A Proposal for a Framework to Accompany Formal Methods Learning Tools (Short Paper)

Norbert Hundeshagen(✉) and Martin Lange

Theoretical Computer Science/Formal Methods, University of Kassel,
Wilhelmshöher Allee 71, 34121 Kassel, Germany
{hundeshagen,martin.lange}@uni-kassel.de

Abstract. We propose a simple concept framework to accompany learning tools in theoretical computer science/formal methods in order to ease their integration into existing courses, balancing out technical issues against light-weight didactical questions.

Keywords: Theoretical computer science · Didactics of informatics · Software learning tools

1 Learning Tools in Theoretical Computer Science

Theoretical computer science (TCS) courses are usually quite formal and mathematical, and therefore often perceived to belong to the hardest subjects that students face. Failure and drop-out rates have always been high [12] with little indication that these rates would go down without active intervention [10]. Solutions to this problem range from the deployment of classic didactical methods (e.g. [10,15,16]), the use of adapted generalised tools, for instance using theorem provers for teaching logic [9], and the design, implementation and evaluation of specialised learning tools for particular subjects or tasks, e.g. [2,7,11].

This paper is concerned with development in the last of these categories, specifically seen from the perspective of theoretical computer scientists with rather little expertise in the field of didactics. We observe a growing interest and activity in the development of learning tools, as witnessed for instance by multiple venues like FMTea, ThEdu, FOMEO, and growing acceptance rates of papers on learning tools at non-educational conferences, c.f. [1,2,8,14]. Yet, in order to maximise learning effects, the development of such software tools needs to be embedded into a larger framework which is guided not only by technical but equally by didactical considerations, specialised to situations and issues commonly found in learning.

Modern learning tools generally make use of advanced technology to produce content and interactive user feedback, so that they can be – up to a certain degree – described under the term *intelligent tutoring systems* [6]. Intelligence is

J. F. Ferreira et al. (Eds.): FMTea 2021, LNCS 13122, pp. 35–42, 2021.
https://doi.org/10.1007/978-3-030-91550-6_3

often achieved using sophisticated formal methods (FM) like equivalence checks, SAT/SMT solving, theorem proving, model checking, semi-decision procedures, machine learning. These impact on didactical consideration which are rather specific to the area of TCS/FM. It is reasonable to assume that they have been implemented with a clear didactical purpose in mind, but this is often done by experts in TCS/FM, not necessarily in didactics, as the development of precise and efficient feedback technology for learning tools naturally requires expertise in the underlying technology.

We propose to accompany TCS/FM learning tools with a simple, yet clearly spelled-out didactical concept broken down to a collection of basic questions. It balances technical considerations with didactical aspects including addressed competence levels and feedback systems, thus encompassing the typical learning cycle. Such a framework would make the tools' developers provide a key element for the integration of such learning tools into an existing course and would help to bridge the gap between expertise in FM and comprehensive didactical theories. Moreover, it aids the comparability amongst learning tools, not necessarily to single out a best one but mainly to be able to quickly judge which tool is most suitable for which teaching situation. This would also enhance the effectiveness of the FM Courses Database[1] for instance by providing lecturers with information on typical didactic considerations, as prerequisites needed to deploy such tools, didactic limitations in using them, how learning outcomes are foreseen to be achieved, etc.

We give two examples of learning tool descriptions according to this framework. We conclude with remarks on continuing conceptual work accompanying the development of learning tools.

2 The Proposed Didactic Framework

Information that is rather obvious and normally given, like what subarea of TCS/FM the tool belongs to, what implementation technology has been used etc., should be complemented in such a framework with didactical considerations in order to facilitate a smooth integration into a course. For this to work, the concept description should particularly address the following issues.

What are the technical requirements for using the tool? This may restrict the answers to several other questions posed here, as it sets some key parameters for the use of the tool. Requirements may refer to the need to create user accounts, the availability of webserver, a database, the runtime environment of a programming language, other software, or the need to be run on a particular operating systems or within an educational platform etc.

What are the technical capabilities of the tool? The answer should be specific to the field of TCS as certain technical capabilities may show negative learning effects in particular fields. For instance, generating automated feedback on

[1] https://fme-teaching.github.io/#fm-courses.

an undecidable problem may impede students' understanding of the concept of decidability. For teachers it is vital to know the characteristics of the underlying technology (testing, bounded search, etc.) in order to be able to put the feedback system into context in students' eyes.

Furthermore, knowing the limits of the implemented technology to create learning effects also helps to judge their limits in creating such effects. For instance, when feedback is given in the form of counterexamples, it is useful to know the characteristics of methods used to produce them, i.e. taken from a fixed set, minimality, generated intelligently to address the student's individual learning problem, etc. Answers give key insight into didactical considerations such as: does the feedback system support the acquisition of certain competences, can feedback be misinterpreted, is it possible to mould an incorrect solution that passes all tests, etc.

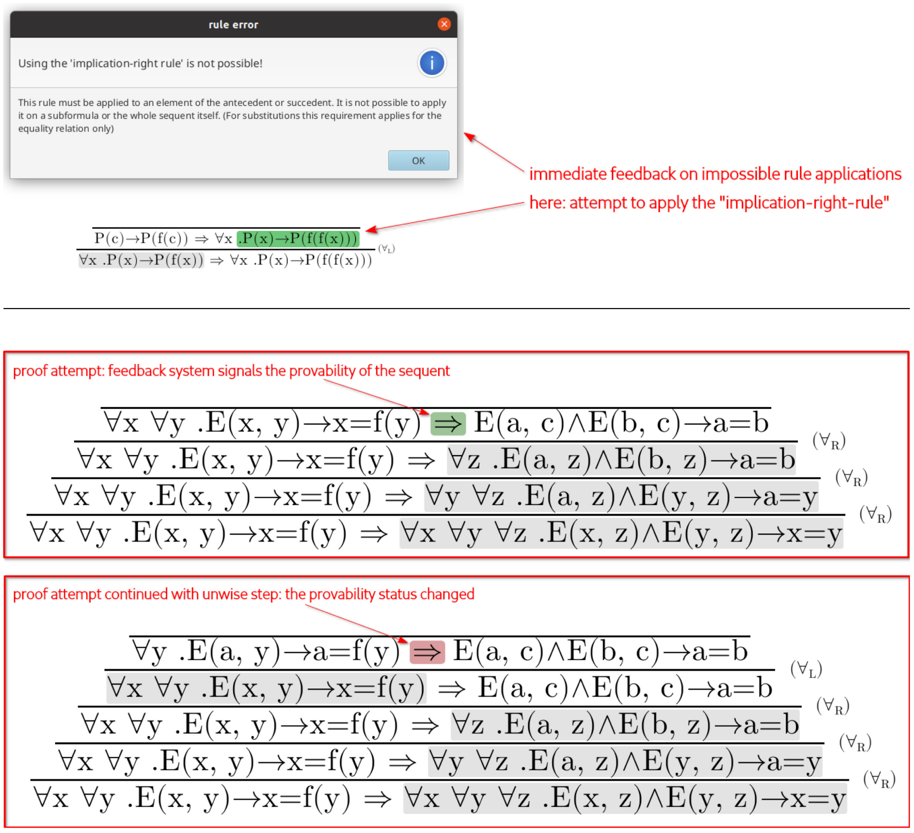
The same considerations in didactical impacts also apply to tools that generate questions or exercises automatically. Technical capabilities that make content persistent or movable like the ability to save solution attempts to disc, to send or submit them, etc. can also influence didactical considerations likewise.

What does the course need to provide in order for the tool to be used? Specifically, what competences are the students assumed to already have acquired before they can start using the tool, for instance familiarity with a particular formal concept or the ability to carry out particular formal tasks. Larger tools which accompany an entire course rather than just target a specific competence may presuppose nothing in this respect.

Where in a course is the tool intended to be used? The standard learning model for courses in TCS still builds on the presentation of material in lectures, a short period of self-study to deepen and review understanding, plus set exercises/tutorial sessions. A clear recommendation on where to place the use of this tool in the line of activities making up a course is helpful, whether it is merely to be used in a lecture to visualise certain concepts, whether students are supposed to work on it autonomously perhaps after receiving a certain amount of tuition, or in a guided way during tutorials, whether the homework exercises can be run through this tool, etc. Multi-purpose in this sense is of course imaginable.

What competences at what level does the tool address? The answer has to be specific; it is not enough to know that tool X "helps students to learn logic." Instead the answer needs to name specific levels of competence, perhaps linking to standard syllabi in the respective area.

Even when there is no established formulation to name such levels yet it should be possible to sketch such levels. Often the analysis and categorisation of typical students' mistakes in exercises gives rise to a hierarchy of competence levels in a particular area, one that is quickly understood by fellow teachers. Listing standard exercises which students are supposed to be able to master after successfully learning with the underlying tool also provides valueable information here. Moreover, it can be useful to consider formulations of general competence


rule error

Using the 'implication-right rule' is not possible!

This rule must be applied to an element of the antecedent or succedent. It is not possible to apply it on a subformula or the whole sequent itself. (For substitutions this requirement applies for the equality relation only)

OK

$$\dfrac{P(c)\to P(f(c)) \Rightarrow \forall x\, .P(x)\to P(f(f(x)))}{\forall x\, .P(x)\to P(f(x)) \Rightarrow \forall x\, .P(x)\to P(f(f(x)))}\,(\forall_L)$$

$$\dfrac{\dfrac{\dfrac{\forall x\ \forall y\ .E(x,y)\to x=f(y) \Rightarrow E(a,c)\wedge E(b,c)\to a=b}{\forall x\ \forall y\ .E(x,y)\to x=f(y) \Rightarrow \forall z\ .E(a,z)\wedge E(b,z)\to a=b}\,(\forall_R)}{\forall x\ \forall y\ .E(x,y)\to x=f(y) \Rightarrow \forall y\ \forall z\ .E(a,z)\wedge E(y,z)\to a=y}\,(\forall_R)}{\forall x\ \forall y\ .E(x,y)\to x=f(y) \Rightarrow \forall x\ \forall y\ \forall z\ .E(x,z)\wedge E(y,z)\to x=y}\,(\forall_R)$$

$$\dfrac{\dfrac{\dfrac{\dfrac{\forall y\ .E(a,y)\to a=f(y) \Rightarrow E(a,c)\wedge E(b,c)\to a=b}{\forall x\ \forall y\ .E(x,y)\to x=f(y) \Rightarrow E(a,c)\wedge E(b,c)\to a=b}\,(\forall_L)}{\forall x\ \forall y\ .E(x,y)\to x=f(y) \Rightarrow \forall z\ .E(a,z)\wedge E(b,z)\to a=b}\,(\forall_R)}{\forall x\ \forall y\ .E(x,y)\to x=f(y) \Rightarrow \forall y\ \forall z\ .E(a,z)\wedge E(y,z)\to a=y}\,(\forall_R)}{\forall x\ \forall y\ .E(x,y)\to x=f(y) \Rightarrow \forall x\ \forall y\ \forall z\ .E(x,z)\wedge E(y,z)\to x=y}\,(\forall_R)$$

immediate feedback on impossible rule applications
here: attempt to apply the "implication-right-rule"
proof attempt: feedback system signals the provability of the sequent
proof attempt continued with unwise step: the provability status changed

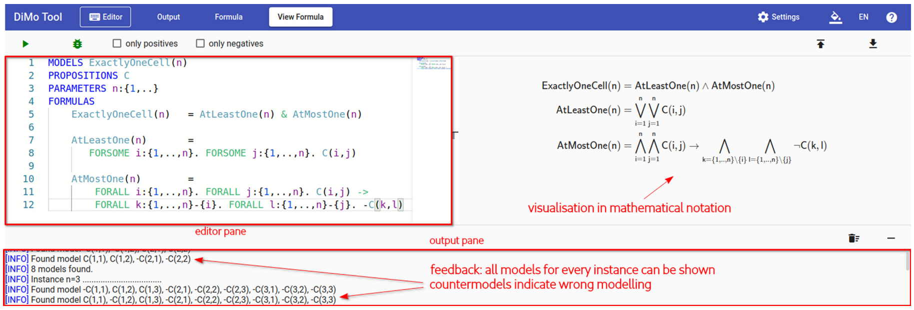
Fig. 1. The feedback system in the Sequent Calculus Trainer.

hierarchies for entire study programmes, e.g. [3] as well as heavy-handed didactical considerations on how to build competence models for theoretical computer science [13].

What learning model does the tool follow? An answer would typically refer to standard behaviouristic, constructivistic, and cognitivistic learning models. Users of those tools, i.e. teachers integrating them into their courses, may not be familiar with such complete theories. It is therefore helpful to refer to specific methods advocated in such theories. Especially present feedback systems should be linked to detailed explanations on how learning is initiated, e.g. in an error-driven way, by immediate feedback on every user interaction, through simple questionnaires on the content, through feedback-loops, etc.

3 Two Exemplary Instances of the Proposed Framework

We exemplify the proposal of an accompanying framework using two specific tools: the Sequent Calculus Trainer [5] and DiMo [8]. The first one is – as the


DiMo Tool
Editor
Output
Formula
View Formula
only positives
only negatives
Settings
EN
editor pane
output pane
visualisation in mathematical notation
feedback: all models for every instance can be shown
countermodels indicate wrong modelling

```
MODELS ExactlyOneCell(n)
PROPOSITIONS C
PARAMETERS n:{1,..}
FORMULAS
    ExactlyOneCell(n)   = AtLeastOne(n) & AtMostOne(n)

    AtLeastOne(n)       =
       FORSOME i:{1,..,n}. FORSOME j:{1,..,n}. C(i,j)

    AtMostOne(n)        =
        FORALL i:{1,..,n}. FORALL j:{1,..,n}. C(i,j) ->
        FORALL k:{1,..,n}-{i}. FORALL l:{1,..,n}-{j}. -C(k,l)
```

$$\mathrm{ExactlyOneCell}(n) = \mathrm{AtLeastOne}(n) \wedge \mathrm{AtMostOne}(n)$$

$$\mathrm{AtLeastOne}(n) = \bigvee_{i=1}^{n} \bigvee_{j=1}^{n} C(i,j)$$

$$\mathrm{AtMostOne}(n) = \bigwedge_{i=1}^{n} \bigwedge_{j=1}^{n} C(i,j) \rightarrow \bigwedge_{k=\{1,..,n\}\setminus\{i\}} \bigwedge_{l=\{1,..,n\}\setminus\{j\}} \neg C(k,l)$$

```
[INFO] Found model C(1,1), C(1,2), -C(2,1), -C(2,2)
[INFO] 8 models found.
[INFO] Instance n=3 ................................
[INFO] Found model -C(1,1), C(1,2), C(1,3), -C(2,1), -C(2,2), -C(2,3), -C(3,1), -C(3,2), -C(3,3)
[INFO] Found model C(1,1), -C(1,2), C(1,3), -C(2,1), -C(2,2), -C(2,3), -C(3,1), -C(3,2), -C(3,3)
```

Fig. 2. Main view of the DiMo web frontend with highlighted feedback system.

Table 1. Didactical concept description for the Sequent Calculus Trainer.

The Sequent Calculus Trainer	
target area	– logic in computer science – formal proof systems (sequent calculus) – not useful for other proof systems like resolution etc.
availability, technical requirements	– free software (BSD-3 license) – written in Java, needs JRE 8.1 or higher – download: https://www.uni-kassel.de/eecs/fmv/software/sequent-calculus-trainer
key technical capabilities	– presentation of formulas in mathematical notation – syntax highlighting – zoom-/scrollable proof display in tree shape – saving/loading/checking of proofs – validity checks for propositional and first-order logic (incomplete, uses SMT solver Z3) – limitations: feedback works best on shallow formulas and terms
provisions by the course	– introduction of the sequent calculus for first-order or propositional logic: proofs, rules, validity – not needed: soundness, completeness of the calculus
intended use	– (by teacher) presentation of proof construction, e.g. during lectures – (by student) (guided) solving of exercises on proving validity of formulas in sequent calculus during Tutorials and/or homework
learning model aspects	– immediate feedback for wrong use of concepts, see Fig. 1 – provision of hints to help students construct correct proofs – SMT solver checks possible next proof steps for feedback on successful use (traffic light system, see Fig. 1)
required competences	– familiarity with the syntax of first-order or prop. logic – helpful: ability to understand the meaning of formulas
addressed competences	– syntactically correct proving in a formal proof system – finding proofs for valid formulas – to a smaller degree: understanding the logical reason for invalidity/existence of counterexamples

Table 2. Didactical concept description for the DiMo tool.

DiMo	
target area	– logic in computer science, discrete modelling
availability, technical requirements	– web application hosted by the University of Kassel under https://dumbarton.fm.cs.uni-kassel.de – no registration required – no collection of user data
key technical capabilities	– IDE with syntax highlighting and auto-completion – presentation of formulas in mathematical notation, see Fig. 2 – saving and loading DiMo programs – satisfiability, validity, equivalence, model enumeration (uses SAT solver) – limitations: only integer parameter, computation timeout depending on formula size and/or instances of the parameters
provisions by the course	– syntax and semantics of propositional logic – definition of generalized logical operators $\bigwedge, \bigvee$ – short introduction of the core concepts of the DiMo-language (given in the manual)
intended use	– (by teacher) to exemplify semantical concepts and discrete modelling; for use as SAT solver interface – (by student) solving exercises on discrete modelling in tutorials and/or homeworks
learning model aspects	– immediate feedback on wrong syntax by syntax highlighting (see Fig. 2) and compiler messages – support of learning the connection between semantics and the modelled problem by output of formula models, see Fig. 2
required competences	– familiarity with the syntax of propositional logic
addressed competences	– understanding semantics of propositional logic – familiarity with satisfiability, validity, equivalence – using the above to solve "real-world"-problems – ability to form reduction to SAT

name suggests – a learning tool for formal proofs in First-Order Logic using Gentzen's sequent calculus which is, besides resolution and natural deduction, one of the proof calculi that is commonly taught in undergraduate courses on logic in computer science or presented in standard textbooks thereof, cf. [4].

The second one supports learning the competence of discrete modelling in propositional logic, which is essential in undergraduate studies in computer science.

The choice for these two tools to exemplify instances of the proposed framework is not made for any specific reason other than the fact that they are being developed by the Theoretical Computer Science/Formal Methods group at the University of Kassel. We would clearly be less competent to present the possibly unformulated didactical concepts underlining tools developed elsewhere.

We do not propose to use a specific format for such a framework at this point. Tool developers should of course have the freedom to choose how best to convey the information about the key didactical aspects of their tool. It is also not clear – certainly not at this point – whether there could be some format that is best suited for all purposes. Here we choose a semi-formal description in order to concisely present the key aspects in textual form.

Table 1 concisely lists information regarding the Sequent Calculus Trainer w.r.t. to the set of framework questions listed above. Similarly, this kind of information on the didactic use of DiMo is given in Table 2.

4 Conclusion

We propose to accompany the development of learning tools in formal methods by instances of a simple didactical framework that gives answers to some key questions about comparability, usability, usefulness and effectiveness of such tools to facilitate an easier integration into existing courses. The elements of this framework are given by key questions to be answered by the tools' developers.

We believe that the most effective learning tools will be created when knowledge and expertise from both formal methods and didactics is combined effectively in the design processes, and it seems like there is still potential to strengthen this kind of combination.

The framework proposed here should be simple enough so that theoretical computer scientists, in particular non-experts in didactics, are able to formulate basic properties pertaining to the didactic aspects surrounding their tools. In this way, didactics provides the means for transferability of formal methods learning tools between educational situations.

The idea that learning tools may be more transferable when shipped with spelled out didactical concepts is of course not restricted to the field of formal methods or theoretical computer science. Here we contained the field of interest in order to work on the basis of a clear and defined picture of recent developments and particular needs (like typical problems students face in this area) etc. This is not to say that other fields may not undergo similar considerations.

At last, the framework here is not considered to be completed. It should be discussed and engineered further under consideration of significant developments in the areas of didactics and formal methods. It also remains to be seen whether formalisation into a more stringent format would be beneficial.

References

1. Andersen, J.R., et al.: CAAL: concurrency workbench, Aalborg edition. In: Leucker, M., Rueda, C., Valencia, F.D. (eds.) ICTAC 2015. LNCS, vol. 9399, pp. 573–582. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-25150-9_33

2. D'Antoni, L., Helfrich, M., Kretinsky, J., Ramneantu, E., Weininger, M.: Automata tutor v3. In: Lahiri, S.K., Wang, C. (eds.) CAV 2020. LNCS, vol. 12225, pp. 3–14. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-53291-8_1
3. Desel, J., et al.: Empfehlungen für Bachelor- und Masterprogramme im Studienfach Informatik an Hochschulen. GI-Empfehlungen (2016)
4. Ebbinghaus, H.D., Flum, J., Thomas, W.: Mathematical Logic. Undergraduate Texts in Mathematics, 2nd edn. Springer, Berlin (1994). https://doi.org/10.1007/978-1-4757-2355-7
5. Ehle, A., Hundeshagen, N., Lange, M.: The sequent calculus trainer with automated reasoning - helping students to find proofs. In: Proceedings of 6th International Workshop on Theorem Proving Components for Educational Software, ThEdu 2017. EPTCS, vol. 267, pp. 19–37 (2017)
6. Freedman, R.: What is an intelligent tutoring system? Intelligence **11**(3), 15–16 (2000)
7. Geck, G., Ljulin, A., Peter, S., Schmidt, J., Vehlken, F., Zeume, T.: Introduction to ILTIS: an interactive, web-based system for teaching logic. In: Proceedings of 23rd Annual ACM Conference on Innovation and Technology in Computer Science Education, ITiCSE 2018, pp. 141–146. ACM (2018)
8. Hundeshagen, N., Lange, M., Siebert, G.: DiMo – discrete modelling using propositional logic. In: Li, C.-M., Manyà, F. (eds.) SAT 2021. LNCS, vol. 12831, pp. 242–250. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-80223-3_17
9. Knobelsdorf, M., Frede, C., Böhne, S., Kreitz, C.: Theorem provers as a learning tool in theory of computation. In: Proceedings of 2017 ACM Conference on International Computing Education Research, ICER 2017, pp. 83–92. ACM (2017)
10. Knobelsdorf, M., Kreitz, C., Böhne, S.: Teaching theoretical computer science using a cognitive apprenticeship approach. In: Proceedings of 45th ACM Technical Symposium on Computer Science Education, SIGCSE 2014. ACM (2014)
11. Rodger, S.H.: JFLAP: An Interactive Formal Languages and Automata Package. Jones and Bartlett (2006)
12. Ross, R., Grinder, M., Kim, S., Lutey, T.: Loving to learn theory: active learning modules for the theory of computing. SIGCSE Bull. **34**(1), 371–375 (2002)
13. Schlüter, K., Brinda, T.: Characteristics and dimensions of a competence model of theoretical computer science in secondary education. In: Proceedings of 13th Annual SIGCSE Conference on Innovation and Technology in Computer Science Education, ITiCSE 2008, p. 367. ACM (2008)
14. Schwarzentruber, F.: Hintikka's world: agents with higher-order knowledge. In: Proceedings of 27th International Joint Conference on A.I., IJCAI 2018, pp. 5859–5861 (2018)
15. Sigman, S.: Engaging students in formal language theory and theory of computation. In: Proceedings of 38th SIGCSE Technical Symposium on Computer Science Education, SIGCSE 2007, pp. 450–453. ACM (2007)
16. Verma, R.M.: A visual and interactive automata theory course emphasizing breadth of automata. In: Proceedings of 10th Annual SIGCSE Conference on Innovation and Technology in Computer Science Education, ITiCSE 2005, pp. 325–329. ACM (2005)