “wrong” immediately helps prevent students from being penalised on a technicality. Separating validity from correctness has been found to be an essential feature of effective online assessment of mathematics.

The interface, implementing question 2, is shown in the left of Fig. 4. The student has typed a number of lines of working into the textarea on the left. STACK has automatically and instantaneously displayed their line by line working in the box to the right. In addition to the expressions typed by the student, STACK has inferred and indicated the natural domain as blue text to the right. The right hand side of Fig. 4 shows the outcome of the assessment. In this case, the line by line reasoning is not correct in three places. Two involve squaring both sides (and additionally a domain enlargement), whereas the last line omits a root from the previous line without justification. Here the system uses a \Leftarrow symbol to indicate the subset of one variety within another.

Notice that in Fig. 1 STACK added in \Leftrightarrow symbols into the validation feedback area to indicate lines are equivalent, whereas these have not been added automatically in the left of Fig. 4. In some situations a teacher may want to immediately indicate

Solve $\sqrt{3x+4} = 2 + \sqrt{x+2}$, working line by line. Leave your answer in fully simplified form.

Correct answer, well done.

$\sqrt{3x+4} = 2 + \sqrt{x+2}$	$x \in [-\frac{4}{3}, \infty)$
$3x+4 = 4 + 4\sqrt{x+2} + x+2$	$x \in [-2, \infty)$
$2x-2 = 4\sqrt{x+2}$	$x \in [-2, \infty)$
$x-1 = 2\sqrt{x+2}$	$x \in [-2, \infty)$
$x^2 - 2x + 1 = 4(x+2)$	
$x^2 - 2x + 1 = 4x + 8$	
$x^2 - 6x - 7 = 0$	
$(x-7)(x+1) = 0$	
$x = 7$ or $x = -1$	
$x = 7$	

The variables found in your answer were: x

$\sqrt{3x+4} = 2 + \sqrt{x+2}$	$x \in [-\frac{4}{3}, \infty)$
? $3x+4 = 4 + 4\sqrt{x+2} + x+2$	$x \in [-2, \infty)$
$\Leftrightarrow 2x-2 = 4\sqrt{x+2}$	$x \in [-2, \infty)$
$\Leftrightarrow x-1 = 2\sqrt{x+2}$	$x \in [-2, \infty)$
? $x^2 - 2x + 1 = 4(x+2)$	
$\Leftrightarrow x^2 - 2x + 1 = 4x + 8$	
$\Leftrightarrow x^2 - 6x - 7 = 0$	
$\Leftrightarrow (x-7)(x+1) = 0$	
$\Leftrightarrow x = 7$ or $x = -1$	
$\Leftarrow x = 7$	

Fig. 4 A student’s attempt at reasoning by equivalence in the STACK system

whether adjacent lines are equivalent. In other situations feedback on whether lines are equivalent constitutes part of what makes up “correctness” of the answer, and so the equivalence symbols are not displayed until the student considers they have a complete and correct solution worthy of assessment. The teacher is faced with a large number of choices about the nature and timing of feedback, and what should be part of the validity check, and what is core to the correctness of an answer.

The interface was developed by me, and question 2 was trialled on a year 1 calculus course at a good UK university. The course contains 568 students, academically comparable to the group who undertook the original paper and pencil trial (but at different institutions, and some three years later). In this situation students were permitted multiple attempts at the quiz, and Fig. 4 shows one student’s attempt. Over 477 students provided valid attempts to this question. Of these attempts 33 students only stated the question (valid, but wrong). A further 33 students stated the question and jumped to the final answer $x = 7$ (valid, and correct) and five students stated the question and gave $x = 1$ as the next line (valid, but wrong, and note (3)). If a teacher wants more than just the question and a final answer an additional property, such as the minimum number of lines, would need to be specified. There were 387 different valid responses, with only 15 being given more than once. What is interesting is the number of lines of working used in students’ final answers. Of the students who did more than state the question and answer, i.e. who provided three or more lines of working, the mean number of lines used was 8.0 with a standard deviation of 2.4 and only 12 students used more than 12 lines. In the online interface students used fewer lines than for the same problem on paper. The ability to copy, paste and delete input lines within the textarea makes it likely students tidied up their answer before final submission, reducing unnecessary lines.

The original goal was to implement minimum functionality to support (i) rearranging a simple equation and (ii) solving linear and quadratic equations. In order to achieve this functionality the following issues needed to be addressed.

- There is general ambiguity about how to express multiplicity of roots. If $(x - 1)^2 = 0$ is not equivalent to $x = 1$ then students need to indicate multiplicity of roots, but I am aware of no consensus on how this should be notated.
- Students need to enter sets of real numbers, sometimes as a set, sometimes using interval notation and at other times via inequalities.

My approach to multiplicity of roots is illustrated in Fig. 5. The equation $(x - 3)^2 = 0$ and the expression “ $x = 3$ or $x = 3$ ” are considered to be equivalent, because they

$(x-3)^2=0$ $x=3$ or $x=3$ $x=3$	<div style="text-align: center;"> $(x - 3)^2 = 0$ \Leftrightarrow (Same roots) </div> $x = 3$ or $x = 3$ $x = 3$
The variables found in your answer were: x	

Fig. 5 Dealing with repeated roots

Fig. 6 Support for assignment of a value to an expression within an argument

have the same roots with the same multiplicity. The expressions “ $x = 3$ or $x = 3$ ” and “ $x = 3$ ” have the same variety, but are not identical. This is, of course, slightly awkward since logical “or” is idempotent, and so “ $x = 3$ or $x = 3$ ” and “ $x = 3$ ” would be equivalent at a symbolic level. For this reason, STACK accepts $x = 3$ as equivalent to $(x - 3)^2 = 0$, but with an acknowledgement.

This leads us onto the issue of entry of sets of real numbers. If reasoning by equivalence is merely choosing a list of representatives of an equivalence class, then any of these representatives describes the set of real numbers! In an educational context, teachers normally look for more normative ways of writing this set, such as $x = 1$. After much thought, and experimentation, I decided to *require* forms such as “ $x = 1$ or $x = 2$ ” rather than the alternatives “ $x = 1, 2$ ” or “ $x = 1$ or 2”. The statement “ $x = 1$ and $x = 2$ ” represents no real numbers (i.e. the empty set) and so is normally simply wrong, although the sentence “The roots are 1 and 2.” is perfectly correct and is in no way ambiguous. Currently I do not support entry of expressions like $x \in \{1, 2\}$ although this will be added in a future version, as will be support for interval notation.

What became clear during the automation of questions from the Scottish Highers Examinations, was that the addition of a small number of additional mathematical moves, combined with reasoning by equivalence, would significantly expand the range of mathematical questions which can be completely assessed automatically. In particular (i) a “let” assigning a value to an expression, (ii) support for equating coefficients, (iii) support for basic symbolic calculus operations.

As an example of the “let” statement consider the response shown Fig. 6, taken from Scottish Highers (2015), paper 2, Q4a. This does not, strictly speaking, maintain an equivalence but it proves very useful indeed if the system will allow evaluation of an expression, followed by subsequent algebraic manipulation. Beyond the scope of this chapter is discussion of support for basic symbolic calculus operations, which the STACK system already supports extensive functionality to implement. Suffice to say, that differentiating an expression, equating to zero and solving the subsequent equation is a very common task at this level.

Designers of other tutor software have taken very different decisions. For example, Mathpert is a stand-alone desktop system which allows its user to solve mathematical problems by constructing a step-by-step solution.

Mathpert is intended to replace paper-and-pencil homework in algebra, trig, and calculus, retaining compatibility with the existing curriculum while at the same time supporting innovative curriculum changes; to provide easy-to-use computer graphics for classroom demonstration in those subjects, as well as for home study; to replace or supplement chalk-and-blackboard in the classroom for symbolic problems as well as graphs. (Beeson, 1998)

Students either pick a built-in topic or enter the problem they wish to solve. Students then use the *calculation window* to solve the problem in a step-by-step fashion. To do this, users select part (or all) of an expression and the system responds with a menu of operations which can be performed on that selection. The software performs the selected operation automatically. Having written my own software, I full endorse its fundamental approach to the mathematical underpinning.

... if we start with an *educational* purpose, [...] it is impossible to achieve ideal results by tacking on some additional “interface” features to a previously existing computational system. To put it another way: it is not possible to entirely separate “interface” considerations from “kernel” considerations. (Beeson, 1998)

Other important works in this area are Heeren et al. (2010) and Prank (2011). In particular, Aplusix focuses on reasoning by equivalence, see Nicaud et al. (2004). One difficulty, highlighted by Nicaud et al. (2004), with the menu-driven application of rules occurs when a single rule is applicable to different sub-expressions. For example, in the expression $x^4 - x^2 - 9$ the rule $a^2 - b^2 \rightarrow (a - b)(a + b)$ can be applied in two different ways, either matching to $x^4 - x^2$ or to $x^4 - 9$. In this situation it is difficult to select the sub-expression $x^4 - 9$ from $x^4 - x^2 - 9$ without re-ordering some of the terms. More detailed discussion of these issues is given by Beeson (1998, 2004).

6 Discussion

This chapter opened by considering the four “patterns of thought” identified by Polya (1962), namely (i) “*the pattern of two loci*”, (ii) “*superposition*”, (iii) “*recursion*” and (iv) the “*Cartesian*” pattern of thought. The Cartesian pattern of problem solving involves *setting up an equation*, then solving it and interpreting any solution. These patterns relate to how to go about solving problems. However, more than half (21/40) of the separate question parts in the examination questions of Sect. 3 do not relate to a problem at all. Rather they instruct students to undertake a well-rehearsed set of techniques, isolated from any problem. For example, highers paper 1 Q6 asks students to evaluate $\log_6(12) + \frac{1}{3} \log_6(27)$. Technique alone does not correspond to the Cartesian pattern of thought. All but one of the other question parts asked students to set up and solve equations. One question part does relate to recursion, expecting students to recognise the limit of a recurrence relation would result in the fixed point

of an equation. Burkhardt (1987), and others, have argued that in a very real sense to students, the subject of mathematics is defined by what we, as a mathematics community, expect students to do in examinations. Currently, school level problem solving and proof is confined to algebraic and calculus methods in which the primary technique is reasoning by equivalence.

Reasoning by equivalence continues to play a central role in mathematics beyond school examinations, and into undergraduate work. As already mentioned, it forms much of the work in symbolic calculus problems. Further, with many proofs reasoning by equivalence is central. For example, in induction to prove results such as $\sum_{k=1}^n k^2 = \frac{1}{6}n(n+1)(2n+1)$ the central work in the induction step is reasoning by equivalence. Therefore all technology designed to support proof in teaching should support reasoning by equivalence as a central component.

In this work I have focused on equivalence, rather than on re-write rules. There is an extensive literature on re-write rules, e.g. Bundy (1983), and these are commonly taught in school mathematics. The Mathpert system of Beeson (1998) avoids the problem of deciding if a student's step is legitimate by providing a menu of steps from which the student must choose. It is certainly possible to infer some of the steps a student has made, and so some analysis of students' steps is a valuable addition to the reasoning by equivalence described here. However, steps alone will not be sufficient. Indeed, one goal of developing fluency in algebra is to become proficient at combining small individual steps into one. Taken to an extreme, a student might well move from the equation $x^2 + 2x + 1 = 0$ directly to the solution $x = -1$. These equations are equivalent, but whether this single step is acceptable will depend on the teacher and context. The teacher might additionally require other properties in a complete solution. For example, they may require a minimum number of lines of working, or that a particular form (e.g. factored) appears as one line of working.

Perhaps the most significant drawback of focusing on equivalence is the problem that addition, removal and permutation of equivalent intermediate steps will all be accepted by a pure equivalence reasoning engine. Firstly I should note that students do, in fact, sometime backtrack or write unnecessary steps. While this might well be sub-optimal or less aesthetically pleasing this does not make their work "wrong". Further work with live students will be needed to decide the extent to which this is actually a problem, and to design mechanisms for avoiding the problems which arise. Ultimately in addition to equivalence the teacher will seek to establish a range of other properties which may include inferring which steps were taken where this is possible, and the relationship (if any) to a teacher's envisaged model answer.

Current dynamic geometry systems provide powerful tools for experimentation. However, none of the current systems of which I'm aware allow students to move beyond experimentation to write up a more formal claim and proof. Allowing students to write simple proofs based on Polya's "*pattern of two loci*" appears to be a sensible starting point, and combining algebra and geometry another.

There are more profound differences at the level of logic. Most automatic theorem proving software makes use of the resolution rule of inference which is closely related to the contrapositive, Bundy (2013). To use existing automatic theorem proving software a more profound shift in how we teach logic is needed.

7 Conclusion

This chapter takes the hypotheses that students continue to need to learn how to perform reasoning by equivalence accurately, regardless of the technology available to them. More specifically, the hypothesis is that all students will need to encounter some reasoning by equivalence as a basic part of algebra: teaching how to solve linear and quadratic equations is unlikely to disappear from curricula in the future. In the version of STACK described here, students are expected to perform the computations themselves and type in their answer, hence the need for flexibility of what is and is not a step. In contrast, when using MathXpert students chose what to do, but were not expected to do it. Not saying what you are doing is defensible when only reasoning by equivalence. However, as soon as other operations are included (e.g. let, equate coefficients, and calculus) then this position becomes unsustainable. One solution is what Back, Mannila, and Wallin (2010) called *structured derivations*. Another very early approach, illustrated in Brancker, Pell, and Rahn (1668), is the three column proof in which the argumentation and reasoning are clearly laid out with reference to numbered lines.

We do have an opportunity, through carefully designed tools, to communicate to students what is and is not important in writing an acceptable mathematical proof. If logical symbols such as \Leftrightarrow matter, then either the system should include them automatically or students should be constrained to write them. Templates provide another alternative in creating constraints in which people work. Constraints can be liberating: they remove the need to worry about whether the overall *form* of the proof is acceptable, leaving the proof author instead to focus on the details. Designing software to automate a process is another, very demanding, type of constraint. I think the attempt to automate assessment of students' proofs is a valuable way of understanding the nature of elementary mathematics, and the challenges associated with teaching and learning mathematics.

References

- Adams, W. W., & Loustaunau, P. (1994). *An introduction to Grobner bases* (Vol. 3). Providence, Rhode Island: American Mathematical Society.
- Back, R. J., Mannila, L., & Wallin, S. (2010). It takes me longer, but I understand better'—Student feedback on structured derivations. *International Journal of Mathematical Education in Science and Technology*, 41(5), 575–593. <https://doi.org/10.1080/00207391003605221>.
- Beeson, M. (1998). Design principles of Mathpert: Software to support education in algebra and calculus. In N. Kajler (Ed.), *Computer-human interaction in symbolic computation* (pp. 89–115). Vienna, Austria: Springer. <https://doi.org/10.1007/978-3-7091-6461-7>.
- Beeson, M. (2004). The mechanization of mathematics. In C. Teuscher (Ed.), *Alan Turing: Life and legacy of a great thinker* (pp. 77–134). Berlin, Germany: Springer. <https://doi.org/10.1007/978-3-662-05642-4>.
- Bernardo, G., & Carmen, B. (2009). The ambiguity of the sign $\sqrt{\quad}$. In *Proceedings of CERME6*, Lyon, France, Working Group 4 (pp. 509–518).

- Boesen, J., Lithner, J., & Palm, T. (2010). The relation between types of assessment tasks and the mathematical reasoning students use. *Educational Studies in Mathematics*, 75(1), 89–105. <https://doi.org/10.1007/s10649-010-9242-9>.
- Bonnycastle, J. F. (1836). *An introduction to algebra* (16th ed.). London, UK: Longman.
- Boole, G. (1847). *The mathematical analysis of logic, being an essay towards a calculus of deductive reasoning*. Cambridge, UK: MacMillan, Barclay, & Macmillan.
- Brancker, T., Pell, J., & Rahn, J. H. (1668). *An introduction to algebra*. London, UK: Printed by W.G. for Moses Pitt.
- Bundy, A. (1983). *The computer modelling of mathematical reasoning*. London, UK: Academic Press.
- Bundy, A. (2013). The interaction of representation and reasoning. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 469(2157). <https://doi.org/10.1098/rspa.2013.0194>.
- Burkhardt, H. (1987). What you test is what you get. In I. Wirszup & R. Streit (Eds.), *The dynamics of curriculum change in developments in school mathematics worldwide*. University of Chicago School Mathematics Project.
- Durell, C. V. (1930). *New algebra for schools* (3 Vols.). London, UK: Bell & Sons.
- Euler, L. (1822). *Elements of algebra* (3rd ed.). London, UK: Longman, Hurst, Rees, Orme and Co. (Translated from the French, with the notes of M. Bernoulli and the Additions of M. de La Grange by J. Hewlett).
- Heeren, B., Jeuring, J., & Gerdes, A. (2010). Specifying rewrite strategies for interactive exercises. *Mathematics in Computer Science*, 3(3), 349–370. <https://doi.org/10.1007/s11786-010-0027-4>.
- Kirshner, D., & Awtry, T. (2004, July). Visual salience of algebraic transformations. *Journal for Research in Mathematics Education*, 35(4), 224–257. <https://doi.org/10.2307/30034809>.
- Leibniz, G. (1966). *Logical papers: A selection*. Oxford, UK: Oxford University Press.
- Levenson, E. (2012, June). Teachers' knowledge of the nature of definitions: The case of the zero exponent. *The Journal of Mathematical Behavior*, 31(2), 209–219. <https://doi.org/10.1016/j.jmathb.2011.12.006>.
- Lund, T. (1852). *The elements of algebra designed for the use of students in the university* (14th ed.). London, UK: Longman, Brown, Green and Longmans.
- Maxwell, E. A. (1959). *Fallacies in mathematics*. Cambridge, UK: Cambridge University Press.
- Newman, M. H. A., et al. (1957). *The teaching of algebra in sixth forms: A report prepared for the Mathematical Association*. London, UK: G. Bell and Sons Ltd.
- Nicaud, J. F., Bouhineau, D., & Chaachoua, H. (2004). Mixing microworlds and CAS features in building computer systems that help students learn algebra. *International Journal of Computers for Mathematical Learning*, 9(2), 169–211. <https://doi.org/10.1023/B:IJCO.0000040890.20374.37>.
- Northrop, E. P. (1945). *Riddles in mathematics: A book of paradoxes*. London, UK: The English Universities Press.
- Polya, G. (1962). *Mathematical discovery: On understanding, learning, and teaching problem solving*. London, UK: Wiley.
- Prank, R. (2011). What toolbox is necessary for building exercise environments for algebraic transformations. *The Electronic Journal of Mathematics and Technology*, 5(3).
- Sangwin, C. J. (2013). *Computer aided assessment of mathematics*. Oxford, UK: Oxford University Press.
- Sangwin, C. J. (2015, July). An audited elementary algebra. *The Mathematical Gazette*, 99(545), 290–297. <https://doi.org/10.1017/mag.2015.37>.
- Sangwin, C. J. (2016). Undergraduates' attempts at reasoning by equivalence in elementary algebra. In *Didactics of Mathematics in Higher Education as a Scientific Discipline: Conference Proceedings* (pp. 335–341). Universität Kassel, Leuphana Universität Lneburg, Universität Paderborn.
- Sangwin, C. J., & Köcher, N. (2016). Automation of mathematics examinations. *Computers and Education*, 94, 215–227. <https://doi.org/10.1016/j.compedu.2015.11.014>.

- Sangwin, C. J., & Ramsden, P. (2007). Linear syntax for communicating elementary mathematics. *Journal of Symbolic Computation*, 42(9), 902–934. <https://doi.org/10.1016/j.jsc.2007.07.002>.
- Tirosh, D., & Evan, R. (1997). To define or not to define: The case of $(-8)^{\frac{1}{3}}$. *Educational Studies in Mathematics*, 33(3), 321–330. <https://doi.org/10.1023/A:100291660>.

Virtual Manipulatives and Students' Counterexamples During Proving



Kotaro Komatsu  and Keith Jones 

1 Introduction

Counterexamples play a vital role in the development of mathematics (Lakatos, 1976), and the learning of counterexamples has many benefits in school mathematics. For example, students can experience the reinvention of mathematics through producing and addressing counterexamples (Larsen & Zandieh, 2008), and such activity could result in students obtaining deeper and more authentic insights into the nature of mathematics (de Villiers, 2010). Counterexamples not only stimulate such mathematical practice, but also enhance the learning of mathematical content, including concept formation and definitions (de Villiers, 1998).

However, several studies (e.g. Hoyles & Küchemann, 2002; Ko & Knuth, 2013; Zaslavsky & Peled, 1996) have reported that students encounter difficulties in producing appropriate counterexamples. In addition, some students cannot properly address counterexamples that they discover themselves or that others, such as classmates and teachers, present to them (Balacheff, 1991). There are at least two options for coping with counterexamples to conjectures: restricting the domains of the conjectures to exclude the counterexamples, or modifying the conjectures so that the counterexamples can be included as examples of the modified conjectures. However, some students do not perform these actions, but rather ignore counterexamples and consider that conjectures remain true despite the existence of counterexamples (Balacheff, 1991).

K. Komatsu (✉)

Institute of Education, Shinshu University, Nagano, Japan

e-mail: kkomatsu@shinshu-u.ac.jp

K. Jones

School of Education, University of Southampton, Southampton, UK

e-mail: d.k.jones@soton.ac.uk

© Springer Nature Switzerland AG 2019

G. Hanna et al. (eds.), *Proof Technology in Mathematics Research and Teaching*, Mathematics Education in the Digital Era 14, https://doi.org/10.1007/978-3-030-28483-1_16

331

Intervention studies focusing on counterexamples would be necessary for addressing these students' difficulties, but such studies are scarce in mathematics education research (Stylianides, Stylianides, & Weber, 2017). We initially focused on this issue by examining task design and teachers' roles for enhancing students' mathematical activity involving proofs and refutations in paper-and-pencil environments (Komatsu, 2017; Komatsu, Tsujiyama, Sakamaki, & Koike, 2014). Given the potential for positive impacts of computer technology on proof and proving (e.g. Arzarello, Bartolini Bussi, Leung, Mariotti, & Stevenson, 2011), we are extending our studies by taking the availability of dynamic geometry environments (DGEs) into consideration (Komatsu & Jones, 2019). In the study with DGEs, we developed theory-informed task design principles, designed sets of tasks based on these principles, and examined the affordances of the tasks by implementing task-based interviews with secondary school students and undergraduate students.

While we developed the tasks according to a set of theory-informed design principles derived from existing studies, our purpose here is to re-analyse these tasks, and illustrate the implemented task-based interviews, using a different framework. Here, a *different* framework for task design means a framework that was not used in the original theory-informed task design. The benefits of networking different theoretical approaches have been deliberated by Prediger, Bikner-Ahsbals and Arzarello (2008, p. 172), who advocate "looking at the same phenomenon from different theoretical perspectives as a method for deepening insights on the phenomenon". In our research, the empirical phenomena are the designed tasks and implemented task-based interviews. Our premise is that if a certain task design is examined through a framework that was not used in the original task design, this deepens insight into the theoretical basis of the task design and adds support for the task design.

To that purpose, we employ Osana and Duponsel's (2016) framework for research on *virtual manipulatives*. We were not aware of their framework when initially developing our task design principles (Komatsu & Jones, 2019). After designing the tasks and implementing the task-based interviews, we became aware of their work and came to realise that their framework can be employed for re-examining our task design.

A virtual manipulative is defined as "an interactive, technology-enabled visual representation of a dynamic mathematical object, including all of the programmable features that allow it to be manipulated, that presents opportunities for constructing mathematical knowledge" (Moyer-Packenham & Bolyard, 2016, p. 13). Some representations constructed in DGEs can be considered as virtual manipulatives. Geometric figures as theoretical objects can be represented with DGE diagrams that are constructed so as to be 'robust'; that is, the diagram has the necessary and sufficient geometrical properties that hold when any point or line is 'dragged' (Healy, 2000; Jones, 2000; Laborde, 2005). With such 'robust' diagrams, it is possible to observe the dynamic transformation of the diagrams in an interactive way by manipulating them through dragging on-screen points and lines. The diagrams' geometrical properties are maintained during this dragging, and thus DGEs can provide an opportunity to gain knowledge about the invariant properties of the geometric figures (for another example of DGEs as virtual manipulatives, see Manizade & Martinovic, 2016).

By reviewing existing studies of concrete and virtual manipulatives, Osana and Duponsel constructed a framework that consists of three elements: the *surface features of the representations*, the *pedagogical support*, and the *students' perceptions and interpretations*. They argue that the framework can be employed for task design. While, for Osana and Duponsel, the focus of their framework is on the learning of mathematical concepts, manipulatives are used not only for enhancing the learning of mathematical concepts and procedures but also for facilitating students' success in mathematical practices such as proving (an example being the 'action proofs' in Semadeni, 1984). Hence, we anticipate that Osana and Duponsel's framework can be applied to task design for proof-related activity, one of the themes of this book. The purpose of this chapter is to re-analyse our designed tasks and an implemented task-based interview using Osana and Duponsel's framework for research on virtual manipulatives.

2 Analysing Task Design for Producing and Addressing Counterexamples Using a Framework on Virtual Manipulatives

Table 1 shows the set of task design principles that we have elaborated for helping students produce and address counterexamples. Figure 1 provides one example of a

Table 1 Task design principles for producing and addressing counterexamples

Principle	Description of the principle
Principle 1	Using statements whose conditions are purposefully implicit and thus allow the production of particular proofs and the occurrence of counterexamples
Principle 2	Providing tools that enhance the production of counterexamples, while making explicit the purpose of the tools' use (i.e. investigating the existence of counterexamples)
Principle 3	Increasing students' recognition of contradictions that can help them revise conjectures/statements and/or proofs

<p>Task 1. In parallelogram ABCD, we draw perpendicular lines AE and CF to diagonal BD from points A and C, respectively. Prove that quadrilateral AECF is a parallelogram.</p>	
<p>Task 2. Construct the diagram shown in Task 1 using a DGE. Move the vertices to change the shape of parallelogram ABCD, and examine whether quadrilateral AECF is always a parallelogram.</p>	

Fig. 1 A set of designed tasks

Table 2 Employment of the principles for the task design

Principle	Task design
Principle 1	Task 1 is based on this principle. The statement includes a hidden condition in the given diagram assuming that perpendicular lines from points A and C always intersect with diagonal BD. A proof of the statement can be constructed for the given diagram. However, if the shape of parallelogram ABCD is changed, cases where quadrilateral AECF is not constructed can be discovered
Principle 2	This principle was used for designing Task 2. Students are asked to use a DGE to construct the diagram given in Task 1, to transform it as required by dragging, and to investigate the truth of the statement
Principle 3	This principle was used for sequencing Tasks 1 and 2. In this sequence, students are asked to construct a proof in Task 1 and then transform the diagram with the DGE in Task 2. The expectation is that proof construction in Task 1 could increase students' conviction of the truth of the statement, and that thereby, a contradiction could likely be evoked more sharply between the students' conviction and its subsequent refutation in Task 2

set of designed tasks, and Table 2 explains how the design principles were employed for designing these tasks. Our intention of this chapter is not to address the design principles shown in Table 1 themselves because we have already discussed their theoretical basis elsewhere; see Komatsu & Jones (2019). Rather, our purpose here is to analyse the designed tasks shown in Fig. 1 in terms of the three elements of Osana and Duponsel's framework.

We consider each of the three elements of Osana and Duponsel's framework in turn, beginning with the surface features of the representations.

2.1 Surface Features of Representations

Mathematical concepts are abstract entities that cannot be directly observed, and manipulatives are used for making the abstract concepts concrete and visible, and thereby accessible for students. Thus, the benefit of manipulatives for mathematical learning cannot be judged only from the manipulatives themselves. Rather, as Osana and Duponsel (2016, p. 97) say, "the use of manipulatives is beneficial to the extent that students make clear associations between the objects themselves and the mathematical concepts they are intended to represent".

There can be various different manipulatives for one mathematical concept, and their representational features can vary on many dimensions, such as the extent to which they resemble the concept they are intended to represent. Based on a thorough review of the literature on the use of manipulatives, Osana and Duponsel (2016) argue that if manipulatives include surface features irrelevant to the underlying ideas the manipulatives target (e.g. attractive and perceptually-rich manipulatives), these manipulatives can distract students' attention and hinder their capacity to see beyond the manipulatives to the targeted ideas. In other words, Osana and Duponsel's frame-

work highlights the importance of task design with careful consideration of the representational features of virtual manipulatives so that the manipulatives better help students achieve the intended learning goals.

The statement in Task 1 (in Fig. 1) is relevant to the consideration of the surface features of representations. Our aim in the task design is to engage students in discovering and addressing counterexamples, and for that purpose, it is first of all necessary to use statements where counterexamples can be produced. A surface feature of the representation in Task 1 is in the given diagram with a particular characteristic: this diagram includes a hidden condition of assuming that perpendicular lines from points A and C always intersect with diagonal BD. While quadrilateral AECF is actually a parallelogram in the given diagram, cases where quadrilateral AECF does not appear can be found if parallelogram ABCD is transformed (as shown later in this chapter).

In proof tasks in school geometry, it is common that particular diagrams (usually one diagram per task) are given and the statements are described with reference to those diagrams (Herbst & Brach, 2006). If the described statements are general statements (such as for-all statements), students have to consider, on the one hand, the statements and proofs while taking into account not only the given diagrams but also certain general domains to which the diagrams belong. On the other hand, one diagram inevitably has to be one particular case and thus may suffer from “one-case concreteness” (Presmeg, 1986; Yerushalmy & Chazan, 1990): one diagram cannot represent the generality of the geometrical concept and may include properties that do not pertain to the concept. Hence, it is likely in some tasks (though not all) that statements based on the given diagrams are true only for subsets of the general domains. The statement in Task 1 has this representational feature—the particularity of the given diagram makes the condition of the statement implicit and thus counterexamples can occur in the subsequent Task 2 (for other tasks having this feature, see Komatsu, 2017).

This representational feature can be reinforced with DGEs because dynamic transformation in DGEs can represent, to some extent, the generality of geometrical concepts and thereby overcome the particularity of one diagram. Through performing dragging in DGEs, students not only can see examples but also may happen upon non-examples and counterexamples (Healy & Hoyles, 2001; Marradez & Gutiérrez, 2000).

Before continuing the examination of our task design, we would like to clarify the meaning of *counterexample* in the context of our study. We use this term subjectively, in that we judge a case to be a counterexample if students consider that a conjecture/statement is refuted by the case, regardless of whether it is really a counterexample in a logical sense. For example, Task 1 assumes, though implicitly, that quadrilateral AECF can be constructed, and under this assumption, the statement infers that this quadrilateral is a parallelogram. Hence, the cases shown later in this chapter are, mathematically, non-examples rather than counter-examples because they do not satisfy this assumption. However, as in Task 1, if statement conditions are not clear, it can be a subtle matter to discern whether a case is a counter-example or a non-example. Thus, we use the term counterexample from a subjective, student standpoint, not from a purely mathematical standpoint.

We now turn to consider the second element of Osana and Duponsel's framework, that of pedagogical support.

2.2 Pedagogical Support About the Use of Tools

According to Osana and Duponsel's framework, the second aspect crucial for task design with virtual manipulatives concerns the provision of adequate pedagogical support. As mentioned above, the benefit of using manipulatives rests on whether students can see meaningful connections between the manipulatives and their target concepts, but students may not construct these connections by themselves. Research on manipulatives shows that students' performance is enhanced when they receive explicit explanations that connect manipulatives and their conceptual referents.

However, some researchers warn against providing students with too much instruction about how to use manipulatives. With reference to Gravemeijer (2002), Osana and Duponsel (2016) state that "when students are given prescriptions for how to use manipulatives, their use of the objects can become highly mechanical, which is not conducive to mathematical understanding" (p. 106). Thus, in task design with virtual manipulatives, while certain instruction about the connections between the manipulatives and their intended goals is necessary, students should be allowed to make explorations with the manipulatives so that they can find their way to the intended goals on their own.

Task 2 is relevant to this level of pedagogical support. In this case, a virtual manipulative is an on-screen object created by students with a DGE, and the task asks students to investigate whether the statement in Task 1 is always true. With respect to DGEs, mathematics education researchers have illustrated several types of dragging modalities and measuring modalities (e.g. Arzarello, Olivero, Paola, & Robutti, 2002; Olivero & Robutti, 2007). On the one hand, this shows the power of DGEs to enhance various proof-related activities such as making conjectures and getting ideas for proof construction. On the other hand, different dragging modalities may lead to different activities. While one of the dragging modalities, *dragging test*, is specifically relevant to the discovery of counterexamples, other dragging modalities (e.g. *wandering dragging*, which is performed casually, without any goal) may not be helpful for that purpose. To focus students' attention on the intended activity, pedagogical support about the *purpose* of the use of a DGE is included in Task 2: "Move the vertices to change the shape of parallelogram ABCD, and *examine whether quadrilateral AECF is always a parallelogram*" (Fig. 1), rather than simply "use a DGE".

Task 2 implicitly has another goal of triggering students' additional mathematical activity—extending the statement in Task 1 to cope with counterexamples. However, this activity is not implied in the wording of Task 2 because of the expectation that students would spontaneously engage in this activity if they are stimulated in a certain way, which is described in the next section. In summary, a certain amount of pedagogical support is included in Task 2 in that while students are directly asked to

investigate the existence of counterexamples with a DGE, they are expected to act on their own initiative after the discovery of counterexamples.

The third element of Osana and Duponsel's framework is that of students' perceptions and interpretations.

2.3 Students' Perceptions and Interpretations

The third element of Osana and Duponsel's framework suggests that in task design with virtual manipulatives, how students might interpret the manipulatives should be taken into account. As mentioned above, the success of mathematical learning with manipulatives rests on how students perceive and interpret the manipulatives. However, students may interpret manipulatives in various ways, some of which may be different than expected.

In our task design, possible interpretations by students were considered in the sequencing of Tasks 1 and 2. It is entirely possible to ask students to construct the given diagram with a DGE and explore various diagrams by dragging before proof construction. In these explorations, they may come upon counterexamples to the statement. However, without proving the statement, they may not be convinced of its truth and, consequently, may not be inclined to improve the statement for addressing the counterexamples. Indeed, research indicates that when students (and teachers) encounter cases resulting in counterexamples, some may merely reject the cases, rather than interpreting them as genuine counterexamples (Balacheff, 1991; Zazkis & Chernoff, 2008).

To address this issue, we have capitalised on the potential of 'confusion', in particular, the benefit of contradictions that students face between their expectations and the results they obtain (D'Mello, Lehman, Pekrun, & Graesser, 2014; Prusak, Hershkowitz, & Schwarz, 2012). As explained in Table 2, Tasks 1 and 2 are sequenced such that proof construction in Task 1 could increase students' conviction of the truth of the statement, and that thereby, a contradiction could likely be evoked between their conviction and the subsequent counterexamples in Task 2; this contradiction could play a role as a catalyst for triggering students to improve the statement.

In the next section, we illustrate the affordances of our task design with a task-based interview that we conducted using the task shown in Fig. 1.

3 A Task-Based Interview

The tasks in Fig. 1 were implemented separately with two pairs of undergraduate students at a Japanese university. The purpose of the task-based interviews was to investigate how the tasks enabled undergraduates to engage in discovering and addressing counterexamples. The first author of this chapter conducted the task-based interviews. The role of the interviewer was simply to give the tasks and observe

the undergraduates' behaviour, and the undergraduates worked on the tasks without significant intervention from the interviewer.

Although the tasks are related to secondary school geometry, undergraduates were chosen as participants because proof construction in Task 1 might be challenging for secondary school students. All of the participants were prospective teachers training to teach at the primary and secondary school levels. The undergraduates were selected based on our anticipation that they would likely provide expansive verbal utterances useful for data analysis. The DGE *GeoGebra* was used in the task-based interviews, and the participants already had considerable experience in using this DGE.

Each task-based interview lasted for approximately 35 min. Two cameras were used for recording each case, one placed to film the undergraduates and the other to record the computer screen. The data we used for analysis were the video recordings, the transcriptions made from the recordings, the worksheets the undergraduates completed, and the DGE files they made. The analysis was conducted with a specific focus on what kind of diagram the undergraduates produced, whether they accepted the diagrams as counterexamples, and whether and how they revised the statement and their proofs (for more details about the data, see Komatsu & Jones, 2019).

Because the two pairs of undergraduates tackled the tasks in almost identical ways, this chapter focuses on one pair (Kenta and Tsubasa; pseudonyms) to avoid repetition. In the task-based interview, we prepared two worksheets, one which included only Task 1 and another one which included only Task 2. The undergraduates began by working on the first worksheet, and then received the second worksheet after they had solved Task 1 in the first worksheet.

3.1 The Proof of the Original Statement and the Types of Produced Diagrams

Because our focus in this chapter is on the undergraduates' encounters with counterexamples that occurred using DGE technology after proof construction, here we describe only briefly how the undergraduates constructed their proof in answer to Task 1.

The two undergraduates, Kenta and Tsubasa, began by planning to show that the diagonals of quadrilateral AECF intersected at their midpoints. Due to its difficulty, however, they abandoned this plan, and then saw another possibility—proving that $\triangle ABE \equiv \triangle CDF$ and thereby showing that a pair of opposite sides of quadrilateral AECF were equal and parallel. Below is the summary of the undergraduates' proof:

$\angle AEB = \angle CFD = 90^\circ$ and $AB = CD$ (the suppositions).

Since $AB \parallel DC$ (the supposition), $\angle ABE = \angle CDF$ (alternate angles).

Hence, $\triangle ABE \equiv \triangle CDF$ (a congruence condition for right-angled triangles), and thus $AE = CF$.

Since $\angle AEF = \angle CFE$ (the supposition), $AE \parallel CF$.

Hence, quadrilateral AECF is a parallelogram (a condition for parallelograms).

The undergraduates moved on to Task 2, where they used the DGE to construct the diagram shown in Task 1 following the conditions of the statement. Here, they selected the perpendicular line tool to draw perpendicular lines that passed through points A and C and intersected with diagonal BD, and defined points E and F as the intersection points of these perpendicular lines and diagonal BD. When doing so, they drew segments AE and CF, and made the perpendicular lines AE and CF not displayed.

After that, the undergraduates performed dragging to produce various types of diagrams (Fig. 2). They first considered the case where the positions of points E and F were reversed (Fig. 2a).¹ They then discovered the case where points E and F (and therefore quadrilateral AECF) disappeared as the perpendicular lines from points A and C did not intersect with diagonal BD (Fig. 2b). When exploring this case, they observed a border case between Fig. 1 and Fig. 2b where points E and F coincided with points B and D, respectively (Fig. 2c). The case where points E and F coincided was also found (Fig. 2d).

The undergraduates thought that the statement in Task 1 was true for Fig. 2a, c. With respect to Fig. 2d, they said, "In this case, there is no quadrilateral [AECF] itself, so difficult". Tsubasa mentioned that rhombuses produced Fig. 2d, including squares as a subtype of rhombus, based on his understanding of the hierarchical classification of quadrilaterals. In what follows, we focus on the undergraduates' engagement with the case of Fig. 2b because this involved the undergraduates dealing with a counterexample.

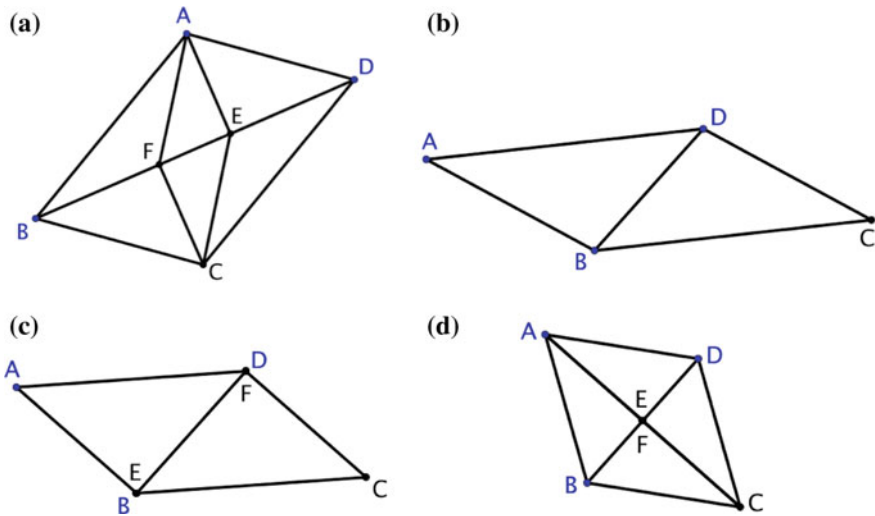


Fig. 2 The types of diagrams produced by the undergraduates in Task 2

¹It is more exact to state 'the type illustrated in Figure X', not just 'Figure X', because the undergraduates considered more than one diagram for one type by dragging. In this chapter, we simply state 'Figure X' for readability.

3.2 *The Discovery of a Counterexample and the Extension of the Statement*

Below is the undergraduates' exchange when they came upon Fig. 2b:

- 153 Tsubasa: The case when [the statement is] not true.... Oh, this is different.
 154 Kenta: Yes.... There is a case where we can't draw the perpendicular lines to the diagonal.
 155 Tsubasa: Well ...
 156 Kenta: What figure?
 157 Tsubasa: That is.
 158 Kenta: The case where [the perpendicular lines are] outside [the parallelogram].
 159 Tsubasa: Yes, it is.
 160 Kenta: Yeah. This is a case when [the perpendicular lines] disappear.
 161 Tsubasa: Well. After this [Fig. 2c].
 162 Kenta: Yes. [The perpendicular lines] disappear.
 163 Tsubasa: [The statement is] not true in this case.

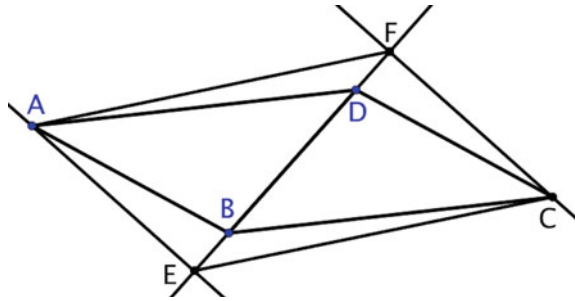
Here the undergraduates encountered a contradiction: on the one hand, before the above exchange, they had been convinced of the truth of the statement by proving it in Task 1, and had increased their conviction at the beginning of Task 2 by verifying the truth of the statement for Fig. 2a; on the other hand, during this exchange, the undergraduates actually discovered Fig. 2b, where quadrilateral AECF disappeared in the DGE because the perpendicular lines from points A and C did not intersect with diagonal BD (lines 154, 160, and 162). On this basis, the undergraduates considered this type to be a counterexample to the statement in Task 1 (line 163). After that, they began to explore this type more deeply:

- 166 Kenta: Because disappearing, we have to make this visible.
 167 Tsubasa: Yes. We have to make visible.²
 168 Kenta: Does it mean changing [diagonal] BD to a line?
 169 Tsubasa: Yes.

Both of the undergraduates said, "we have to" (lines 166 and 167), and this shows that the contradiction they faced triggered their spontaneous activity. For addressing Fig. 2b, the undergraduates extended the statement by drawing 'line BD', constructing perpendicular lines that passed through points A and C and intersected with line BD, and redefining points E and F as the intersection points of these perpendicular lines and line BD (Fig. 3).

This left the undergraduates with the matter of proving the extended statement.

²It might appear that the undergraduates perceived the problem just from whether they correctly constructed the diagram in the DGE. However, their construction was logically correct as it followed the conditions of the statement (e.g. the definitions of points E and F mentioned in the previous section). Hence, they perceived the problem from the viewpoint of logic, not just from the viewpoint of correct construction.

Fig. 3 The extended case

3.3 The Proof of the Extended Statement

Tsubasa proposed to prove that quadrilateral AECF in Fig. 3 was also a parallelogram. In the subsequent phase of the interview, he reflected on this moment: “I supposed [quadrilateral AECF was] a parallelogram at first glance, but I couldn’t be convinced”.

The undergraduates planned to show that $\triangle ABE \cong \triangle CDF$ in Fig. 3, and then Kenta suggested using their earlier proof produced in Task 1: “Well, how about a similar way to the previous one?” They found that $AE \parallel CF$ could be proved in a similar way, and started to write a proof for Fig. 3 by copying their earlier proof. However, they noticed that it was not possible to show that $\angle ABE = \angle CDF$ in the same way as before³:

251 Tsubasa: First of all, we have to show this parallel. AB parallel CD.

252 Kenta: We show the parallel. This is the same as before.

255 Tsubasa: Next is, because alternate angles are equal, right?

266 Kenta: There [$\angle ABE$] and where?

267 Tsubasa: Here [$\angle ABE$] and here [$\angle CDF$].

274 Kenta: What? Are those alternate angles?

275 Tsubasa: Can’t we say that?

276 Kenta: Alternate angles are ... inside and inside of parallel, right?

277 Tsubasa: I see. We can’t say [that]. We must consider the supplement.

278 Kenta: Since [$\angle ABE$ and $\angle CDF$ are] outside and outside.

279 Tsubasa: Well, we can make a subtraction from 180.

Tsubasa initially thought that their earlier proof was applicable here, as he supposed that $\angle ABE$ and $\angle CDF$ were alternate angles (lines 255, 267, and 275). However, Kenta’s objections (lines 266, 274, and 276) helped Tsubasa realise his mistake (line 277). After noting that $\angle ABD = \angle CDB$ (since $AB \parallel CD$), the undergraduates considered the supplements of these angles (line 279) and showed that $\angle ABE = 180^\circ - \angle ABD = 180^\circ - \angle CDB = \angle CDF$. They completed their activity by using their earlier proof to show the remaining parts. Thus, the undergraduates

³Some parts of the transcripts are omitted in this excerpt because the full transcripts are lengthy.

managed to prove the extended statement, thereby showing that Fig. 2b, which was initially considered as a counterexample, could be incorporated as an example of the extended statement.⁴

4 Discussion

In the task-based interview, the undergraduates successfully worked on a mathematical activity through which they discovered and addressed a counterexample. Their activity was stimulated by the tasks they tackled, and as discussed earlier, the design of the tasks is supported by Osana and Duponsel's (2016) framework for research on virtual manipulatives. Task 1 consists of the given diagram, which includes a certain hidden condition as the representational feature. This feature, and pedagogical support explaining the purpose of the DGE use in Task 2, allowed the undergraduates to discover counterexamples (Figs. 2b, d) to the statement in Task 1. Task 2 gave the undergraduates the opportunity to explore on their own by not specifying what they were expected to do when discovering counterexamples. The sequence of Tasks 1 and 2, which was designed by taking students' interpretations of counterexamples into consideration, stimulated the undergraduates' spontaneous activity where they succeeded in addressing the case of Fig. 2b by extending the original statement (Fig. 3) and proving that this case was no longer a counterexample to the extended statement.

The tasks in Fig. 1 were originally developed according to the task design principles (Table 1) that we elaborated based on existing studies (Komatsu & Jones, 2019). In this chapter, we have re-analysed these tasks by employing a different framework: the framework for research on virtual manipulatives (Osana & Duponsel, 2016). As mentioned in the introduction to this chapter, our work is motivated by one of the benefits of networking theoretical approaches: combining theoretical approaches contributes to obtaining multi-faceted, deeper insights into an empirical phenomenon (Prediger, Bikner-Ahsbahs, & Arzarello, 2008). One empirical phenomenon in our case is the set of tasks shown in Fig. 1, and this chapter shows that the design of the tasks is now supported by two different theoretical approaches—the task design principles in Table 1 and Osana and Duponsel's framework. The advantage of the tasks is also empirically illustrated with the results of the task-based interview (for another case study, see Komatsu & Jones, 2019).

⁴The undergraduates did not reformulate the statement by replacing the term *diagonal BD* in the problem sentence (Fig. 1) with *line BD*. Instead, the subsequent phase of the task-based interview revealed that they extended the statement by discarding the typical definition of 'diagonal', namely a *segment* connecting two nonadjacent vertices of a polygon, and redefining a diagonal as a *line* connecting such vertices. While the undergraduates succeeded in incorporating the counterexample by their redefinition, there is a problem here because their redefinition is not compatible with the normal definition of diagonals. See Komatsu and Jones (2019) for the related data and our further discussion.

Task design has recently gained more attention in mathematics education research (Jones & Pepin, 2016; Watson & Ohtani, 2015), but the benefit of task design is commonly examined in a relatively small number of case studies. Some research addresses ways to increase the possibility that task design can successfully achieve the intended learning goals, such as the iteration of design-implementation-analysis cycles (Stylianides & Stylianides, 2009). Another method exemplified in this chapter is to re-analyse existing theory-informed task design from a different theoretical perspective. Such re-analysis, which seems lacking in current research, at least in the area of proof, could strengthen the theoretical underpinnings of the task design and improve the design where necessary.

Although we implemented secondary-school-geometry tasks with prospective teachers (undergraduates) rather than secondary school students, this can be considered an advantage for teacher education (for a task-based interview with secondary school students, see Komatsu & Jones, 2019). Teachers may teach mathematics in a manner similar to their own experiences learning mathematics when they were students. For example, if teachers received instruction where proofs were treated as ritualistic and authoritative, their proof teaching is likely to follow the same instructional design. In other words, a possible way to improve proof teaching is to help prospective teachers accumulate the experiences of engaging in meaningful proving activities with a range of school mathematics tasks. As shown in this chapter, the tasks developed in our study enable prospective teachers to work with a dynamic process where mathematics grows through the dialectic of proofs and refutations. It is expected that the trainee teachers will subsequently introduce the tasks into their future classes and share these fruitful experiences with their students.

Future research could implement the tasks undertaken in this study in a classroom context and take teachers' roles into consideration. For instance, if Task 2 in Fig. 1 is enacted, students would produce a variety of diagrams as shown in Fig. 2. It is probably better for teachers to select and sequence these cases purposefully (Stein, Engle, Smith, & Hughes, 2008), rather than completely leaving them to students, so that, for example, students can focus on a worthwhile case to be explored further, and cases tackled by students can gradually become more challenging. In our task design, pedagogical support in the sense of Osana and Duponsel's framework (2016) is given in the task sentence about the purpose of the tool's use. Given the critical roles of teachers in mathematics classrooms (Johnson, Coles, & Clarke, 2017), it is imperative to consider both task design and teachers' roles in implementing the tasks.

5 Conclusion

In this chapter we present a re-analysis, using Osana and Duponsel's (2016) framework for research on virtual manipulatives, of a task design for students producing and addressing counterexamples. We illustrate the affordances of our task design through a re-analysis of a task-based interview that we conducted using one of our

designed tasks. Through the re-analysis of our task design using Osana and Duponcel's framework, and through the illustrative example, we show the benefits of networking theoretical approaches to obtain a multi-faceted and deeper insight into the empirical phenomena of students producing and addressing counterexamples using tasks that include virtual manipulatives in a DGE environment. Such an analysis helps to strengthen the theoretical underpinnings of the task design.

Authors' note

The illustrations in Figs. 1, 2 and 3 are copyright the authors. While the chapter has been written specifically for this book, we make use of previously published work (Komatsu & Jones, 2019) adapted to the theme of the book.

Acknowledgements We wish to express our thanks to the reviewer for providing helpful comments on the earlier version of this chapter. This study is supported by the Japan Society for the Promotion of Science (Nos. 15H05402, 16H02068, and 18K18636).

References

- Arzarello, F., Bartolini Bussi, M. G., Leung, A. Y. L., Mariotti, M. A., & Stevenson, I. (2011). Experimental approaches to theoretical thinking: Artefacts and proofs. In G. Hanna & M. de Villiers (Eds.), *Proof and proving in mathematics education: The 19th ICMI study* (pp. 97–143). New York, NY: Springer.
- Arzarello, F., Olivero, F., Paola, D., & Robutti, O. (2002). A cognitive analysis of dragging practises in Cabri environments. *ZDM—The International Journal on Mathematics Education*, 34(3), 66–72.
- Balacheff, N. (1991). Treatment of refutations: Aspects of the complexity of a constructivist approach to mathematics learning. In E. von Glasersfeld (Ed.), *Radical constructivism in mathematics education* (pp. 89–110). Dordrecht, Netherlands: Kluwer Academic Publishers.
- de Villiers, M. (1998). An alternative approach to proof in dynamic geometry. In R. Lehrer & D. Chazan (Eds.), *Designing learning environments for developing understanding of geometry and space* (pp. 369–393). Mahwah, NJ: Lawrence Erlbaum Associates.
- de Villiers, M. (2010). Experimentation and proof in mathematics. In G. Hanna, H. N. Jahnke, & H. Pulte (Eds.), *Explanation and proof in mathematics: Philosophical and educational perspectives* (pp. 205–221). New York, NY: Springer.
- D'Mello, S., Lehman, B., Pekrun, R., & Graesser, A. (2014). Confusion can be beneficial for learning. *Learning and Instruction*, 29, 153–170.
- Gravemeijer, K. (2002). Preamble: From models to modeling. In K. Gravemeijer, R. Lehrer, B. van Oers., & L. Verschaffel (Eds.), *Symbolizing, modeling, and tool use in mathematics education* (pp. 7–22). Dordrecht, Netherlands: Kluwer Academic Publishers.
- Herbst, P., & Brach, C. (2006). Proving and doing proofs in high school geometry classes: What is it that is going on for students? *Cognition and Instruction*, 24(1), 73–122.
- Hoyles, C., & Küchemann, D. (2002). Students' understandings of logical implication. *Educational Studies in Mathematics*, 51(3), 193–223.
- Healy, L. (2000). Identifying and explaining geometrical relationship: Interactions with robust and soft Cabri constructions. In T. Nakahara, & M. Koyama (Eds.), *Proceedings of the 24th conference of the international group for the psychology of mathematics education* (Vol. 1, pp. 103–117). Hiroshima, Japan: PME.

- Healy, L., & Hoyles, C. (2001). Software tools for geometrical problem solving: Potentials and pitfalls. *International Journal of Computers for Mathematical Learning*, 6(3), 235–256.
- Johnson, H. L., Coles, A., & Clarke, D. (2017). Mathematical tasks and the student: Navigating “tensions of intensions” between designers, teachers, and students. *ZDM: Mathematics Education*, 49(6), 813–822.
- Jones, K. (2000). Providing a foundation for deductive reasoning: Students' interpretations when using dynamic geometry software and their evolving mathematical explanations. *Educational Studies in Mathematics*, 44(1–3), 55–85.
- Jones, K., & Pepin, B. (2016). Research on mathematics teachers as partners in task design. *Journal of Mathematics Teacher Education*, 19(2–3), 105–121.
- Ko, Y. Y., & Knuth, E. J. (2013). Validating proofs and counterexamples across content domains: Practices of importance for mathematics majors. *Journal of Mathematical Behavior*, 32(1), 20–35.
- Komatsu, K. (2017). Fostering empirical examination after proof construction in secondary school geometry. *Educational Studies in Mathematics*, 96(2), 129–144.
- Komatsu, K., & Jones, K. (2019). Task design principles for heuristic refutation in dynamic geometry environments. *International Journal of Science and Mathematics Education*, 17(4), 801–824.
- Komatsu, K., Tsujiyama, Y., Sakamaki, A., & Koike, N. (2014). Proof problems with diagrams: An opportunity for experiencing proofs and refutations. *For the Learning of Mathematics*, 34(1), 36–42.
- Laborde, C. (2005). Robust and soft constructions: Two sides of the use of dynamic geometry environments. In S. C. Chu, W. C. Yang, & H. C. Lew (Eds.), *Proceedings of the tenth Asian technology conference in mathematics* (pp. 22–35). South Korea: Advanced Technology Council in Mathematics.
- Lakatos, I. (1976). *Proofs and refutations: The logic of mathematical discovery*. Cambridge, England: Cambridge University Press.
- Larsen, S., & Zandieh, M. (2008). Proofs and refutations in the undergraduate mathematics classroom. *Educational Studies in Mathematics*, 67(3), 205–216.
- Manizade, A. G., & Martinovic, D. (2016). Developing an interactive instrument for evaluating teachers' professionally situated knowledge in geometry and measurement. In P. S. Moyer-Packenham (Ed.), *International perspectives on teaching and learning mathematics with virtual manipulatives* (pp. 323–342). Cham, Switzerland: Springer.
- Marradez, R., & Gutiérrez, Á. (2000). Proofs produced by secondary school students learning geometry in a dynamic computer environment. *Educational Studies in Mathematics*, 44(1), 87–125.
- Moyer-Packenham, P. S., & Bolyard, J. J. (2016). Revisiting the definition of a virtual manipulative. In P. S. Moyer-Packenham (Ed.), *International perspectives on teaching and learning mathematics with virtual manipulatives* (pp. 3–23). Cham, Switzerland: Springer.
- Olivero, F., & Robutti, O. (2007). Measuring in dynamic geometry environments as a tool for conjecturing and proving. *International Journal of Computers for Mathematical Learning*, 12(2), 135–156.
- Osana, H. P., & Duponsel, N. (2016). Manipulatives, diagrams, and mathematics: A framework for future research on virtual manipulatives. In P. S. Moyer-Packenham (Ed.), *International perspectives on teaching and learning mathematics with virtual manipulatives* (pp. 95–120). Cham, Switzerland: Springer.
- Prediger, S., Bikner-Ahsbabs, A., & Arzarello, F. (2008). Networking strategies and methods for connecting theoretical approaches: First steps towards a conceptual framework. *ZDM—The International Journal on Mathematics Education*, 40(2), 165–178.
- Presmeg, N. C. (1986). Visualisation in high school mathematics. *For the Learning of Mathematics*, 6(3), 42–46.
- Prusak, N., Hershkowitz, R., & Schwarz, B. B. (2012). From visual reasoning to logical necessity through argumentative design. *Educational Studies in Mathematics*, 79(1), 19–40.
- Semadeni, Z. (1984). Action proofs in primary mathematics teaching and in teacher training. *For the Learning of Mathematics*, 4(1), 32–34.

- Stein, M. K., Engle, R. A., Smith, M. S., & Hughes, E. K. (2008). Orchestrating productive mathematical discussions: Five practices for helping teachers move beyond show and tell. *Mathematical Thinking and Learning*, 10(4), 313–340.
- Stylianides, G. J., & Stylianides, A. J. (2009). Facilitating the transition from empirical arguments to proof. *Journal for Research in Mathematics Education*, 40(3), 314–352.
- Stylianides, G. J., Stylianides, A. J., & Weber, K. (2017). Research on the teaching and learning of proof: Taking stock and moving forward. In J. Cai (Ed.), *Compendium for research in mathematics education* (pp. 237–266). Reston, VA: National Council of Teachers of Mathematics.
- Watson, A., & Ohtani, M. (Eds.). (2015). *Task design in mathematics education: An ICMI study 22*. New York, NY: Springer.
- Yerushalmy, M., & Chazan, D. (1990). Overcoming visual obstacles with the aid of the Supposer. *Educational Studies in Mathematics*, 21(3), 199–219.
- Zaslavsky, O., & Peled, I. (1996). Inhibiting factors in generating examples by mathematics teachers and student teachers: The case of binary operation. *Journal for Research in Mathematics Education*, 27(1), 67–78.
- Zazkis, R., & Chernoff, E. J. (2008). What makes a counterexample exemplary? *Educational Studies in Mathematics*, 68(3), 195–208.

Afterword

Proof Technology and Learning in Mathematics: Common Issues and Perspectives



Nicolas Balacheff and Thierry Boy de la Tour

1 Introduction

Mathematical proof is undoubtedly the cornerstone of mathematics. Indeed, no mathematical work is definitively complete without the final QED. Mathematics educators know this centrality of proof, the challenges it presents in terms of teaching, and the complexity of its learning. The rapid spread of mathematical digital technology over the last three decades has changed the field of mathematical experience for learners and opened new avenues in the teaching of proof. The example of dynamic geometry, as Maria-Alessandra Mariotti observes in this volume, plays a very special role in this context. The availability of more and more powerful digital technologies has also changed the work of mathematicians to the point where the very notion of mathematical proof is being questioned. Beyond mechanizing computation, these technologies are now mechanizing mathematical reasoning and proving with unprecedented consequences: logicians and computer scientists have made such progress that not only are automated theorem proving tools available, but even proof assistants. How will these new technologies impact mathematics teaching and learning?

Written by researchers from both sides—from the fields of automated theorem proving and of mathematics education—the chapters in this volume unpack the complexity and meaning of this question, open new issues, offer responses, and invite readers to think about the next step. Which comprehensive multidisciplinary projects in the near future will trigger a breakthrough in offering the smartest technology-enhanced environments for the learning and teaching of mathematical proof?

N. Balacheff (✉) · T. Boy de la Tour
Univ. Grenoble Alpes, CNRS, Grenoble INP, LIG, 38000 Grenoble, France
e-mail: nicolas.balacheff@imag.fr

T. Boy de la Tour
e-mail: thierry.boy-de-la-tour@imag.fr

© Springer Nature Switzerland AG 2019
G. Hanna et al. (eds.), *Proof Technology in Mathematics Research and Teaching*, Mathematics Education in the Digital Era 14,
https://doi.org/10.1007/978-3-030-28483-1_17

As authors of this afterword, we also come from both sides. In the following, we share our discussion of the ideas suggested by the preceding chapters. The next sections reflect our readings and visions of the future—first, in relation to automated theorem proving and second, in relation to the design of technology-enhanced environments for the learning of proof in mathematics. In the epilogue, we venture a bit further to consider a world where the mechanization of proving is fully achieved.

2 Automated Theorem Proving

As the performance of computers and Automated Theorem Provers (ATPs)¹ increases, we can expect a greater number of users of these systems either through direct learning of the indispensable but austere skills they require or through the specialized tools that academies and industries may provide. Thus, as with computer algebra systems and numerical simulations, these tools are likely to impact all activities related to mathematics. For instance, in 1964, Martinus Veltman developed the Schoonschip program for symbolic evaluation of algebraic expressions. This contributed to his work on particle physics and, in 1999, this work won him the Nobel prize.

2.1 Computer Proofs

Even if ATPs have not yet contributed to winning a Fields Medal, they have settled certain conjectures. A recent example published in Heule, Kullmann, and Marek (2016) determined the maximal integer n for which the set $\{1, \dots, n\}$ can be partitioned in two parts such that none contains three numbers a, b, c such that $a^2 + b^2 = c^2$. This number happens to be 7824. It is actually easy to prove that the set $\{1, \dots, 7824\}$ admits such a partition simply by providing a bi-partition and running a simple program to check that no triple of integers in each part has the property above. It is much more difficult to prove that *all* bi-partitions of $\{1, \dots, 7825\}$ contain such triples of integers because they total 2^{7825} . It would require a very elaborate program to explore such a huge search space in a reasonable period of time. Fortunately, SAT-solvers (ATPs devoted to propositional logic) are very elaborate programs. Indeed, it is quite remarkable that a program that was not designed specifically to solve this problem was actually able to do so simply from a standard translation into propositional logic (and some adequate tuning that need not concern us here).

¹We do not distinguish here between fully and semi-automated theorem provers. Since techniques of automated proving are present in both, we will use the acronym ATP in a generic way to make the reading smoother for non-experts. The context should be sufficient to identify the differences when necessary.

This being said, the computation took four years of CPU time, dispatched among 800 cores in parallel. The mere fact that this computation happened cannot be considered a proof of the conjecture: computing errors can occur (e.g., a cosmic ray may hit the wrong core at the wrong time) and the SAT-solver could be bugged. The specificity of ATPs compared to other mathematical software is that they are able to support their computations (in a successful event) by a proof. Thus the provability of the conjecture is reduced to the validity of the proof, and we only need to read it to be convinced. Alas, the proof produced by this computation took 200 terabytes of memory and can obviously not be read by a mathematician.

So what have we gained with this huge amount of data? Simply the fact that this proof can be checked for correctness by a simple program and independently of the SAT-solver that produced it (this is known as the *de Bruijn criterion*). Running this proof checker several times minimizes the probability of a computing error, thereby reducing the provability of the conjecture to the correctness of a simple program. Further, it is not difficult to prove that such simple proof checkers only accept valid proofs.

Of course, this kind of proof is very different from those produced by mathematicians. This is not surprising as mathematicians themselves have been unable to prove the conjecture. Only a computer has so far been able to find a proof. This is likely one characteristic of the future of ATPs in mathematics—they will help mathematicians with tasks they cannot achieve alone while also creating frustration. This is linked not only to the length of computer proofs, but also to the fact that many ATPs translate statements into an internal format (a normal form) that yields very uniform proofs. This uniformity benefits efficiency but not clarity.

2.2 *Mathematical Proofs*

The concept of proof we have sketched so far is close to the notion of *witness*, as a natural number n is the witness of membership² of an element $f(n)$ in a recursively enumerable set $\{f(n) \mid n \in \mathbb{N}\}$ (where f is a computable total function). From this purely computational point of view, sets of theorems are nothing more than recursively enumerable sets, and theorems are produced by the rules of a meaningless game. This is fine for computers, but mathematicians have a habit of endowing mathematical statements with *meaning*. The standard set-theoretic semantics of first and higher order logic constitute a good approximation of this meaning (even if categorical or other semantics may seem more appropriate as a foundation). In this semantic context, we can only admit inference rules that preserve the meaning of mathematical statements. This is what guarantees the correctness of mathematical deductions (and excludes inductive reasoning, a common but incorrect inference that should not be confused with mathematical induction).

²Let $S = \{f(n) \mid n \in \mathbb{N}\}$, then $x \in S$ iff $\exists n \in \mathbb{N}$ such that $x = f(n)$. Hence if $x \in S$, there is a witness n which proves this fact. On the opposite side, if $x \notin S$ then there is no witness to prove it.

In mathematical proofs found in textbooks, inference rules are generally implicit. Statements follow each other separated by “thus”, “hence”, and “therefore,” sometimes requiring a great deal of thinking (and writing) just to understand why one statement is implied by others. By leaving standard or secondary details to the reader, a proof can be made shorter and thus emphasize its core arguments—those that the writer perceives as pivotal. Sometimes arguments are supported by figures, as is common in category theory and geometry, or in the schematic proofs in the chapter by Alan Bundy and Mateja Jamnik in this volume. It is clear that usual mathematical proofs do not meet the *de Bruijn criterion*; no simple program can check that they are correct.

Yet mathematicians do make mistakes—sometimes superficial, sometimes deep—and every mathematical proof is as likely to contain bugs as programs are. Every devoted referee knows how difficult and time consuming it can be to hunt for bugs in proofs written by colleagues. These are strong incentives to develop software that can automatically verify mathematical proofs. As computers can only handle syntax, this means that mathematical proofs should be translated into formal proofs in a convenient logic. This is a job for ATPs.

One pioneering project in this direction is the Mizar system developed by the group founded by Andrzej Trybulec in 1973 (see Bancerek et al., 2018). The aim of this system is to automatically check given texts in a language as close as possible to standard mathematics and hence readable by a human being. The author of an article is asked to fill in intermediary details (using feedback from the system) until Mizar achieves its verification. Of course, it is tempting to use such systems not simply to check a completed mathematical proof, but also to develop an incomplete proof in a partially automated way—in other words, to use the system as a *proof assistant* by exploiting its proving capabilities.

Many proof assistant systems are designed to provide this kind of help. Among the most popular are *PVS*, *Isabelle*, and *Coq* (see the chapter by Chantal Keller in this volume). All these systems have an input language that offers the possibility of writing mathematical definitions and statements, and triggering a number of automated proving tools. Many satisfy the *de Bruijn criterion* in the sense that statements are recognized as theorems only if they have a formal proof in the logic of the system.

This leads to a concept of proof as the information provided to a proof assistant that allows it to find a formal proof of a statement (or at least to verify it). This means that what is considered a proof depends on the state of the system (including its libraries of mathematical knowledge and automated tools). More fundamentally, the problem of verifying such proofs (including mathematical proofs) is undecidable, and contrary to formal proofs where inference rules should always be decidable relations.

One important aspect of proof assistants is the language in which they require users to write mathematics. Obviously these languages, called *logical frameworks*, should be both general enough to express any kind of mathematics and close enough to the standards of mathematical language. The burden of translating from the latter to the former is left to the user and would be very difficult to automatize. It requires a good knowledge of the target language, but also of the level of detail required for

a verification to be possible. In terms of length, this translation can be expected to multiply the original text by a factor of approximately four. This is currently more time consuming than refereeing a paper but, of course, it is safer.

2.3 Proof Theory

We can also see proof assistants as systems that translate (some form of) mathematical proofs into formal proofs. We may thus examine the possibility of a reverse translation—from formal proofs to the kind of proofs that mathematicians write, and that we can read. This kind of translation would ideally leave aside all the gory details that readers do not need to know and provide only the essence of the proof. Two important methods for structuring proofs are illustrated in the chapter by Rob Arthan and Paulo Oliva: factorizing recurring terms into *definitions* and splitting the proof between *lemmas*.

Alternatively, we could design ATPs that produce readable proofs directly. For instance, Pedro Quaresma and Vanda Santos address in their chapter the challenging problem of producing proofs in geometry that can be illustrated step by step on figures and illustrate that this is not simply a problem of proof search, but mostly of the logic in which proofs are searched for. Another example is the chapter by Mohan Ganesalingam and William Timothy Gowers where most of the effort is devoted to designing a logic that would “imitate human thought.”

These problems are related to *proof theory*—the study of the structure of mathematical proofs—and particularly to the work of Gerhard Gentzen (see Takeuti, 1975). In his famous *sequent calculus*, theorems have the form $\Gamma \vdash \Delta$, where Γ and Δ are sequences of formulas (statements). The intended meaning is that the logical disjunction of the formulas on the right can be deduced from the formulas on the left. This grants a nice symmetry (or duality) between the inferences of formulas on the left and right sides. It is also undeniable that sequents of the form $\varphi \vdash \varphi$, for any formula φ , should be accepted as axioms.

Another obvious inference (at least when Δ is empty) is the *cut rule*

$$\frac{\Gamma \vdash \Delta, \varphi \quad \varphi, \Gamma' \vdash \Delta'}{\Gamma, \Gamma' \vdash \Delta, \Delta'}$$

where can be considered as a lemma that is proved in the first premise and used in the second to complete the proof. This rule is inconvenient for ATPs because using it to attempt to prove the conclusion requires guessing the lemma. The fundamental result obtained by Gentzen, known as Gentzen’s *Hauptsatz* or *cut-elimination theorem*, shows that in first order logic every proof can be transformed into a cut-free proof (possibly with a huge increase in size). Thus ATPs can conveniently search for cut-free proofs, which is close to the so-called *tableaux method* (see Hähnle, 2001). But such proofs could be much reduced if we could introduce cut rules, and therefore

lemmas, by inverting cut-elimination. Recent results in this direction have been presented in Ebner, Hetzl, Leitsch, Reis, and Weller (2018).

Most of the logical frameworks mentioned above are formulated as sequent calculi with various features convenient for proving and expressing mathematical statements. An interesting example is the $\lambda\pi$ -Calculus Modulo Theory (Assaf et al., 2016) that offers the option of separating computations from deductions, thus hiding computations in a possible translation of such proofs in natural language. Sequent calculi offer many possibilities to express statements and also proofs in different forms, and to study their possible transformations. It is clear that the future of proof technology does not rely simply on the design of efficient ATPs but also, and fundamentally, on the development of proof theory.

3 Designing Technology for Enhancing Proof Learning

3.1 Proof Tutors

Interest in ATP research for educational applications emerged in the early 70s with the belief that “building a theorem prover is an exciting alternative to the usual classroom presentation” (Goldstein, 1973, p. 3). Technically, Goldstein’s geometry prover was based on representations of procedural knowledge with the objective of “developing a high-level, ‘natural’ formalism for representing mathematical knowledge.” (Ibid.). The identified challenge was to find efficient methods for “finding a proof amidst a surfeit of geometric knowledge” (Ibid.). However, the choice of backward chaining to model reasoning was not as natural as the formalism could have been. Nevins (1974), thanks to an “efficient representation” of the knowledge base and to “confining” the exploration to objects (points and lines) implicit in the statement but appearing in the diagram, proposed a solution with forward chaining. These seminal works prepared the ground for *Geometry Tutor* (Anderson, Boyle, & Yost, 1988)—the emblematic tutor for the learning of mathematical proof designed on the principles of the ACT* model of cognition (Anderson, 1983). *Geometry Tutor* was “based on considerable theoretical analysis of the characteristics of the problem domain and on a great empirical observation of students’ behavior” (Anderson et al., 1988, p. 4). It included an “ideal model” in order to “generate proofs in what we felt was a natural, human way—in contrast to some methods of proof” (Ibid., p. 5) and “buggy models” to allow diagnosing student errors as production rules. The *Tutor* traces students’ behaviour through its ideal and buggy models and provides immediate or “near immediate” feedback to keep them on the right track.

However, *Geometry Tutor* was criticized for the restricted possibilities it gave learners, the limited relevance of its feedback, and its static diagrams. These criticisms led to the *Angle Project* (Koedinger & Anderson, 1990) which gave a more active role to diagrams, and then to a new generation of Cognitive tutors including a cognitive

model of the content to be learned and more subtle scaffolding strategies for feedback and advice (Anderson, Corbett, Koedinger, & Pelletier, 1995).

Although these tutors went to school and success was reported in various domains, they showed limits inherited from production rule systems, especially in relation to mathematical proof. These systems satisfied the requirements for modeling competences, but imposed severe limits on modeling mathematical problem solving due to their incapacity to properly express mathematical knowledge.

ATP research developed long before the birth of research on mathematical proof learning environments. It was focused on modeling human reasoning, mainly with a view to implementing propositional logic. The first significant project, the *Logic Theorist* (Newell & Simon, 1956), made a breakthrough. It had some educational use for the teaching of information processing languages (Stefferd, 1963), but this remained limited. Based on the *Principia Mathematica*, the *Logic Theorist* relied on mathematicians' introspection and a few empirical observations of students. This weakness was later overcome by a systematic observation of students solving proof problems in symbolic logic. It led to an improved model, the *General Problem Solver* (Newell, Shaw, & Simon, 1959), but this model still had to face a much more demanding challenge: "problem solving is the battle of selection techniques against a space of possibilities that keeps expanding exponentially" (Ibid., p. 26). The lessons learned from the *Logic Theorist* shaped the design of the *Geometry Machine* (Gelernter, 1959) by *limiting the domain* and searching for a *specifics heuristic* (e.g., drawing benefit from diagrams) that could model "the discovery of proofs for theorems in elementary Euclidean plane geometry in the manner, let us say, of a high school sophomore" (Ibid., p. 135). To some extent, the approach was pragmatic:

[T]he machine is granted the same privileges enjoyed by the high-school student who is always assuming (i.e., introducing as additional axioms) the truth of a plethora of 'obviously self-evident' statements concerning, for example, the ordering properties of points on a line and the intersection properties of lines in a plane. (Ibid., p. 139)

Such a statement and the evidence that the *Geometry Machine* could solve high school geometry problems opened some perspective for the design of Technology Enhanced Learning (TEL) environments for mathematical proving. Unfortunately that was not the case even though the Gelernter paper, first published in *Computer and Thoughts* in 1959, was reprinted in *Intelligent Tutoring Systems* in 1963. The convergence of ATP and educational technology fell short.

Several issues had to be addressed to make a convergence possible: the depth of the gap between machine and human ways of reasoning, the multiplicity of representations in geometry (including diagrams and hence reasoning based on visualization), the constraints on human-machine communication—just to mention a few.

Interestingly, while ATP research focused on computational models, the *Geometry Tutor* project came to recognize that "placing interface design ahead of production-systems design represents a major restructuring of our approach to tutor construction" (Anderson et al., 1995, p. 35). This was because of the impact of the interface on the skills students acquire; the interface is the space in which they experience and learn. Beneath the interface, the implemented models must ensure that communication

makes representations accessible to students and enables interactions that enhance learning—hence the following objective from the early 2000s which could be shared by many projects:

Our long-term goal is to build an intelligent tutoring system for elementary geometry. Any proofs and constructions found by our automated geometry theorem prover [GRAMY] must be stated with the common ontology of Euclidean geometry—the axiomatized geometry system taught in schools. (Matsuda & VanLehn, 2004, p. 3)

However, provers have to comply not only with the constraints of the interface, but also with the professional responsibilities of teachers. This imposes an additional filter: “the desired geometry theorem prover must not only be able to find a single comprehensible proof, it should also be able to find all proofs that are considered acceptable to instructors” (Ibid., p. 4). Along with providing ATP features for mathematicians, computer-based tutors must take three additional categories of users into account: the curriculum decision makers (who specify the standard of mathematical validation at a given grade), the teachers (who orchestrate learning and decide what counts as a proof in relation to a standard), and the learners (who are simultaneously constructing an understanding of proof and of the related content).

3.2 From Empirical Validation to Mathematical Proof

How to assess the validity of learners’ statements is a question throughout primary and secondary schooling. Indeed, the means and criteria of assessment from the perspective of the learner and the perspective of the teacher at each grade level differ in a significant way. Even at the university level, students continue to shape their understanding of what counts as a mathematical proof; for example, despite being familiar with proving in advanced algebra or calculus, they are often destabilized by proving in discrete mathematics (e.g., graph theory). However, the gap between advanced learning and early learning constitutes another order of magnitude. At the earliest stages, validation is dominated by naive induction and empirical verification. It is constrained by language, discourse structures, and the available representations of mathematical objects and relations. This approach gradually develops into more elaborate types of argumentation based on generic examples or thought experiments. These early forms of argumentation reflect the everyday structure of argumentative discourse, which is not devoid of logic but can display the ill-defined content and implicitness of ad hoc situations. Hence, at some point in the middle school curriculum, mathematics teachers face the challenge of introducing students to the sociomathematical norms of argumentation (Yackel & Cobb, 1996, p. 466 sqq) and ultimately to the basics of mathematical proof. This is also a challenge for the design of ATP-based TEL environments, whether based on proof assistants for pedagogical use or proof tutors.

Learning mathematical proof is less an evolution than an epistemological revolution: students must evolve from the position of practitioner, driven by the practical

realities of a situation, to the position of theoretician, driven by the requirements of knowledge. The validity of a mathematical statement does not depend on the beliefs and conceptions of the proposer (*epistemic value*, Duval, 2007, p. 138) or of a community, but only on the relations it has with other previously validated statements following given norms of inference which ensure that it has a place within a mathematical theory (*ontic value*, Hanna, 2017). Being aware of the existence of such a reference theory bounding the discourse supporting the validity of a mathematical statement is as critical as understanding and mastering logical rules—hence the claim that a *mathematical theorem* is not a statement backed by a proof but, as Mariotti states in her chapter, by “the system of statement, proof and theory.” This conception of theorem is consistent with the conception driving the design of TEL environments. It offers a starting point for the convergence of both fields of research by emphasizing that decisions must be based not only on the nature and complexity of proofs, but also, and inseparably, on the underlying theory.

This definition of theorem builds a first bridge between the two sides of a proof—validation and explanation. Explanation, in the sense of Duval (1992), refers here to a system of relations in which the statement to be explained finds its place. While solving a problem comforts the positive epistemic value of the solution statement, its meaning comes from the coherence of this statement’s membership with the problem-solver system of knowledge. Hence, a consequence for the relevance and viability of the pedagogical use of ATP-based TEL environments is that these environments should either facilitate learners’ appropriation of the epistemic value of the solution (providing an argumentation) or situate the proof in a theoretical framework compatible and coherent with the learners’ system of knowledge (providing an explanation). The difficulty in doing so comes from the inherent formal nature of ATPs, although logicians and computer scientists are aware that “at the research level, mathematics is often quite vaguely formulated because mathematicians can usually rely on the deep understanding and intuition of themselves and their fellows to keep them out of trouble” (Harrison, Urban, & Wiedijk, 2014, p. 57). As a matter of fact, in a less “radical” way, this also holds true for the learner.

When students propose a proof, and teachers present a proof to the class, they are supposed to find the right balance between what must be made explicit and what is tacitly accepted. This does not mean a lack of rigour or a tolerance of vague approximations. It is just a property of human communication that applies to mathematics as it does to other domains: a mathematical proof is a discourse that should be both convincing and meaningful. Indeed, throughout its history, mathematics has framed its discourse and established a standard centered on formalization as a means of ensuring the rigour and validity of its outcomes. But as the Bourbaki group—the paragon of formalism—claimed, this mathematical discourse remains “naïve” insofar as it includes natural language and necessarily admits some shortcuts in proofs.

Looking at proof as a discourse reveals an unresolved tension. In the most classic teaching tradition, solving a problem is presented step by step in such a way that the end produces at once both the solution and the proof. This is reinforced by the use of representations such as two-column proofs, flow-chart proofs, and standardized paragraph proofs. These reify the mathematical norm, monitor or scaffold

students' activity, and serve as a professional pedagogical resource (Weiss, Herbst, & Chen, 2009). The private activity of the student, like the private activity of the mathematician, is foreign to this ideal picture. When not reduced to the application of a few theorems or the implementation of a standard reasoning, problem solving is driven and supported by *heuristic argumentation* (Duval, 1992, p. 51). Heuristic argumentation is not meant to convince, but to allow choices in a context where not all statements and possibilities have been strictly verified or even made explicit, as is the case, for example, with plausible reasoning (Polya, 1945). Writing a proof consists of logically structuring, making explicit, and verifying all the statements that relate the hypothesis to the claimed solution. In terms of the content, a continuity exists between the heuristic argumentation and the proof, precisely observed and conceptualized by Paulo Boero as the “cognitive unity of theorems”:

During the production of the conjecture, the student progressively works out his/her statement through an intensive argumentative activity functionally intermingled with the justification of the plausibility of his/her choices. During the subsequent statement-proving stage, the student links up with this process in a coherent way, organising some of the previously produced arguments according to a logical chain. (Garuti, Boero, & Lemut, 1998)

When associated with a purely deductive problem-solving process, this continuity facilitates a natural transition to proof. But when the problem has some complexity, abduction and induction having played a key role, a structural distance emerges between argumentation and proof (Pedemonte, 2007) even though the discursive surface structure of argumentation may be close to the structure of a proof (Duval, 1992, p. 38). The transition from problem solving to proving is not straightforward, and students have to achieve it in a demanding context. First, the representations and controls they use may lead to technical difficulties. Second, they have to adapt to sociomathematical norms they are in the process of learning and understanding. Third, they have to comply with the teacher's evolving expectations. Despite the analogies between the work of mathematicians and students, there are differences of an epistemological nature that result from the process of teaching mathematical proof (Herbst & Balacheff, 2009) and from the distance from professional values and norms (Dawkins & Weber, 2016).

Results from research on the learning of mathematical proof show that the contribution of ATPs requires understanding and solving new problems. Some design issues are common—mainly bridging the gap between human reasoning and ATP models, and delivering readable write ups. But addressing learning raises other issues such as: (i) bridging the gap between argumentation and proof, (ii) facilitating the transition from problem solving to proving, and (iii) scaffolding the transition from empirical validation to the mathematical norms at play in the classroom.

3.3 Technology Enhanced Learning of Mathematical Proof

The largest body of research on technology and the learning of proof has focused on Dynamic Geometry Environments (DGEs) (see, e.g., Sinclair et al., 2017). This

is an effect of the influence of Logo and the concept of microworld born in the 80s. The strength of DGEs is that they provide students with a *field of experience* (Boero et al., 1995) for exploring mathematical facts, making conjectures, shaping their argumentation, and proposing proofs (Baccaglini-Frank & Mariotti, 2010). Because research on proof tutors was essentially related to AI and cognitive science, the contribution of ATP research to research on mathematics education has until recently been rather limited (Hauer, Kovács, Recio, & Vélez, 2018, p. 2). One of the many DGEs, *Cinderella*, includes a randomized theorem prover that offers the possibility of checking and proving properties. But, as Kortenkamp and Richter-Gebert (2004) point out: “it has been developed as a tool for mathematicians in research, publishing and teaching” (p. 1). Actually, to mathematics educators, ATP technology seems far from satisfying the constraints of the educational context. Some ATP researchers are well aware of this critique: “[ATP] generated proofs are unnatural and incomprehensible as these systems do not approach geometry problems like how secondary school students are taught” (Wang & Su, 2017, p. 10). But are models that approach mathematics in the same way as learners do necessary to TEL environments? The example of *Cinderella* suggests this is not strictly the case:

Sometimes the semantics of a user’s action can be unclear. For example, when he tries to insert the intersection of three lines by placing a point on it, it is unclear whether he assumes that these always meet in one point or not. A software that “knows” whether the three lines are always concurrent can react properly in that situation: When they are, it is not necessary to ask which pair of points should define the intersection, as all three define the same one. When they are not, the software can signalize that it needs the attention of the user to resolve an ambiguity. (Kortenkamp & Richter-Gebert, 2004, p. 8)

Cinderella did not primarily target an educational audience. The seminal project *Cabri-géomètre*, which does, included an algebraic oracle that could verify properties and support student inquiries with messages like: “The property is true in general” or “The property seems true on the figure but is false in general. A counterexample can be proffered” or “The property is true for this figure but *Cabri-géomètre* cannot tell for the general case” (Laborde, 1990, p. 137). This feature was exploited to develop the proof microworld *Cabri-Euclide* (Luengo, 1997).

The question then is: which services can ATP provide in a TEL environment? Or, more precisely, what kind of feedback could ATP provide while the student is engaged in problem solving and proving?

Depending on whether the student is solving a problem or validating a solution, the feedback could target the strategy rather than a particular rule, the related knowledge and not the logical thread, or the standard for a successful achievement at a given grade and not the underlying logical structure. It could immediately spot the misuse of a theorem or a flaw in reasoning, or wait to provide a counterexample to the proposed proof. It could be textual or visual and, in the case of geometry, preferably both. In this sense, geometry is the best case for addressing the difficulties in designing such environments, especially for early learning.

A DGE is a natural working space that enables free heuristic argumentation for solving a geometrical problem. These environments have a built-in validating feedback that is prompted by messing up a construction (Healy, Hoelzl, Hoyles, & Noss,

1994). ATP could provide further services in terms of confirming or invalidating certain properties on demand. For example, the DGE *GeoGebra* “[can answer] a query posed by a user about the truth or falsity of any geometric statement” or “present further hypotheses that should be considered for the proposition to become true” (Hauer, Kovács, Recio, & Vélez, 2018, p. 2). Teachers may expect a service such as scaffolding the writing of a proof, hence providing feedback on a textual representation of the solution to a problem. Still, feedback could have a textual or a visual form.

Feedback responds to actions of the student who, in turn, can respond with new actions and hence with new decisions on how to solve a problem or shape a proof. Unlike ATP, the reasoning of the student is semi-empirical rather than apodictic. It is not exact and stable, and may inherit from experience as well as formal learning. Let’s call this knowledge under construction a *conception* (which is not necessarily a misconception). One could characterize it as the operators, controls, and semiotic representations the student uses in order to solve a given set of problems (Balacheff, 2010). The feedback may lead to an evolution of the reasoning, the proof, or related conceptions. From the perspective of early learning, an ATP-based environment is less a tool than a rational agent able to communicate and argue, and build its reasoning on information gathered from student activities and conceptions as they evolve (Luengo, 1999).

Nowadays, digital technologies can be used by mathematicians to solve problems and supplement their proofs. Significantly, the editors of the *Annals of Mathematics* outline the journal’s conditions for considering “computer-assisted proof of important mathematical theorems”:

The human part of the proof, which reduces the original mathematical problem to one tractable by the computer, will be refereed for correctness in the traditional manner. The computer part may not be checked line-by-line, but will be examined for the methods by which the authors have eliminated or minimized possible sources of error [...] We will print the human part of the paper in an issue of the *Annals*. The authors will provide the computer code, documentation necessary to understand it, and the computer output, all of which will be maintained on the *Annals of Mathematics* website online. (*Annals of Mathematics*, retrieved 01-22-2019)

Thanks to their progress, ATP-based tools can be expected to contribute to solving the most important problems in mathematics or checking very long and complex proofs such as Fermat’s last theorem, and hence become part of the “computer-assisted proofs” the editors of the *Annals of Mathematics* mention.

Nevertheless, this progress is not sufficient for educational needs; research needs to take further steps to solve the specific problems raised by learning. Such improvement is possible by integrating a *didactic heuristic* into the best suited tools, as well as a distributed architecture for allocating services. The latter reflects Gelernter’s vision of splitting complexity among specialized entities: a “syntax computer,” a “diagram computer,” and a “heuristic computer” (Gelernter, 1959, p. 139). For example, a “write-up computer” that uses the services of an ATP engine could be in charge of high level communication. This suggests looking precisely at what a multi-agent architecture could offer (Webber, Pesty, & Balacheff, 2002). As to the former, the

specificity of learning situations in mathematics creates favorable conditions for ATP searches for a solution to a problem or the verification of a proof. First, it is possible to use the hypothesis of “proving in a closed world” (Trilling, 1996): any statement is true because claimed so by the problem statement, or can be proved with limited effort. Second, the ATP logical model can be augmented with a complete description of a priori accepted statements (theorems, definitions), instantiating the theory and completing the triplet (*statement, proof, theory*). Third, such a *theory* need not be complete with respect to mathematics itself, nor to one of its subdomains. It could be a *micro-theory* in the sense of Minsky (i.e., sufficient for solving a cluster of problems at a specific grade level). Fourth, the degree of implicitness and the desirable structure of the proof can be specified by the teacher providing multiple examples. Comparing proofs replaces the obligation to conform to an “ideal” proof.

Finally, what could a TEL environment look like that takes advancements in research on mathematics education into account? First, it would facilitate identifying different types of activity and enabling navigation among them. A problem-solving space (e.g., a mathematical microworld) could be identified and associated with a formulation space (e.g., a digital notebook for expressing and freely linking mathematical statements) and with a validation space (for expressing a proof in line with the curriculum; e.g., as a two column proof, flowchart proof, graph, or paragraphs). Second, the ATP would be initialized with axioms and theorems chosen by the teacher or stipulated by the curriculum. This initialization would create a theoretical reference framework in line with Mariotti’s model of theorem (statement, proof, theory) and be easily accessible to students. It would give students a foundation for understanding that they are proving a statement within a theory.

The problem-solving space would be open to students’ strategies and friendly to empirical validation. The formulation space would set the context and gather data for grounding ATP services such as verifying proofs, diagnosing flaws, building counterexamples, comparing proofs, or providing hints. The productive links between the different spaces would create a digital *mathematical working space* responding to students’ early need for support in pragmatic and theoretical thinking (see the chapter by Philippe Richard, Fabienne Venant, & Michel Gagnon). These links would also enhance the ATP’s ability to get an accurate picture of student activity from their DGE diagrams (Caferra, Peltier, & Puitg, 2001). Some of these features are already included in existing designs—*Cabri-Euclide* (Luengo, 1997); *Baghera* (Soury-Lavergne, 2003); *Advanced Geometry Tutor* (Matsuda & VanLehn, 2004); *AgentGeom* (Cobo, Fortuny, Puertas, & Richard, 2007); *PACT Geometry Tutor* (Aleven, 2010); and *QED-Tutrix* (Leduc, 2016)—hence making these suggestions plausible. Integrating them fully and using the full power of ATP would be a breakthrough.

4 Epilogue

The editors of this book remind us of the dream of Vladimir Voevodsky: “to have an electronic proof assistant check any theorem before its publication, as a way to accelerate the reviewing process.” This dream could be extended further, questioning the need for both mathematicians and mathematics teachers.

The development of collaborative mathematics through online social media, as addressed in the chapter by Lorenzo Lane and his colleagues, opens the same question as other collaborative online activities: can participants be replaced by machines? This is an important problem for online poker or chess where machines now exist that are much better players than humans. This is of course not the case for mathematics, but it suggests a kind of Turing test for ATPs which could be granted the status of *artificial mathematicians* provided they were able to participate in a collaborative mathematical project and remain undetected by human participants. To pass this test, a machine would need to be able to read mathematical texts, develop mathematical ideas related to the project, and write them down as a human would (this last part of the test is indeed considered in the first chapter of this volume). This seems impossibly difficult, but may be more accessible to machines than the kind of knowledge needed to live in a human society with a human body, which machines can hardly experience. At least artificial mathematicians would be more worthy of artificial intelligence than the famous chatbot Tay, more adequately classified among the artificial deplorables.

But the challenge of designing such systems cannot be disconnected from the social significance of mathematics and the reasons that compel us to search, write, and teach mathematical proofs. One may argue that the reason for funding these activities is essentially economic, but as an economy is always required for funding anything, this is not a specific explanation. Jean Dieudonné asserted that mathematics exist “for the honor of the human mind.” There is grandeur in this view of mathematics, but it seems incompatible with the use of computers to overcome our deficiencies. Whatever the reason may be, it certainly has to do with the discrepancy between our natural abilities and the requirements of rational thinking. This suggests that learning mathematical proofs is not simply a challenge for students (or their teachers), but part of normal mathematical activity and deeply intertwined with searching and writing proofs, and learning mathematics. As noted by Jeremy Avigad in this volume, after having worked with the *Lean* ATP, “logic [was] relegated to the background, and the courses turned to the fundamental building blocks of mathematics.” Both mathematicians and learners are engaged in constructing new mathematical content and shaping new mathematical concepts and tools. Mathematical proof as the cornerstone should not hide the beauty and essence of this construction.

Ultimately, one is always a student who tries to find a proof, and always a teacher who writes it down. The nature of the relation between these two aspects is hard to fathom, but should be considered a major issue in automated theorem proving and the teaching of proof.

References

- Aleven, V. (2010). Rule-based cognitive modeling for intelligent tutoring systems. In R. Nkambou, J. Bourdeau, & R. Mizoguchi (Eds.), *Advances in intelligent tutoring systems* (Vol. 308, pp. 33–62). Berlin, Heidelberg: Springer. https://doi.org/10.1007/978-3-642-14363-2_3.
- Anderson, J. R. (1983). *The architecture of cognition*. Cambridge, MA, USA: Harvard University Press.
- Anderson, J. R., Boyle, C. F., & Yost, G. (1988). *The geometry proof tutor* (Advanced Computer Tutoring Project). Carnegie-Mellon University, Pittsburgh, PA 15213. Retrieved from <http://act-r.psy.cmu.edu/wordpress/wp-content/uploads/2012/12/124GeoTutor.ABYost.pdf>.
- Anderson, J. R., Corbett, A. T., Koedinger, K. R., & Pelletier, R. (1995). Cognitive tutors: Lessons learned. *Journal of the Learning Sciences*, 4(2), 167–207. https://doi.org/10.1207/s15327809jls0402_2.
- Annals of Mathematics. (n.d.). Statement by the editors on computer-assisted proofs. Retrieved 22 January, 2019, from <http://annals.math.princeton.edu/board>.
- Assaf, A., Burel, G., Cauderlier, R., Delahaye, D., Dowek, G., Dubois, C., ... Saillard, R. (2016). *Expressing theories in the $\lambda\Pi$ -calculus modulo theory and in the Dedukti system*. Presented at the 22nd International Conference on Types for Proofs and Programs (TYPES 2016), Novi Sad, Serbia: Springer.
- Baccaglioni-Frank, A., & Mariotti, M. A. (2010). Generating conjectures in dynamic geometry: The maintaining dragging model. *International Journal of Computers for Mathematical Learning*, 15(3), 225–253. <https://doi.org/10.1007/s10758-010-9169-3>.
- Balacheff, N. (2010). Bridging knowing and proving in mathematics: An essay from a didactical perspective. In G. Hanna, H. N. Jahnke, & H. Pulte (Eds.), *Explanation and proof in mathematics* (pp. 115–135). Berlin Heidelberg: Springer.
- Bancerek, G., Byliński, C., Grabowski, A., Kornilowicz, A., Matuszewski, R., Naumowicz, A., et al. (2018). The role of the Mizar mathematical library for interactive proof development in Mizar. *Journal of Automated Reasoning*, 61(1–4), 9–32. <https://doi.org/10.1007/s10817-017-9440-6>.
- Boero, P., Dapuzeto, C., Ferrari, P., Ferrero, E., Garuti, R., Lemut, E., ... Scali, E. (1995). Aspects of the mathematics—Culture relationship in mathematics teaching-learning in compulsory school. In L. Meira & D. Carraher (Eds.), *Proceedings of the Annual Conference of the International Group for the Psychology of Mathematics Education* (17 pp.). Recife. Retrieved from http://didmat.dima.unige.it/progetti/COFIN/biblio/art_boero/boero%26c_PME_XIX.pdf.
- Caferra, R., Peltier, N., & Puitg, F. (2001). Emphasizing human techniques in automated geometry theorem proving: A practical realization. In J. Richter-Gebert & D. Wang (Eds.), *Presented at the Workshop on Automated Deduction in Geometry*, Zurich, Switzerland (Vol. LNAI 2061, pp. 268–305). Berlin, Heidelberg: Springer. Retrieved from <https://link-springer-com.gaelnomade-1.grenet.fr/content/pdf/10.1007%2F3-540-45410-1.pdf>.
- Cobo, P., Fortuny, J. M., Puertas, E., & Richard, P. R. (2007). AgentGeom: A multiagent system for pedagogical support in geometric proof problems. *International Journal of Computers for Mathematical Learning*, 12(1), 57–79. <https://doi.org/10.1007/s10758-007-9111-5>.
- Dawkins, P. C., & Weber, K. (2016). Values and norms of proof for mathematicians and students. *Educational Studies in Mathematics*, 95(2), 123–142. <https://doi.org/10.1007/s10649-016-9740-5>.
- Duval, R. (1992). Argumenter, prouver, expliquer: continuité ou rupture cognitive? *Petit x*, 31, 37–61.
- Duval, R. (2007). Cognitive functioning and the understanding of mathematical processes of proof. In P. Boero (Ed.), *Theorems in school: From history, epistemology and cognition to classroom practice* (pp. 137–161). Sense Publishers.
- Ebner, G., Hetzl, S., Leitsch, A., Reis, G., & Weller, D. (2018). On the generation of quantified lemmas. *Journal of Automated Reasoning*, 1–32. <https://doi.org/10.1007/s10817-018-9462-8>.
- Garuti, R., Boero, P., & Lemut, E. (1998). Cognitive unity of theorems and difficulties of proof. In A. Olivier & K. Newstead (Eds.), *Proceedings of the 22th Conference of the International*

- Group for the Psychology of Mathematics Education* (Vol. 2, pp. 345–352). Stellenbosch (SA). Retrieved from <http://www.mat.ufrgs.br/~portosil/garuti.html>.
- Gelernter, H. (1959). Realization of a geometry-theorem proving machine. In J. H. Siekmann & G. Wrightson (Eds.), *Automation of reasoning* (pp. 99–122). Berlin, Heidelberg: Springer. https://doi.org/10.1007/978-3-642-81952-0_8.
- Goldstein, I. (1973). *Elementary geometry theorem proving* (AIM No. 280) (p. 46). MIT AI Laboratory. Retrieved from <https://dspace.mit.edu/bitstream/handle/1721.1/5798/AIM-280.pdf?sequence=2>.
- Hähnle, R. (2001). Tableaux and related methods. In A. Robinson & A. Voronkov (Eds.), *Handbook of automated reasoning* (Vol. 1, pp. 101–178). Elsevier Science B.V.
- Hanna, G. (2017). Connecting two different views of mathematical explanation. In *Enabling mathematical cultures*. Mathematical Institute, University of Oxford. Retrieved from <https://enablingmaths.wordpress.com/abstracts/>.
- Harrison, J., Urban, J., & Wiedijk, F. (2014). History of interactive theorem proving. in *handbook of the history of logic* (Vol. 9, pp. 135–214). Elsevier. <https://doi.org/10.1016/B978-0-444-51624-4.50004-6>.
- Hauer, B., Kovács, Z., Recio, T., & Vélez, P. (2018). Automated reasoning in elementary geometry: Towards inquiry learning. *Pädagogische Horizonte*, 2(2), 14.
- Healy, L., Hoelzl, R., Hoyles, C., & Noss, R. (1994). Messing up. *Micromath*, 10(1), 14–17.
- Herbst, P., & Balacheff, N. (2009). Proving and knowing in public: The nature of proof in a classroom. In D. A. Stylianou, M. L. Blanton, & E. J. Knuth (Eds.), *Teaching and learning proof across the grades: A K-16 perspective* (pp. 40–63). New York: Routledge.
- Heule, M. J. H., Kullmann, O., & Marek, V. W. (2016). *Solving and Verifying the Boolean Pythagorean Triples problem via Cube-and-Conquer* (Vol. LNCS 9710, pp. 228–245). Presented at the SAT 2016. Springer. https://doi.org/10.1007/978-3-319-40970-2_15.
- Koedinger, K., & Anderson, J. R. (1990). Theoretical and Empirical Motivations for the Design of ANGLE: A New Geometry Learning Environment. Presented at the Knowledge-Based Environments for Learning and Teaching, Stanford University. Retrieved from <http://pact.cs.cmu.edu/pubs/Koedinger,%20Anderson%20-90.pdf>.
- Kortenkamp, U., & Richter-Gebert, J. R. (2004). Using automatic theorem proving to improve the usability of geometry software. In *Proceedings of MathUI 2004* (p. 12). Retrieved from <https://pdfs.semanticscholar.org/8892/faa455ea7442438d3f126bd05ba4d8c51e81.pdf>.
- Laborde, J.-M. (1990). *Cabri-géomètre - Manuel de l'utilisateur*.
- Leduc, N. (2016). *QED-Tutrix: Système tutoriel intelligent pour l'accompagnement d'élèves en situation de résolution de problèmes de démonstration en géométrie plane*. Montréal: Université de Montréal.
- Luengo, V. (1997). *Cabri-Euclide : un micromonde de preuve intégrant la réfutation*. Université Joseph Fourier (Grenoble 1), Grenoble. Retrieved from https://www.researchgate.net/publication/34765259_Cabri-euclide_un_micromonde_de_preuve_integrant_la_refutation_principes_didactiques_et_informatiques_Realisation.
- Luengo, V. (1999). Semi-empirical agent to learn mathematical proof. In *Proceedings of Artificial Intelligence in education (AIED 99)* (p. 10). Le Mans, France: Amsterdam: IOS.
- Matsuda, N., & VanLehn, K. (2004). GRAMY: A geometry theorem prover capable of construction. *Journal of Automated Reasoning*, 32(1), 3–33. <https://doi.org/10.1023/B:JARS.0000021960.39761.b7>.
- Nevins, A. J. (1974). *Plane geometry theorem proving using forward chaining* (AIM No. 303) (p. 35). MIT AI Laboratory. Retrieved from <https://dspace.mit.edu/bitstream/handle/1721.1/6218/AIM-303.pdf?sequence=2>.
- Newell, A., Shaw, & Simon, H. (1959). *Report on a general problem-solving program* (p. 27). RAND Corporation. Retrieved from http://bitsavers.trailing-edge.com/pdf/rand/ipl/P-1584_Report_On_A_General_Problem-Solving_Program_Feb59.pdf.

- Newell, A., & Simon, H. (1956). *The logic theory machine—A complex information processing system* (No. P-868) (p. 40). The Rand Corporation. Retrieved from <http://shelf1.library.cmu.edu/IMLS/MindModels/logictheorymachine.pdf>.
- Pedemonte, B. (2007). How can the relationship between argumentation and proof be analysed? *Educational Studies in Mathematics*, 66(1), 23–41. <https://doi.org/10.1007/s10649-006-9057-x>.
- Polya, G. (1945). *How to solve it*. Princeton University Press. Retrieved from <https://press.princeton.edu/titles/669.html>.
- Sinclair, N., Bartolini Bussi, M. G., de Villiers, M., Jones, K., Kortenkamp, U., Leung, A., & Owens, K. (2017). Geometry education, including the use of new technologies: A survey of recent research. In G. Kaiser (Ed.), *Proceedings of the 13th International Congress on Mathematical Education* (pp. 277–287). Cham: Springer International Publishing. https://doi.org/10.1007/978-3-319-62597-3_18.
- Soury-Lavergne, S. (Ed.). (2003). *Baghera an hybrid and emergent educational society* (Cahier du laboratoire Leibniz No. 81). Laboratoire Leibniz - IMAG.
- Stefferd, E. (1963). *The logic theory machine: A model of heuristic program* (Memorandum No. RM-3731-CC) (198 pp.). The Rand Corporation. Retrieved from <https://history-computer.com/Library/Logic%20Theorist%20memorandum.pdf>.
- Takeuti, G. (1975). *Proof Theory*. Amsterdam: North Holland.
- Trilling, L. (1996). Rétrospective sur le projet Mentoniez. *Sciences et Technologies de l'Information et de la Communication pour l'Éducation et la Formation*, 3(2), 157–162. <https://doi.org/10.3406/stice.1996.1294>.
- Wang, K., & Su, Z. (2017). Interactive, intelligent tutoring for auxiliary constructions in geometry proofs. [arXiv:1711.07154](https://arxiv.org/abs/1711.07154) [Cs, Math]. Retrieved from <http://arxiv.org/abs/1711.07154>.
- Webber, C., Pesty, S., & Balacheff, N. (2002). A multi-agent and emergent approach to student modelling. In F. van Harmelen (Ed.), *15th European Conference on Artificial Intelligence (ECAI 2002)* (pp. 98–102). IOS Press. Retrieved from <https://telearn.archives-ouvertes.fr/hal-00003043>.
- Weiss, M., Herbst, P., & Chen, C. (2009). Teachers' perspectives on “authentic mathematics” and the two-column proof form. *Educational Studies in Mathematics*, 70(3), 275–293. <https://doi.org/10.1007/s10649-008-9144-2>.
- Yackel, E., & Cobb, P. (1996). Sociomathematical norms, argumentation, and autonomy in mathematics. *Journal for Research in Mathematics Education*, 27(4), 458–477. <https://doi.org/10.2307/749877>.

Author Index

A

Abánades, M., 230, 238
Adams, M., 15, 85
Adams, W. W., 316
Aho, A. V., 122
Aldon, G., 309
Aleven, V., 361
Allen, S. F., 73
Alsina, C., 141, 146
Anderson, J. R., 291, 294, 354, 355
Antonini, S., 174, 187, 188, 190
Appel, K., 139, 154
Armand, M., 74, 80, 83
Arora, S., 125
Arsac, G., 118, 183
Artaud, M., 128
Artigue, M., 165, 288
Arzarello, F., 157, 176, 178, 183, 332, 336, 342
Asher, N., 21
Asperti, A., 15, 73, 85
Assaf, A., 354
Attridge, N., 290
Avigad, J., 15, 73, 85, 278
Awtry, T., 314

B

Baccaglioni-Frank, A., 174, 184, 185, 188, 190, 359
Bachmair, L., 74
Back, R. J., 328
Baeta, N., 250
Bainville, E., 256
Baker, S., 65

Balacheff, N., 128, 142, 158, 173, 233, 331, 337, 358, 360
Ballantyne, A. M., 21
Bancerek, G., 73, 352
Barak, B., 125
Barany, M., 205
Barbin, É., 141
Barbosa, H., 75
Baron, G. L., 117
Barra, M., 149
Barrett, C., 74, 77
Bartle, R. G., 158
Bartolini Bussi, M. G., 147, 175, 177, 178, 332, 358
Barwise, J., 289
Barzilay, R., 19
Basu, S., 116
Bauer, G., 15, 85
Baulac, Y., 219
Beeson, M., 22, 314, 326, 327
Bellemain, F., 219
Bérard, B., 127
Berger, M., 255
Bernardo, G., 314, 320
Bertot, Y., 15, 85
Besson, F., 73, 78, 79
Bezem, M., 238, 245
Bieda, K. N., 173
Biere, A., 74
Bikner-Ahsbans, A., 332, 342
Björner, N., 74, 75
Blanchette, J. C., 74
Blazy, S., 86
Bledsoe, W. W., 21
Blok, W. J., 96

- Blum, W., 147
 Boaler, J., 204, 205, 207
 Boero, P., 173, 177, 182, 183, 358, 359
 Boesen, J., 313, 322
 Boespflug, M., 81
 Böhme, S. B., 74
 Bolyard, J. J., 332
 Bonacina, M. P., 86
 Bonnycastle, J. F., 320
 Boole, G., 314
 Borromeo-Ferri, R., 147
 Borwein, J. M., 116, 256
 Bosbach, B., 96
 Botana, F., 218, 219, 230, 238, 240, 250
 Bouallegue, S., 250
 Bouhineau, D., 313, 326
 Boulmé, S., 86
 Bourdieu, P., 201–203
 Boutin, F., 143
 Bouton, T., 74, 78
 Boyer, R. S., 21, 22
 Boyle, C. F., 291, 354
 Brach, C., 335
 Brancker, T., 328
 Brousseau, G., 128, 140, 156, 158, 164
 Bruillard, É., 117
 Brummayer, B., 74
 Bryant, J., 147
 Buchberger, B., 20, 220, 221, 223
 Büchi, J. R., 96
 Bueno-Ravel, L., 117
 Bundy, A., 14, 66, 70, 209, 327
 Burel, G., 354
 Burkhardt, H., 327
 Burris, S., 91
 Byliński, C., 73, 352
- C**
 Caferra, R., 361
 Calegari, F., 203
 Campos, H., 250
 Carmen, B., 314, 320
 Carter, N. C., 290
 Castelnuovo, E., 149
 Cauderlier, R., 354
 Chaachoua, H., 313, 326
 Chabert, J. L., 116
 Chazan, D., 335
 Chen, C., 358
 Cheng, Y.-H., 294
 Chernoff, E. J., 337
 Chevillard, Y., 157
- Chou, S. C., 215, 218, 219, 238–242, 245–247, 249
 Clairaut, A. C., 140–142, 148, 154
 Clarke, D., 343
 Clarke, E., 22
 Cobb, P., 240, 356, 361
 Coen, C. S., 73
 Cohen, C., 15, 85
 Coles, A., 343
 Colliard, M. N., 121
 Colton, S., 116
 Constable, R. L., 19, 73
 Conway, C. L., 74
 Coquand, T., 245
 Corbeil, J. P., 154–156
 Corbett, A. T., 294, 355
 Corless, R. M., 220
 Cormack, L. B., 147
 Cormen, T. H., 125
 Cornilleau, P.-E., 78, 79
 Corpuz, J., 217, 218
 Coutat, S., 147, 158
 Crăciun, A., 20
 Cramer, M., 22
 Crowdmath, 207
 Cruz-Filipe, L., 76
- D**
 Da Costa, N. C. A., 119
 Dahn, B. I., 91
 Dale, R., 27
 Dang, D. T., 15, 85
 Dapueto, C., 359
 Davis, P. J., 205, 231
 Dawkins, P. C., 358
 Deaney, R., 239
 de Bruijn, N. G., 60
 Déharbe, D., 74, 78
 Delahaye, D., 354
 Delignat-Lavaud, A., 74, 86
 de Morgan, A., 199
 de Moura, L. M., 73–75, 77
 Dénès, M., 81
 de Oliveira, D., 74, 78
 Derouet, C., 145, 156
 Deters, M., 74
 de Villiers, M., 110, 173, 191, 219, 232, 233, 237, 239, 256, 291, 331, 358
 Dimova, D., 74
 D’Mello, S., 337
 Douady, R., 130
 Dowek, G., 354

Doyle, W., 294
 Dreyfus, T., 165, 176
 Dubois, C., 354
 Duponsel, N., 332–334, 336, 337, 342–344
 Dupré, S., 139, 148, 149
 Durand-Guerrier, V., 118, 121
 Durell, C. V., 320
 du Sautoy, M., 200, 203, 209
 Duval, R., 140, 141, 158, 173, 176, 357, 358
 Dweck, C. S., 204

E

Eaton, R., 73
 Ebner, G., 354
 Edelson, D., 294
 Efremovitch, V. A., 269
 Ehrhardt, C., 202
 Ekici, B., 74, 81, 85
 Elia, I., 144
 Engle, R. A., 343
 Etchemendy, J., 289
 Euler, L., 314
 Evan, R., 314, 320

F

Faure, G., 74, 80, 83
 Felty, A., 19, 20
 Ferrari, P., 359
 Ferreirim, I. M. A., 96
 Ferrero, E., 359
 Ferri, F., 177
 Fietzke, A., 74
 Filliâtre, J.-C., 74, 86
 Fischbein, E., 186
 Flajolet, P., 125
 Fontaine, P., 74, 78
 Font, L., 140, 163
 Forest, S., 74, 86
 Fortuny, J. M., 140, 240, 361
 Foster, C., 221
 Fournet, C., 86
 Frege, G., 119
 Fugard, A., 66
 Fugard, A. J. B., 68
 Fujita, T., 173, 291, 292, 296, 298, 300, 308
 Fuller, E., 308
 Fu, Z., 74

G

Gagatsis, A., 145
 Gagnon, M., 140, 154–156, 163
 Galperin, G. A., 255
 Ganesalingam, M., 21, 24

Ganzinger, H., 74
 Gao, X. S., 238–242, 245–247, 249
 García-Diez, M., 142
 Gardner, M., 261
 Garillot, F., 15, 85
 Garuti, R., 173, 177, 182, 183, 358, 359
 Geeraerts, L., 149
 Gelernter, H., 218, 355, 360
 Gentry, C., 86
 Gerdes, A., 326
 Giorgiutti, I., 141
 Gödel, K., 59, 65
 Godot, K., 121
 Goldstein, I., 354
 Gonthier, G., 15, 85
 González, G., 291
 Gordon, M., 73
 Gowers, T., 202, 205, 206
 Grabowski, A., 73, 352
 Graesser, A., 337
 Graham-Lengrand, S., 86
 Gravemeijer, K., 336
 Gravier, S., 121, 150, 153
 Gray, E., 150
 Greeno, J. G., 298
 Grégoire, B., 73, 74, 80, 81, 83
 Grenier, D., 117, 121
 Gribomont, P., 118, 119, 125, 126
 Grice, H. P., 29
 Grover, B. W., 295
 Gueudet, G., 117
 Gutiérrez, Á., 256, 335

H

Hadarean, L., 74
 Hadas, N., 176, 178
 Hähnle, R., 353
 Haken, W., 139, 154
 Hales, T. C., 15, 85
 Halmos, P. R., 220
 Hanna, G., 173, 176, 204, 231, 232, 237, 238, 256, 291, 357
 Hardy, G. H., 199, 200, 206, 207, 209, 210
 Harel, G., 173, 183, 256
 Harris, M., 200, 203, 204
 Harrison, J., 15, 73, 85, 357
 Hart, E. W., 117
 Hašek, R., 230
 Haspekian, M., 156
 Hauer, B., 359, 360
 Healy, L., 332, 335, 359
 Heeren, B., 326
 Heidigger, M., 208

Heinze, A., 294
 Henneman, W. H., 21
 Hennessy, S., 239
 Henningsen, M., 295
 Herbelin, H., 73
 Herbst, P. G., 173, 291, 335, 358
 Hershkowitz, R., 178, 337
 Hersh, R., 205
 Hetzl, S., 354
 Heule, M. J. H., 74, 75, 85, 350
 Hilbert, D., 60
 Hoang, T. L., 15
 Hoare, C. A. R., 125
 Hodges, R., 21
 Hoelzl, R., 359
 Hoffmann, D. L., 142
 Hohenwarter, M., 216, 218, 219, 221, 232, 238, 240, 250
 Holland-Minkley, A. M., 19
 Hollings, C., 199
 Hölzl, R., 183
 Hopcroft, J., 122, 124, 126
 Houdebine, J., 141
 Howell, J., 86
 Howson, A. G., 116, 117, 231
 Hoyles, C., 239, 331, 335, 359
 Hritcu, C., 74, 86
 Hsieh, F. J., 237
 Huet, G. P., 73
 Hughes, E. K., 343
 Humayoun, M., 22
 Hurd, J., 74

I

Il'jashenko, J. S., 269
 Inglis, M., 290
 Ingold, T., 198, 208, 209
 Iwanaga, Y., 296

J

Jackiw, N. R., 219
 Jahnke, H. N., 173
 Jamnik, M., 65, 66
 Janičić, P., 218, 219, 238–242, 245–247, 250
 Jaworski, B., 220, 221
 Jebelean, T., 20
 Jenkins, A., 201, 202, 210
 Jeuring, J., 326
 Jiang, J., 239
 Johnson, H. L., 343
 Jones, K., 173, 239, 291, 292, 295, 296, 298, 308, 309, 311, 332, 334, 338, 342–344, 358

Jones, P. L., 220
 Jourdan, J.-H., 86
 Jovanovic, D., 74

K

Kahane, J. P., 116, 117
 Kaliszzyk, C., 74
 Kapur, D., 218, 240
 Katz, G., 74, 81, 85
 Kaufmann, M., 74
 Keller, A. G., 148
 Keller, C., 74, 80, 81, 83, 85, 86
 Keller, O., 142
 King, T., 74
 Kirshner, D., 314
 Knipping, C., 173
 Knott, A., 28, 29
 Knuth, E. J., 331
 Köcher, N., 313, 314, 318–320
 Koedinger, K. R., 294, 354, 355
 Koepke, P., 22
 Kohlhase, M., 86
 Kohlweiss, M., 86
 Koike, N., 332
 Komatsu, K., 291, 332, 334, 335, 338, 342–344
 Konev, B., 74–76
 Kong, S., 73, 278
 Kornilowicz, A., 73, 352
 Kortenkamp, U., 219, 233, 238, 358, 359
 Kovács, L., 20
 Kovács, Z., 163, 216, 218–222, 224, 228, 230, 232, 233, 238, 240, 250, 359, 360
 Ko, Y. Y., 331
 Krause, E. F., 228
 Kreitz, C., 73
 Krieger, M. H., 152, 157
 Küchemann, D., 331
 Kuhlwein, D., 22
 Kullmann, O., 74, 75, 85, 350
 Kumar, R., 74
 Kutsia, T., 20
 Kutzler, B., 218
 Kuzniak, A., 140, 144, 145, 157

L

Laborde, C., 141, 147, 158, 179, 332
 Laborde, J. M., 178, 179, 219, 256, 359
 Lagrange, J. B., 145
 Lakatos, I., 5, 60–62, 64, 69, 207, 292, 331
 Lane, L., 201, 202, 205
 Laporte, V., 86
 Larsen, S., 331

- Lascarides, A., 21
 Laval, D., 145, 157
 Lawrie, C., 256
 Leduc, N., 154–156, 361
 Lehman, B., 337
 Le Hoang, T., 85
 Leibniz, G., 314
 Leiserson, C. E., 125
 Leiß, D., 147
 Leitsch, A., 354
 Lemut, E., 173, 182, 183, 358, 359
 Leron, U., 186
 Leroy, X., 86
 Lescuyer, S., 74
 Leung, A. Y. L., 174, 178, 183, 186, 188, 190, 332, 358
 Levenson, E., 314, 320
 Levi-Strauss, C., 208
 Lewis, R. Y., 278
 Li, H., 240
 Lin, F. L., 237, 294
 Lisitsa, A., 74–76
 Lithner, J., 313, 322
 Lopez-Real, F., 183, 186
 Lorigo, L., 73
 Loustaunau, P., 316
 Lovász, L., 117
 Luengo, V., 292, 297, 308, 359–361
 Łukasiewicz, J., 99
 Lund, T., 320
 Lynce, I., 74
- M**
- Mackenzie, D., 73, 205, 208
 Mahboubi, A., 73
 Malik, S., 74
 Mancosu, P., 15
 Manizade, A. G., 332
 Mannila, L., 328
 Mante, M., 183
 Maréchal, A., 86
 Marek, V. W., 74, 75, 85, 350
 Margolinas, C., 158
 Marhajan, Y., 74
 Marić, M., 250
 Mariotti, M. A., 173–175, 177, 178, 180, 181, 183–185, 187, 188, 190, 332, 359
 Marradez, R., 335
 Martinovic, D., 332
 Martin, T. S., 157, 291
 Martin, U., 198, 199, 207
 Maschietto, M., 147
 Mason J., 204
- Matsuda, N., 356, 361
 Matuszewski, R., 73, 352
 Mawer, R. F., 46
 Maxwell, E. A., 317
 McCrone, S. S., 291
 McCune, W., 91
 Meavilla Seguí, V., 240
 Meavilla, V., 140, 158, 159, 161
 Meksout, A., 74, 81, 85
 Mejia-Ramos, J. P., 308
 Meyer, A., 125
 Michael-Chrysanthou, P., 140
 Miller, D., 19, 20, 86
 Milton, J. A., 142
 Miyazaki, M., 173, 291, 292, 296, 298, 300, 308
 Modeste, S., 117, 125, 128, 150, 153, 154
 Moggi, E., 102
 Monks, K. G., 290
 Moore, J. S., 22, 74
 Morris, C. W., 119
 Morselli, F., 173
 Motwani, R., 122, 124, 126
 Moutet, L., 145, 157
 Moyer-Packenham, P. S., 332
 Murray-Rust, D., 209
- N**
- Nakagawa, K., 20
 Narboux, J., 238, 240, 242, 246, 247
 Naumowicz, A., 73, 352
 Neale, V., 206, 209, 210
 Nelsen, R. B., 65, 66, 141, 146, 239
 Netz, W., 148
 Nevins, A. J., 354
 Newell, A., 355
 Newman, M. H. A., 320
 Nicaud, J. F., 313, 326
 Nieuwenhuis, R., 75
 Noel, W., 148
 Norell, U., 73
 Northrop, E. P., 317, 320
 Noss, R., 359
- O**
- Ohtani, M., 295, 343
 Oliveras, A., 73
 Olivero, F., 183, 336
 Oller, A. M., 140, 158, 159, 161
 Oller Marcén, A. M., 240
 Osana, H. P., 332–334, 336, 337, 342–344
 Ouvrier-Bufferet, C., 117, 150, 153
 Owen, E., 46

Owens, K., 358
 Owens, T. M., 96
 Owre, S., 73

P

Palm, T., 313, 322
 Paneque, J., 240
 Panero, M., 00
 Paola, D., 183, 336
 Parisse, B., 219
 Parno, B., 86
 Paskevich, A., 22, 74, 86
 Paulson, L. C., 74, 23
 Pavlović, V., 238, 242, 245
 Payan, C., 117, 121
 Pedemonte, B., 183, 358
 Pegg, J., 256
 Pekrun, R., 337
 Peled, I., 331
 Pelletier, R., 294, 355
 Pell, J., 328
 Peltier, N., 361
 Pepin, B., 295, 309, 343
 Perrin, D., 116
 Pesty, S., 360
 Petrović, I., 218, 219, 238, 240, 250
 Pettitt, P. B., 142
 Pichardie, D., 78, 79, 86
 Pike, W. G., 142
 Pollack, R., 116
 Polya, G., 164, 165, 205, 313, 326, 358
 Polymath, D. H. J., 198, 205–207, 209, 210
 Potari, D., 165
 Prank, R., 326
 Prediger, S., 165, 332, 342
 Presmeg, N. C., 335
 Protzenko, J., 86
 Prusak, N., 337
 Puertas, E., 361
 Puitg, F., 361
 Pulte, H., 173
 Pye, D., 208

Q

Quaresma, P., 238–242, 246, 247, 250
 Quine, W. V., 119, 120

R

Rabardel, P., 146, 147, 160
 Rabe, F., 86
 Raffalli, C., 22
 Rahn, J. H., 328
 Ramsden, P., 322, 323
 Ranise, S., 77

Rastogi, A., 74, 86
 Raykova, M., 86
 Recio, T., 140, 145, 154, 163, 216, 218, 219,
 221, 230, 232, 238, 240, 250, 359, 360
 Reid, D. A., 173
 Reiser, B. J., 294
 Reis, G., 354
 Reiss, K., 294
 Reiter, E., 27
 Reynolds, A. J., 74, 81, 85
 Reynolds, J. C., 128
 Rhoads, K., 308
 Riazanov, A., 74
 Ribbens, D., 118, 119, 125, 126
 Ricciotti, W., 73
 Rice, A., 199
 Richard, P. R., 140, 144–147, 154–156, 158,
 163, 230, 240, 361
 Richter-Gebert, J. R., 238, 359
 Rivest, R. L., 125
 Roberts, S., 200, 202, 203
 Robinson, A., 60, 197
 Robutti, O., 174, 183, 336
 Roy, M. F., 116
 Rushby, J. M., 73
 Ruthven, K., 165, 239
 Ryle, G., 200, 206, 208–210

S

Saillard, R., 86, 354
 Sakamaki, A., 332
 Samkoff, A., 308
 Sangwin, C. J., 147, 313–315, 317–323
 Santos, V., 250
 Scali, E., 359
 Schiffler, K., 228
 Schneider-Kamp, P., 76
 Schoenfeld, A. H., 178, 294
 Schröder, B., 22
 Schulz, S., 74
 Schumann, H., 256, 268, 270
 Schwarz, B. B., 178, 337
 Sedgewick, R., 125
 Selden, A., 182
 Selden, J., 182
 Semadeni, Z., 333
 Semliakov, A. N., 255
 Sethi, R., 122
 Sewell, W. H., 201
 Shankar, N., 73, 86
 Shaw, 355
 Sherbert, D. R., 158
 Sherin, B., 294
 Shoenfield, J. R., 64

Sidoli, N., 238
 Sieg, W., 289
 Sierpinska, A., 177
 Silva, J. P. M., 74
 Simon, H., 355
 Simon, M. A., 183
 Sinaceur, H., 116, 121
 Sinclair, N., 174, 358
 Smith, M. S., 343
 Sólyom-Gecse, C., 230, 238
 Soury-Lavergne, S., 361
 Sowder, L., 173, 183
 Standish, C. D., 142
 Stefferud, E., 355
 Stein, C., 125
 Stein, M. K., 295, 343
 Stevenson, I., 178, 332
 Stojanović, S., 238, 242, 245
 Strässer, R., 178
 Straubing, H., 126
 Stump, A., 77, 78
 Stylianides, A. J., 173, 179, 332, 343
 Stylianides, G. J., 179, 332, 343
 Suda, M., 74
 Sullivan, M. C., 141
 Su, Z., 239, 291, 359
 Swamy, N., 74, 86
 Sweller, J., 46

T

Tabachnikov, S., 255
 Takeuti, G., 353
 Tall, D., 150
 Tanguay, D., 144, 145, 149, 157
 Tanswell, F., 204, 208
 Tao, T., 203–205, 207, 210
 Tarski, A., 99, 119–121
 Tassi, E., 73
 Tessier-Baillargeon, M., 154–156
 Théry, L., 74, 78, 80, 83
 The Theorema Working Group., 230
 Thomas, R., 202, 210
 Thomas, W., 126, 127
 Thompson, D. R., 190
 Tinelli, C., 73, 74, 77, 81, 85
 Tirosch, D., 314, 320
 Tomašević, J., 242
 Tošić, D., 242
 Trahan, A., 152, 153, 162, 167
 Trilling, L., 361
 Trouche, L., 117, 146
 Trybulec, A., 22

Tseitin, G., 79
 Tsujiyama, Y., 332
 Turnow, E., 198

U

Ufer, S., 294
 Ullman, J. D., 122, 124, 126
 Urban, J., 74, 357

V

Vajda, R., 230
 Valcarce, J. L., 219
 van Doorn, F., 73, 278
 van Maaren, H., 74
 VanÚ, K., 356, 361
 Vélez, M. P., 140, 154, 163, 221, 230, 238
 Vélez, P., 359, 360
 Velleman, D., 289
 Venant, F., 152, 178
 Vergnaud, G., 128
 Vershinin, K., 22
 Villani, C., 200
 Villiers, M., 141
 Vitrac, B., 148
 Vivier, L., 145
 von Raumer, J., 73, 278
 Voronkov, A., 60, 74
 Vujošević-Janičić, M., 242

W

Wallin, S., 328
 Walsh, T., 74
 Wang, D., 240
 Wang, K., 239, 291, 359
 Ward, M. R., 46
 Watson, A., 295, 343
 Weber, K., 292, 293, 307, 308, 332, 358, 360
 Weber, T., 74
 Weidenbach, C., 74
 Weil, P., 126
 Weiss, M., 358
 Weller, D., 354
 Werner, B., 74, 80, 83
 Wiedijk, F., 357
 Wiles, A., 197, 202, 209, 210
 Wilson, B., 231
 Winker, S., 91
 Winter, H., 256
 Wischniewski, P., 74
 Wittgenstein, L., 119, 120
 Wolper, P., 118, 119, 125, 126
 Wu, W., 239, 240

YYackel, E., [356](#)Yerushalmy, M., [335](#)Ye, Z., [238](#), [240–242](#), [246](#), [249](#)Yost, G., [291](#), [354](#)**Z**Zach, R., [59](#), [60](#)Zahradnik, J., [230](#)Zandieh, M., [331](#)Zaslavsky, O., [331](#)Zazkis, R., [337](#)Zeitz, P., [205](#)Zhang, J. Z., [238–240](#), [245–247](#)Zhao, X., [22](#)Zilhão, J., [142](#)

Subject Index

A

ABC-conjecture, 203
Advanced geometry tutor, 361
AgentGeom, 361
Algebraic Geometry, 215, 218, 228
Algorithmic, 117–119, 123–125, 127–129, 131–134, 136, 150–157, 161, 162, 164, 166, 167, 232
Algorithms, 6, 59, 73, 74, 76, 86, 115–118, 122–125, 127–135, 140, 143, 145, 150–154, 157, 161, 164, 215, 220, 221, 224, 231, 232, 242, 297
Analogy, 200, 202, 256, 258, 259, 358
Analysis, 24, 54, 66, 70, 91, 94, 122, 125, 128, 129, 132–136, 145, 159, 175, 176, 178, 179, 184, 185, 191, 200, 217, 218, 256, 279, 295, 301, 302, 306, 308, 309, 318, 319, 322, 327, 338, 344, 354
Android/iOS math apps
 Edukera, 218
 Euclidean, 218
 mathway, 218
 photomath, 218
Angle project, 354
Antiprism
 n-gonal antiprism, 256
Argumentation, 8, 176, 183, 190, 322, 323, 328, 356–359
Artificial mathematician, 362
ATP/automated theorem proving/automated provers, 14, 60, 65, 74, 75, 85, 86, 238, 242, 349–362
Automata, 122, 124, 126, 127, 131
Automated deduction
 automated deduction in geometry, 218, 219

Automated proof search, 91
Automatic discovery, 217, 231
Automatic proving/automatic theorem prover
 geometry automated theorem prover, 237–239, 250
Automation in combinatorics, 75, 85

B

Baghera, 361
Billiards
 Elementary billiards, 257
Binary operation, 20, 92, 96, 284
Bricolage, 208

C

Cabri 3D, 256, 257
Cabri-Euclide, 297, 359, 361
Cabri Geometry/Cabri-géomètre, 219, 359
Case discrimination, 256
Catalan solids, 271
Causality, 160, 185, 256
Charisma, 203, 204
Cinderella, 219, 238, 359
Complexity, 27, 124, 125, 129, 130, 132, 133, 136, 154, 157, 162, 177, 191, 192, 200, 239, 240, 349, 357, 358, 360
Composition, 36, 51, 122, 125, 126, 208, 256, 286
Computer algebra, 215, 219, 224, 350
Computer algebra system
 Giac, 219
Computer science, 3, 5, 6, 60, 76, 85, 86, 115–119, 121, 122, 124–126, 128, 130, 132, 135, 136, 154, 278, 281, 282, 287, 289, 290

- Concept, 6, 21, 70, 102, 120, 125, 128–130, 132, 139–141, 144, 145, 157, 159, 163, 165, 203, 206–208, 219, 220, 239, 280, 284, 287, 331, 333–336, 351, 352, 359
 Conception, 119, 128–131, 133–135, 145, 158, 284, 357, 360
 Conditionality, 182, 183, 185
 Conjecture, 3, 5, 7, 15, 62, 65, 68, 74–77, 85, 86, 91, 92, 94–98, 100, 101, 103, 107, 116, 142, 154, 162, 167, 173, 182–185, 187, 191, 199, 203, 206, 207, 216, 217, 220–223, 226, 237–241, 243, 245, 249, 250, 256, 331, 333, 335, 336, 350, 351, 358, 359
 Construction, 6, 14, 26, 50, 55, 56, 127, 136, 143, 147, 149, 174, 175, 178–186, 188–192, 201, 202, 209, 215, 216, 223, 224, 237–243, 246, 249, 250, 256, 259, 260, 265, 271, 279, 280, 292, 294, 297, 309, 334, 336–338, 355, 356, 359, 360, 362
 Continuation monad, 102
 Constructive omega rule, 64–66, 69
 Contradictions, 55, 59, 79, 93–95, 131, 156, 185, 186, 279, 289, 314, 320, 333, 334, 337, 340
 Convex hull, 256, 264, 268, 269, 272
 Correctness, 14, 15, 22, 60, 61, 66, 69, 74, 76, 80, 84–86, 123–126, 129, 132, 133, 179, 180, 278, 308, 315–317, 320, 323, 324, 351, 360
 Counterexample, 5, 8, 95, 101, 121, 207, 223, 239, 331, 333, 335–340, 342, 343, 359, 361
 Craft, 6, 197, 198, 202, 205, 208, 210
 Cube, 61, 63, 64, 169, 256, 257, 259, 260, 262, 264, 265, 267
 Cuboid, 256, 262
 Cut rule, 353
D
 3D dynamic geometry systems, 255–257
 3D hexagon
 Equiangular 3D hexagon, 260
 Equilateral 3D hexagon, 261, 267
 Point symmetric 3D hexagon, 266, 269
 Regular 3D hexagon, 267, 268
 Self-intersecting 3D hexagon, 268
 Symmetric 3D hexagon, 270
 3D kite, 271
 3D parallelogram, 271
 3D polygon, 61, 271
 3D quadrangle, 257, 270, 271
 3D rhomb, 270
 De Bruijn criterion, 351, 352
 Decompilation of compiled code, 16
 Deduction, 19, 26, 33, 38–40, 45, 46, 49, 55, 65, 78, 121, 122, 125, 126, 158, 241, 250, 251, 279, 280, 282–284, 290, 297, 351, 354
 Deltoid, 269
 Diagram, 36, 66, 68, 69, 145, 146, 148, 149, 157, 159, 162, 216, 222, 297, 332, 334, 335, 337–339, 342, 343, 354, 355, 360, 361
 Diagrammatic reasoning, 141
 Didactic heuristic, 360
 Didactical/didactics, 6, 115, 116, 118, 125, 127, 128, 135, 136, 144, 163, 166
 Double pyramid
 Three-sided double pyramid, 268, 269
 Dragging test, 179, 180, 336
 Dynamic geometry, 4, 7, 9, 116, 146, 147, 149, 155, 157, 160, 162, 163, 166, 174, 175, 183, 215–219, 231, 232, 237, 240, 250, 256, 291, 327, 332, 349, 358
 Dynamic Geometry Environment (DGE)/
 Dynamic Geometry Systems (DGS), 174, 176, 178–192, 216, 237, 239, 241, 250, 332–334, 336–340, 342, 344, 359–361
E
 Effectiveness, 118, 154, 190
 Elementary geometry
 apollonius circle, 227
 geometric mean theorem, 181, 230
 Pythagoras' theorem, 230
 right triangle altitude theorem, 230
 Thales' circle theorem, 228
 triangle inequality, 228, 257
 Empirical validation, 356, 358, 361
 Enrichment, 165
 Epistemic value, 148, 176, 357
 Epistemological revolution, 356
 Epistemology, 115, 117
 Ethnography, 204
 Explanation, 7, 15, 75, 80, 110, 188, 217, 237, 238, 241, 250, 256, 305, 336, 357, 362
F
 Feedback, 8, 86, 233, 281, 287, 291, 295–299, 303, 304, 308, 314, 315, 323, 324, 352, 354, 355, 359, 360

Field of experience, 359
 Formalism, 59, 60, 126, 132, 278, 354, 357
 Formalization, 5, 15, 59, 152, 154, 357

G

Generalization, 36, 152, 166–168, 228, 230, 256, 262, 265
 General Problem Solver, 355
 Genius, 6, 197, 202–205, 210
 GeoGebra
 Automated Reasoning Tools (ART), 221, 224, 231, 233
 envelope command, 230
 LocusEquation command, 217, 222, 225–229
 Relation command, 216, 223, 224, 226, 227
 Prove command, 230, 240
 ProveDetails command, 230
 Geometer's Sketchpad, 219, 291
 Geometer Supposer, 219
 Geometric reasoning
 algorithm, 217
 conjecture, 216
 theorem, 217
 Geometry
 geometrical, 178–181
 geometric construction, 175, 218, 237, 239, 240, 242, 243, 246
 geometric locus, 162, 222, 227
 spatial geometry, 255, 256
 Geometry machine, 355
 Geometry tutor, 354, 355, 361
 Google Play, 218

H

Heuristic
 Heuristic argumentation, 358, 359
 Hexahedron, 265, 268–270
 Heyting algebra, 96, 100–102, 110
 Hilbert's program, 59, 60, 63, 64, 66, 69, 70
 Hoop, 91, 96–103, 107, 110
 Human-oriented
 human-like output, 22, 25, 29, 31
 human-like write-ups, 24
 human-oriented theorem proving, 21

I

Implicit locus, 221, 222, 228
 Indirect proof, 174, 185–187, 190–192, 292
 Induction, 65, 68, 69, 74, 100, 129, 131, 154, 256, 280, 287, 313, 327, 351, 356, 358

Information and Communication Technology (ICT), 6, 173, 174
 Interactive theorem prover, 5, 14, 15, 73, 278, 281, 284, 288

J

Johnson solids, 271
 Justification, 18, 27, 45, 78, 93, 94, 133, 244, 313, 320, 323

K

Kite, 268, 269, 271

L

Language, 7, 19, 20, 27, 59, 115, 118, 120–124, 126, 127, 131, 132, 134, 141, 165, 190, 205, 237, 238, 240–242, 245, 246, 248, 250, 277–282, 284, 286, 288, 289, 297, 314, 352, 354–357
 Lean, 8, 278–289, 362
 Learning
 discovery learning, 7, 9, 143, 154, 205, 218, 220, 221, 237
 Learning and teaching of geometry
 computer-mediated thinking, 218, 220
 creativity spiral, 217, 220, 226
 geometry calculator, 231–233
 mentor, 204, 217, 221
 teaching triad, 220
 technology-mediated paradigm, 217
 Logic, 7, 8, 23, 30, 60, 73, 75, 85, 86, 91, 92, 96, 99, 100, 115, 116, 118, 119, 121, 122, 125–128, 131, 136, 145, 149, 155, 156, 163, 176, 179, 218, 242, 245, 277–284, 286, 288–290, 327, 350–353, 355, 362
 Logical framework, 127, 352, 354
 Logic theorist, 355

M

Mace4, 5, 91, 92, 95–98, 100, 101, 110
 Manipulatives, 8, 332–337, 342–344
 Mathematical concept, 157, 280, 333, 334
 Mathematical practice, 116, 198, 204, 205, 208, 331, 333
 Mathematical virtues, 204
 Mathematical working space, 140, 144, 157, 162, 361
 Mathematics, 3–6, 8, 9, 15, 23, 27, 59–61, 68, 76, 85, 100, 115–118, 121–123, 128, 130, 131, 134–136, 140, 143, 144,

- 147–149, 153, 154, 156–159, 163–166, 173, 176, 177, 181, 191, 197–200, 202–205, 207, 208, 210, 215, 217, 220, 228, 231, 233, 237, 238, 256, 278, 280, 282, 284–292, 294, 295, 300, 301, 313, 314, 318–320, 323, 327, 328, 331, 332, 336, 343, 349–352, 356, 357, 359–362
- Meshwork, 209
- Metaphors of space and landscape, 198
- Method
 - heuristic method, 163, 265
- Metric space, 13, 25, 30, 32, 47
- Microworld, 359, 361
- Mindset, 204, 205, 207
- Model, 21, 24, 26, 27, 34, 41, 43, 75, 77, 79, 85, 91, 95, 103, 110, 120–123, 125–129, 131–134, 141–148, 151–153, 156–160, 162, 163, 173, 186, 188, 207, 237, 239, 277, 281, 317, 319, 327, 354, 355, 358, 359, 361
- Modelling, 25, 26, 126, 139, 143, 145, 147, 151, 157, 158, 163, 164
- Model-theoretic, 110, 119–121, 127
- N**
- Natural language, 7, 19, 21, 27, 29, 123, 141, 165, 240, 242, 245, 246, 250, 281, 354, 357
- Networking theoretical approaches, 342, 344
- 2n-gon
 - Regular 2n-gon, 255
 - Spatial 2n-gon, 256
- Non-example, 335
- O**
- Ontic value, 357
- Orbit, 267
- Ordinal sum, 99, 100
- P**
- PACT Geometry Tutor, 361
- Parallelogram, 146, 167, 184, 187, 188, 190, 243, 260, 270, 333–336, 338, 340, 341
- Pedagogical support, 231, 233, 333, 336, 342, 343
- Perimeter
 - minimal perimeter, 7, 256, 268, 270
- Platonic solids, 61, 271
- Pocrim, 98, 99
- Polygon
 - convex polygon, 168, 255
 - hexagonal polygon, 257
 - inscribed polygon, 256
 - Petrie polygon, 256
 - point symmetric polygon, 262, 268–270
 - spatial polygon, 269
- Polyhedra
 - convex polyhedra, 7, 255, 271
- Polyhedron
 - 2n-faced polyhedron, 255, 256
- Polymath, 6, 198, 205–207, 209, 210
- Pragmatics, 40, 119, 123, 136, 154, 280, 355, 361
- Programming, 83, 117, 118, 122, 123, 132–135, 143, 150, 154, 220, 221, 226, 278
- Proof
 - proof certificate, 308
 - proofs with a natural language rendering, 7, 242
 - proofs with a visual rendering, 7, 246
 - proof theory, 353, 354
 - proof tutor, 289, 354, 356, 359
 - rigorous proof, 5, 60, 63, 152, 157, 224
 - schematic proof, 60, 63–70, 352
- Properties, 158, 160, 163, 167, 180, 184, 185, 190, 237–241, 243, 250, 257, 258, 280, 286, 300, 302, 305, 307, 308, 315, 316, 324, 327, 350, 357, 359
- Prover9, 5, 91–97, 100–104, 107, 110
- Q**
- QED-Tutrix, 140, 155, 156, 163, 164, 361
- Quadrilateral, 64, 184, 187, 188, 190, 300, 333–336, 338–341
- R**
- Reasoning
 - reasoning backwards, 40, 42, 46
 - reasoning forwards, 24, 28, 35, 38, 40, 46, 47
 - reasoning tactics, 42, 43, 45–47
- Reflection, 27, 135, 148, 173, 201, 204, 246, 256, 257, 259, 260, 264
- Refutation, 3, 245, 291, 332, 334, 343
- Reinforcement, 204, 238, 240, 256, 284, 335, 357
- Representations, 37, 128, 129, 131–136, 139–141, 145–147, 149, 159, 160, 162–164, 188, 203, 210, 242, 282, 284, 297, 316, 332–335, 354–358, 360
- Restructuring, 256, 355

Reversibility, 256, 265, 321
 Rhombus, 339
 Robbins conjecture, 91

S

Semantics, 99, 118, 119, 122, 123, 125, 132, 135, 136, 158, 176, 250, 279, 289, 351, 359
 Semilattice, 91, 92, 94–97
 Semiotic potential, 174–176, 178–183, 185, 190, 191
 Sequent calculus, 353
 Skill, 4, 6, 66, 136, 173, 198, 202, 204, 205, 207–210, 221, 290, 350, 355
 Social machine, 198, 206, 209
 Space
 Euclidean space, 255
 virtual space, 257
 Specification, 135, 256, 278, 287, 318
 Square, 122, 157, 168, 187, 188, 233, 257–261, 284, 339
 Structural induction, 126
 Students' perceptions, 333, 337
 Symmetry, 24, 104, 116, 203, 266, 268, 269, 353
 Syntax, 8, 83, 92–94, 119, 120, 122, 123, 126, 135, 136, 222, 281, 284, 288, 289, 322, 352, 360
 Synthesis, 174, 256

T

Task-based interview, 332, 333, 337, 338, 342, 343
 Task design, 8, 294, 295, 301, 309, 332–337, 342–344

Technology Enhanced Learning (TEL), 355–359, 361

Termination, 124

Tetrahedra, 270

Theorem

 Cognitive unity of theorem, 358
 model of theorem, 361

Theorem proving/provers

 automatic theorem proving/provers, 4, 5, 7, 8, 15, 16, 231, 350
 interactive theorem proving/provers, 8, 14, 15, 33, 73, 279–281, 284

Theory of semiotic mediation, 6, 174, 175

Trajectory, 255, 257, 258, 260, 263, 264, 267

Trapezohedron, 268

Trapezoid, 269

Turing test, 362

Twin Primes conjecture, 206

V

Variation, 29, 159, 166, 184, 228, 256

Verification, 3, 7–9, 15, 24, 60, 66, 86, 110, 118, 127, 129, 132–134, 139, 152–154, 160, 168, 176, 217, 221, 232, 237, 238, 240, 242, 256, 278, 289, 322, 352, 353, 356, 361

Vertices, 61, 126, 131, 184, 189, 221, 222, 247, 256, 262, 263, 268–270, 333, 336

Virtual manipulative, 8, 332, 333, 335–337, 342, 343

W

Wayfarer, 198, 209