

Richard J. Lipton · Kenneth W. Regan

People, Problems, and Proofs

Essays from Gödel's Lost Letter: 2010

 Springer

People, Problems, and Proofs

Richard J. Lipton · Kenneth W. Regan

People, Problems, and Proofs

Essays from Gödel's Lost Letter: 2010

 Springer

Richard J. Lipton
College of Computing
School of Computer Science
Georgia Institute of Technology
Atlanta, GA, USA

Kenneth W. Regan
Dept. of Computer Sci. & Engineering
The State University of New York
Buffalo, NY, USA

ISBN 978-3-642-41421-3

ISBN 978-3-642-41422-0 (eBook)

DOI 10.1007/978-3-642-41422-0

Springer Heidelberg New York Dordrecht London

Library of Congress Control Number: 2013956437

© Springer-Verlag Berlin Heidelberg 2013

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. Exempted from this legal reservation are brief excerpts in connection with reviews or scholarly analysis or material supplied specifically for the purpose of being entered and executed on a computer system, for exclusive use by the purchaser of the work. Duplication of this publication or parts thereof is permitted only under the provisions of the Copyright Law of the Publisher's location, in its current version, and permission for use must always be obtained from Springer. Permissions for use may be obtained through RightsLink at the Copyright Clearance Center. Violations are liable to prosecution under the respective Copyright Law.

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.

While the advice and information in this book are believed to be true and accurate at the date of publication, neither the authors nor the editors nor the publisher can accept any legal responsibility for any errors or omissions that may be made. The publisher makes no warranty, express or implied, with respect to the material contained herein.

Printed on acid-free paper

Springer is part of Springer Science+Business Media (www.springer.com)

*To our dear children: Andrea, Jen, and Rob;
Alex and Rebecca*

Preface

People, problems, and proofs are the lifeblood of theoretical computer science, which we call *theory* for short. When you switch on any of the computing devices that have transformed our lives, you are thinking of applications, *apps* for short. But behind many applications there are clever algorithms, and for every worthwhile algorithm there is a problem that it solves and a proof that it works. Before this proof there was an open problem: can one create an efficient algorithm to solve the computational problem? If so, how? If not, why not? Finally behind these questions are people who are excited by fundamental issues about our computational world.

In this second book based on posts from the weblog *Gödel's Lost Letter and $P = NP$* (GLL), we tell stories of what motivates the people toward their problems and—in many cases—proofs. We begin, however, with a long chapter spanning several posts on a case that attacked the famous problem in our blog's title, but did not end with a valid proof. We follow with some chapters on the language of theory, on the nature of proofs, and on several important conjectures that are seeking proofs. Then we give a selection of other stories that we posted in the year 2010. They are not in chronological order.

Every post on our blog, and every chapter in this book, begins with the names of one or more people as featured subjects. The chapter may cover their work, and important ideas of theirs, or an important association to them in work by others. Then our second paragraph makes a short statement of the chapter's topic. On the blog we always begin that with the word “today,” but in this book our pitch invites the reader in other ways. Another invariant is that our posts and chapters never close—the “closing” is a section titled “Open Problems” with further questions to pursue. Our chapters here also have a “Notes and Links” section that includes all references that were hyperlinked in the post, and sometimes additional remarks and references.

GLL was started by me, Dick Lipton, in February 2009, assisted by my student Subruk Kalyanasundaram and some other Georgia Tech locals. That summer I contacted Ken Regan, whom I have known in the field since the 1980s and worked with in the 1990s, about the subjects of a few posts, featuring him in one. Ken joined my copy-editing rota, and his role became larger by summer 2010. By late 2010 he started co-authoring and writing posts fairly regularly, and I am glad to welcome him as a co-author of this collection. We have kept the usage of “I” referring to me as it was in the posts, except in the long first chapter and Chaps. [32](#) and [60](#) which

were written by him. Posts in which I referred to him retain the way his name was given then. We both hope you will enjoy this collection.

Atlanta, GA, USA
Buffalo, NY, USA
June 2013

Richard J. Lipton
Kenneth W. Regan

Contents

1	The Claimant, the Readers, and the Crowd	1
1.1	The News	1
1.2	The Paper and First Post	2
1.3	My Snap-Doubt and Comment	3
1.4	Comments and Comments and Comments	4
1.5	Back to the Future	6
1.6	The Group	8
1.7	Internet Non-resistance	9
1.8	Company...	11
1.9	... And a Crowd	12
1.10	The Issues Crystallize	13
1.11	The Falling Action	14
1.12	What Did We Learn?	16
1.13	Notes and Links	17
2	Kenneth Iverson: Notation and Thinking	19
2.1	Good Notation	20
2.2	Good Notation?	22
2.3	Open Problems	23
2.4	Notes and Links	23
3	Edmund Hillary: Proofs and Mountain Climbing	25
3.1	A Disclaimer	25
3.2	Climbing	25
3.3	Solving	26
3.4	Open Problems	27
3.5	Notes and Links	28
4	Leonardo da Vinci: Proofs as Art	29
4.1	Studying Great Art	29
4.2	Studying Great Proofs	30
4.3	Some Great Proofs	31
4.4	What Can One Learn from This Study?	31
4.5	Open Problems	33
4.6	Notes and Links	34

5	Michael Atiyah: The Role of Proof	35
5.1	Does Any Proof Matter?	36
5.2	Does a Proof of $P \neq NP$ Matter?	36
5.3	Open Problems	37
5.4	Notes and Links	37
6	Subhash Khot: Unique Games Conjecture	39
6.1	Act I: The Unique Games Conjecture	39
6.2	Act II: The Conjecture's Applications	41
6.3	Act III: Is It True?	41
6.4	A Comment on Expanders	42
6.5	Open Problems	43
6.6	Notes and Links	43
7	Arno van den Essen: An Amazing Conjecture	45
7.1	The Jacobian Conjecture	45
7.2	JC Approaches	47
7.3	Amazing Conjectures	48
7.4	Open Problems	49
7.5	Notes and Links	50
8	Richard Hamilton: Group Efforts	51
8.1	Fermat's Last Theorem	51
8.2	Applying the Idea for a Partial Result	52
8.3	Poincaré Conjecture	53
8.4	Unique Games Conjecture	54
8.5	Open Problems	54
8.6	Notes and Links	54
9	Grigori Perelman: A New Clay Problem	57
9.1	Problems	58
9.2	Open Problems	58
9.3	Notes and Links	59
10	Eric Allender: Solvable Groups	61
10.1	Allender on the Permanent	62
10.2	A Result to Dream of	62
10.3	<i>SOLVE</i>	63
10.4	Open Problems	63
10.5	Notes and Links	64
11	Enrico Bombieri: On Intuition	65
11.1	Number Theory	66
11.2	Geometry	67
11.3	Groups	67
11.4	Reid's Proof	67
11.5	Complexity Theory	69
11.6	Open Problems	69
11.7	Notes and Links	69

12 Fred Hennie: Lower Bounds	71
12.1 The Models	71
12.2 Hennie’s Result	72
12.3 Since Hennie	72
12.4 Toward Even Stronger Bounds	73
12.5 Open Problems	74
12.6 Notes and Links	74
13 Volker Strassen: Amazing Results	75
13.1 Fast Matrix Product	75
13.2 All for the Price of One	76
13.3 Finding Triangles	76
13.4 One Bit, Two Bits, Three Bits, n Bits	77
13.5 Open Problems	77
13.6 Notes and Links	77
14 Adam Smith: Dumb Channels	79
14.1 Computationally Limited Channels	80
14.2 The New Results	80
14.3 Open Problems	81
14.4 Notes and Links	81
15 Georg Cantor: Diagonal Method	83
15.1 Proofs	83
15.2 A Variant of the Classic Proof	84
15.3 A Probability-Based Proof	85
15.4 Open Problems	86
15.5 Notes and Links	86
16 Raymond Smullyan: The Reals Are Uncountable	87
16.1 Alice and Bob Play Some Games	88
16.2 Let’s Look at Bob’s Strategies	89
16.3 Is Diagonalization Cheating?	89
16.4 Can We Fix This?	90
16.5 Open Problems	90
16.6 Notes and Links	91
17 William Tutte: Flow Problems	93
17.1 Flow Conjectures	94
17.2 Tensor Trick	95
17.3 Sketch of Proof	95
17.4 Open Problems	97
17.5 Notes and Links	97
18 Basil Rathbone: Writing a Major Result	99
18.1 My Suggestions	99
18.2 Open Problems	102
18.3 Notes and Links	103

19	Elwyn Berlekamp: Dots and Boxes	105
19.1	A Colorful Game	106
19.2	Packing Graphs	107
19.3	Open Problems	108
19.4	Notes and Links	108
20	David Johnson: Galactic Algorithms	109
20.1	Galactic Algorithms	109
20.2	Some Examples	110
20.3	Open Problems	111
20.4	Notes and Links	112
21	Warren Hirsch: Guessing the Truth	113
21.1	Guesses and the Truth	113
21.2	Open Problems	116
21.3	Notes and Links	116
22	Shimon Even: A Promise Problem	119
22.1	The Promise Problem	120
22.2	The Complexity of TWOPATHS	121
22.3	Open Problems	122
22.4	Notes and Links	122
23	Matei David: Improving Noam Nisan's Generator	123
23.1	The Nisan Generator	124
23.2	Read It Again?	124
23.3	Open Problems	125
23.4	Notes and Links	125
24	Ryan Williams: A New Lower Bound	127
24.1	The Result	127
24.2	The Proof Schema	127
24.3	Open Problems	128
24.4	Notes and Links	128
25	Joel Seiferas: More on the New Lower Bound	129
25.1	The Landscape of Ignorance	130
25.2	Outline of the Proof	130
25.3	Open Problems	132
25.4	Notes and Links	133
26	Victor Klee: Big Results	135
26.1	Three Big Results	135
26.2	One More	137
26.3	Open Problems	138
26.4	Notes and Links	138

27	George Dantzig: Equations, Equations, and Equations	139
27.1	Linear Equations	140
27.2	Linear Equations over Non-negative Numbers	140
27.3	Linear Equations over Natural Numbers	141
27.4	Programming IP	142
27.5	Programming Tricks	143
27.6	Limits on the Power of IP	144
27.7	Complexity Theory	144
27.8	Conventional Wisdom	145
27.9	Open Problems	145
27.10	Notes and Links	145
28	Srinivasa Ramanujan: The Role of Amateurs	147
28.1	Who Is an Amateur?	148
28.2	Some Results of Amateurs	148
28.3	Problem Statements	149
28.4	Can Amateurs Help?	150
28.5	Open Problems	150
28.6	Notes and Links	151
29	John Rhodes: Approaches to Problems	153
29.1	Approaches Used by Mathematics and Theory	153
29.2	Approaches Unique to Mathematics?	154
29.3	Open Problems	156
29.4	Notes and Links	156
30	John Nash: Connections	159
30.1	Connections	160
30.2	The Connection	161
30.3	Open Problems	161
30.4	Notes and Links	162
31	Chee Yap: Computing Digits of π	163
31.1	The BBP Algorithm	164
31.2	The BBP Algorithm Is Not an Algorithm	164
31.3	Yap’s Theorem	164
31.4	Open Problems	165
31.5	Notes and Links	165
32	Henri Lebesgue: Projections Are Tricky	167
32.1	Let’s Be Nice	168
32.2	Projections	168
32.3	Shadows and Slices	169
32.4	Strassen’s Complexity Measure	169
32.5	How the Measure Plays Nice	170
32.6	The Problem with Projections	171
32.7	Endgame?	172

32.8	Open Problems	173
32.9	Notes and Links	173
33	Nina Balcan: A New Model of Complexity	175
33.1	Classic Complexity Models	175
33.2	A New Model	176
33.3	Application to Clustering	176
33.4	Open Problems	177
33.5	Notes and Links	178
34	Sam Buss: Bounded Logic	179
34.1	A Problem with First-Order Logic	180
34.2	Open Problems	181
34.3	Notes and Links	181
35	Anton Klyachko: Car Crashes	183
35.1	Car Crash Theorem	184
35.2	A Proof Sketch	185
35.3	Another View	186
35.4	Why Is It Important?	187
35.5	The Application	187
35.6	Open Problems	188
35.7	Notes and Links	188
36	Bernard Chazelle: Natural Algorithms	189
36.1	Natural	190
36.2	Form of Bernard’s Talk	191
36.3	Content of the Talk	191
36.4	Open Problems	192
36.5	Notes and Links	192
37	Thomas Jech: The Axiom of Choice	195
37.1	Finite Choice	195
37.2	The General Theorem	196
37.3	Open Problems	197
37.4	Notes and Links	197
38	Alfonso Bedoya: Definitions, Definitions, and Definitions	199
38.1	Definitions	199
38.2	Mathematical Definitions	200
38.3	Computational Definitions	201
38.4	Justification of Definitions	202
38.5	Open Problems	203
38.6	Notes and Links	203
39	Hartley Rogers: Complexity Classes	205
39.1	The Big Three Ideas	206
39.2	How Many Classes Are There?	207

39.3	Special Classes	208
39.4	Properties of Complexity Classes	209
39.5	Open Problems	209
39.6	Notes and Links	210
40	Ron Fagin: Second-Order Logic	211
40.1	Courcelle’s Theorem	211
40.2	Treewidth	212
40.3	MSO	212
40.4	Proof Idea	213
40.5	Open Problems	214
40.6	Notes and Links	214
41	Daniel Lokshtanov: Knapsack Problem	215
41.1	The Method	216
41.2	Constant Term Method	217
41.3	Applications of CEM	217
41.4	A Lemma	218
41.5	Open Problems	219
41.6	Notes and Links	219
42	Albert Einstein: Beyond Polynomial Equations	221
42.1	Polynomials Plus	222
42.2	Polynomials Plus Plus	223
42.3	Gravity Lenses	224
42.4	Open Problems	224
42.5	Notes and Links	225
43	Denis Thérien: Solvable Groups	227
43.1	Equations over a Group	228
43.2	No Fundamental Theorem of Groups	228
43.3	A Partial Fundamental Theorem	229
43.4	Toward the General Case	229
43.5	Universal Groups	230
43.6	Open Problems	231
43.7	Notes and Links	231
44	Andreas Björklund: Hamiltonian Cycles	233
44.1	The Result	233
44.2	An Overview of the Proof	234
44.3	Proof Summary	235
44.4	Open Problems	236
44.5	Notes and Links	236
45	David Hilbert: The Nullstellensatz	237
45.1	Hilbert’s Nullstellensatz	238
45.2	An Application	238
45.3	How to Express Injective as an Existential Formula	239

45.4	Open Problems	240
45.5	Notes and Links	240
46	John Hopcroft: Thinking out of the Box	241
46.1	A National Meeting?	242
46.2	Federated Conferences	242
46.3	Open Problems	243
46.4	Notes and Links	243
47	Dick Karp: The Polynomial Hierarchy	245
47.1	Kannan’s Theorem	245
47.2	Kannan’s Proof	246
47.3	Kannan’s Proof—Can We Do Better?	246
47.4	How to Compare Circuits Fast	247
47.5	Open Problems	248
47.6	Notes and Links	248
48	Nick Howgrave-Graham and Antoine Joux: Attacking the Knapsack Problem	249
48.1	The Problem	250
48.2	The New Algorithm	251
48.3	The Schroeppele and Shamir Algorithm	251
48.4	Add Hashing	252
48.5	Open Problems	253
48.6	Notes and Links	253
49	Hedy Lamarr: The Role of Amateurs	255
49.1	Spread Spectrum	255
49.2	Why Did She Fail?	256
49.3	Open Problems	257
49.4	Notes and Links	257
50	Nicolas Courtois: The Linearization Method	259
50.1	Linearization Method	259
50.2	Linearization in Practice	260
50.3	Learning Noisy Parity	261
50.4	Open Problems	262
50.5	Notes and Links	262
51	Neal Koblitz: Attacks on Cryptosystems	263
51.1	Early Days of Cryptography	263
51.2	Proving an OS Kernel Is Secure	264
51.3	Provable Security	265
51.4	Provable Security?	266
51.5	Neal’s Main Attack	267
51.6	Open Problems	267
51.7	Notes and Links	268

52	Richard Feynman: Miracle Numbers	269
52.1	Some Numerology	270
52.2	A Surprising Result	271
52.3	The First-Order Connection	272
52.4	How to Represent the Group	273
52.5	The Proof	273
52.6	Open Problems	274
52.7	Notes and Links	274
53	Patrick Fischer: Programming Turing Machines	275
53.1	The Two Theorems	276
53.2	Applications	276
53.3	Proofs of the Theorems	277
53.4	Open Problems	278
53.5	Notes and Links	278
54	Roger Apéry: Explaining Proofs	281
54.1	Apéry’s Proof	281
54.2	Some Suggestions	282
54.3	Open Problems	285
54.4	Notes and Links	285
55	Ron Rivest: Mathematical Gifts	287
55.1	Gift I	287
55.2	Gift II	288
55.3	Gift III	289
55.4	Gift IV	290
55.5	Open Problems	291
55.6	Notes and Links	291
56	Frank Ryan: The Quarterback Teaches	293
56.1	Ryan’s Seminar	294
56.2	Learning Methods	295
56.3	Open Problems	295
56.4	Notes and Links	295
57	Leonard Schulman: Associativity	297
57.1	Multiplication Tables	297
57.2	Testing Associativity	298
57.3	Open Problems	299
57.4	Notes and Links	299
58	Paul Seymour: Graph Minors	301
58.1	The Result of Grohe	302
58.2	Fixed-Point Logic with Counting	303
58.3	Why Are Minor Results so Major?	303
58.4	Open Problems	304
58.5	Notes and Links	304

59	Alfred Tarski: Lower Bounds on Theories	305
59.1	Upper Bounds on the Theory of the Reals	305
59.2	Lower Bounds on the Theory of the Reals	306
59.3	Lower Bounds on the Complexity of Theories	307
59.4	A Recursion Trick	307
59.5	A Name Trick	308
59.6	Lower Bounds on the Theory of the Complex Numbers?	308
59.7	Open Problems	309
59.8	Notes and Links	309
60	Ken Thompson: Playing Chess	311
60.1	Chess Programs and the Big Match	311
60.2	The Book of Chess	312
60.3	Building Big Files	314
60.4	The Search Problem	314
60.5	Shorter Tablebase Descriptions?	315
60.6	The Tablebase Graphs	316
60.7	The Book as “Deep” Oracle?	316
60.8	Open Problems	317
60.9	Notes and Links	317
61	Virginia Vassilevska: Fixing Tournaments	319
61.1	Ratings	320
61.2	Tournaments	320
61.3	Fixing Tournaments	321
61.4	Open Problems	322
61.5	Notes and Links	322
62	Arkadev Chattopadhyay: Computing Modulo Composites	325
62.1	Representations of Boolean Functions	325
62.2	A Question	325
62.3	Uniqueness	326
62.4	Another Question	327
62.5	A Peek Ahead	327
62.6	Open Problems	328
62.7	Notes and Links	328
63	Charles Bennett: Quantum Protocols	329
63.1	Quantum Protocols	330
63.2	Attacks	331
63.3	A Theorem	332
63.4	Open Problems	332
63.5	Notes and Links	333

Vinay Deolalikar could be the featured subject of this chapter. Or it could be the small group of readers of his 102-page paper in August 2010 claiming to prove $P \neq NP$, and thus resolve in the negative the central open problem in our field of complexity theory. Initially this was a private group to whom the draft paper had been circulated, before its existence was leaked and made public on the blog of Greg and Kat Baker on August 7, 2010. However, the larger interest of the story told here by Ken is the way the review was *crowdsourced* and found what still seem to be all major issues in under two weeks.

Our story of the people and our field's greatest problem centers on a proof, or rather the claim of a proof and the social process this entails. A proof really ceases to be a proof when it is found to have mistakes, though we can still call its written-up form a "proof." The two-week adventure of the process that played out is best told as we experienced it personally, rather than reproducing the posts on our blog. The posts were not the real story anyway. Ken picks up the story.

1.1 The News

After a tumultuous summer attending the birth of a new niece in June just as my wife suddenly needed eye surgery, and a trip to Maine and New Jersey beginning in late July, I expected to settle into a few weeks of August summer routine at home before the Fall 2010 term began. Our first full day back in Buffalo was Sunday, August 8. After church service I forgot to switch my cellphone back on, and unusually for me, did not go online either. I had helped with a post late that Friday night from New Jersey, which Dick had posted Saturday afternoon, so I put the blog out of mind until Monday. Debbie cleaned while I mowed the lawn, and then we discussed what the family routine should be heading into the next school year, before taking a break with the newspapers. As dinner approached I did laundry in the basement, still oblivious to several progressively more excited e-mails and cell calls from Dick about the story which had broken the previous night. Then I switched on my laptop down there and remembered my cellphone too. If you recall a scene in a sci-fi movie where a starship's console suddenly lights up, that's what it felt like.

Dick had in fact already drafted a post. He had glitches with the Georgia Tech e-mail system and expressed worry whether I had received any of the mails. I sent a quick acknowledgment at 5:39pm to Dick and Subruk, saying I would catch news online before reading the draft. I found some discussion about the paper already, including a quotation from Steve Cook that it looked like a “serious attempt,” and the potential dimension of this started widening in my mind. With minimal delay of our 6:30 dinner, I modified a paragraph where Dick had raised a potential objection on whether the proof avoided the so-called *relativization barrier*, and added a paragraph on how it might avoid the *natural proofs* barrier. I also extended a third paragraph to try to remedy the paper’s immediately obvious lack of an actual super-polynomial lower bound on deterministic time for a particular NP-complete problem such as SAT, which all but the most mysteriously indirect proofs of $P \neq NP$ should be expected to do.

1.2 The Paper and First Post

After dinner, still doing laundry in our basement, I noticed that Dick had obtained and included a link to the paper itself. I sat down at my laptop and examined the new wonder, whose first version had 63 pages. Though it ranged from physics to first-order logic to conditional probability spaces, I had some background in all of its areas, and could readily follow the structure of the argument. Most in particular, I had covered Gibbs distributions—as applied to neural networks—in a seminar in the early 1990’s while supervising my first graduate student, Dr. Arun K. Jagota. I had first picked up relevant parts of mathematical physics during my graduate study at Oxford under Dominic J.A. Welsh, even fielding a question about Boltzmann distribution in my oral qualifying exam and refuting an assertion someone had made about percolation in a physics paper.

About forty minutes into reading, I perceived a potential flaw in the paper, one that harked back to a speculative idea I’d had in 1990. I sent Dick a query on something related to it which I didn’t understand, and while waiting for reply, I started writing a potential section with my own objection. When we talked a half-hour later at 8:30, I explained my line of thought, and Dick concurred.

We both decided, however, that my point was better as a comment right below the main post, as we felt it was too early to put any judgments in the main body. Dick had sole control then of the blog’s starship console, so he undertook the sometimes-wonky conversion from our \LaTeX source to the WordPress HTML format. After one more typo-catch exchange he sent me a note at 9:28pm saying he had posted and to go with my comment—the times show an hour earlier since the blog’s clock stays on Standard Time.

Dick titled it, “A Proof That P Is Not Equal To NP ?” He noted that Deolalikar had sent the paper link with permission, and also linked the post by Baker which had broken the news the previous night. We were not fully aware that the news had been leaked to Baker against Deolalikar’s own wishes to keep the paper in private circulation, a policy that we ourselves have observed and recommended in other

cases. Dick posed the question, “Is the paper correct?” Several times starting in the next sentence, he—meaning we—referred to Deolalikar’s paper as a “proof,” “his proof.” This caused consternation among many who would say we should refer to it only as a “claimed proof” or the like, but I had approved using “proof” to mean a piece of text representing a coherent mathematical argument, even though I myself already had a concrete doubt on its correctness. After a quick read, and before letting departmental colleagues know of this by e-mail, I started to put in my comment.

1.3 My Snap-Doubt and Comment

When I clicked to open a comment box there were as yet no comments, but two would sneak in ahead of mine before I finished fixing up the symbols and changing some things I had written. I hadn’t reckoned with a basic fact of “Blog Geometry” that sundry replies to those two comments would soon push mine far down the page. Here is what I wrote, formatted back into \LaTeX :

Having seen this only since 5:30 this (Sunday) afternoon, here’s my attempt at a somewhat more fleshed-out description of the proof strategy, still at a very high (and vague) level: Deolalikar has constructed a vocabulary V such that:

- (1) Satisfiability of a k -CNF formula can be expressed by NP-queries over V —in particular, by an NP-query Q over V that ties in to algorithmic properties.
- (2) All P-queries over V can be expressed by FO+LFP formulas over V .
- (3) $\text{NP} = \text{P}$ implies Q is expressible by an LFP+FO formula over V .
- (4) If Q is expressible by an LFP formula over V , then by the algorithmic tie-in, we get a certain kind of polynomial-time LFP-based algorithm.
- (5) Such an algorithm, however, contradicts known statistical properties of randomized k -SAT when $k \geq 9$.

If there is a conceptual weakness, it might be in step 3. This step may seem obvious given step 2, and maybe the paper has been set up so that it does follow immediately. However, it does need comment, because general statements of the P–LFP characterization are careful to say that every polynomial-time decidable query over a vocabulary V can be encoded and represented by an LFP+FO formula over Strings.

I went on to make an analogy with that old idea from 1990, in which I defined a class Q based on polynomial-time machines that could learn their input only via certain logical queries. Then NQ equaled NP , because the machines could guess queries to make that would reveal the entire input, but Q itself is only a small subclass of P . Thus I had $\text{NQ} \neq Q$ with $\text{NQ} = \text{NP}$, but this did not entail $Q = \text{P}$. I finished my comment by speculating that the paper might also be defining a “ Q ” that likewise falls short of being equal to P . This hunch turned out to be right in several respects.

No one replied to the comment, but I took that as meaning no one found fault with my outline of the paper’s strategy. After sending e-mail telling colleagues about this, I went up from the basement to rejoin my family. But I snuck back down after making sure Debbie was comfortable and saying good-whatever to our more-nocturnal kids. I noticed a comment with a query from Cris Moore, and answered it tying back to my long comment. Then I looked for more information around the Net, and noted that certain other computational theory blogs had not yet picked

up the story. And so to bed, still with little idea what I'd be waking up to the next morning—especially given that we had posted in time for morning not just in Europe but even the Far East.

1.4 Comments and Comments and Comments

When I arrived at my office in mid-morning, there were 13 new top-level comments, some with replies already, beginning with one Moore had entered right after midnight as a followup to the query I'd answered. The last was a comment from David Mix Barrington passing on a potential flaw noted by Lance Fortnow. It shows as "9:30am" because the blog times are Eastern Standard, i.e., GMT-5. Of course I could not make 13 replies, so I chose one detailed comment by Arthur Milchior, a student from France who also does standup comedy and edits Wikipedia pages in logic and computer science theory. I answered him but also addressed Barrington's comment and some other points. Then after printing a longer version of the paper which Deolalikar had released in the meantime, and starting to read it more closely, I noticed Barrington himself had replied to *my* reply only 30 minutes later. This was my first cue about the degree to which the comment threads would become *interactive*.

Dick sent me e-mail noting that WordPress projected the post would have over 30,000 readers in 24 hours, fifteen times our typical rate, and could become one of the top WordPress pages for the day. My colleague Atri Rudra noted that fellow complexity theory blogger Scott Aaronson had picked up the story that morning, and had offered \$200,000 of his own money as a supplement if the proof turned out to be correct and full enough to win the \$1,000,000 Clay Millennium Prize. Lance Fortnow and Bill Gasarch, the original complexity bloggers, had links to that and to us, and gave some specific reasons for doubting that the ingredients of Deolalikar's paper would measure up. I could have spent more time troving for evaluations on other blogs besides those.

However, I did what one might expect a researcher rather than an online maven to do. I went over with the paper to the University at Buffalo Commons to buy lunch and stake out a table outside in the shade for an afternoon of reading on a beautiful August Monday. I looked over the first-order logic aspects that they all had been querying, but was most interested in the latter part of the paper where the actual lower bound was argued. The writing in terms of "phase transitions" was hazy, and as we had already noted in the post, did not give a concrete time lower bound on a concrete problem.

Most in particular, I thought it must be possible to extract from the argument a particular *hardness predicate* that was being employed. A hardness predicate $R(H, n)$ is one that is false for all sufficiently large n when H is an easy problem, and true for all sufficiently large n (or at least infinitely many n) for *some* other problems H . The issue is whether you can define R so that it applies to a problem H that you are interested in, such as SAT, and then whether you can *prove* $R(H, n)$

for the n 's you need. Then you have a proof that H is hard—and if H is SAT and your “easy” class really does include all of P , then you have a proof that $NP \neq P$.

The celebrated 1994 “Natural Proofs” paper of Alexander Razborov and Stephen Rudich placed notable restrictions on predicates R for which this strategy could succeed, when “easy” was taken to refer to the class P/poly of problems having polynomial-sized *circuits*, which includes P . They proved that unless *all* one-way functions—including the ones underlying the RSA cryptosystem as used for Internet commerce—are substantially less secure than currently propounded, then a predicate R for which this strategy could possibly succeed must either be *strangely thin* or *extraordinarily complex*. The former means that $R(H, n)$ would hold only for a negligible fraction of problems H that are actually hard. The latter means that merely evaluating $R(H, n)$ —given any representation of the problem H on inputs of length n such as a 2^n -sized table of values—would take time more than singly exponential in n , such as doubly exponential in n . After demonstrating that all hardness predicates heretofore employed in successful lower bounds were neither thin nor complex, and noting that basically all properties commonly used in relevant mathematical fields are decidable in time singly exponential in n , Razborov and Rudich generated a consensus that theirs was a substantial barrier any hardness proof such as Deolalikar’s must overcome. There was a potential “out” in the difference between P/poly and P , but it was not clear that Deolalikar’s strategy was really driving at it.

I had studied hardness predicates arising in algebraic geometry and the theory of ideals that were hard for exponential space—hence ostensibly outside single exponential time—and yet were mathematically tractable to work with. I had also written a survey article explaining how the predicates employed in a far-reaching program by Ketan Mulmuley escaped a single exponential bound. Hence I expected a similarly complex predicate to lie at the heart of the phase-transition argument, one whose novelty would be valuable even if the paper itself failed. I tried to tease it out, but falling short of success as the afternoon wore on, I started getting frustrated with the paper’s lack of explicitness. I returned to my office after 3pm. Little did I know that from then on my time would be mostly occupied in systematizing and summarizing the way others were reading the paper, not so much from other blogs but from myriad comments in ours.

I came back to find all the comment threads expanding, while some other researchers had provided detailed notes and queries on points in the paper. Indeed I noticed that the overnight comment by Moore, which my morning reply had passed over, actually struck at more-immediate aspects of the part of the argument I had just been examining. When Dick and I spoke that afternoon, we realized the need to organize and summarize what people were saying.

The comments clustered around four major issues. We started drafting a post titled “Issues in the Proof That $P \neq NP$ ”. We hyperlinked certain comments in the original post for each of the four issues. We finished and uploaded it Monday evening. Right after it went up, I added a comment noting a question raised by Ryan Williams on his Twitter account, a comment related to one of the issues in Slashdot’s item from the previous day, and several matters raised by a commenter named

“vloodin” whose real identity I still do not know. The next morning a commenter named “harrison” replied by querying another issue raised by Ryan about isolation of k -SAT formulas in the phase-transition argument, and Ryan himself replied twice in return. I had an urge to reply with matters from my read of the paper that we had left out of the post, but realized doing so would take me too long past midnight and so went to bed.

1.5 Back to the Future

When I awoke Tuesday morning, I found a flurry of e-mails in my box from between 1am and 3:30am, four from Terence Tao, two from “Geomblog” creator Suresh Venkatasubramanian, one from Gil Kalai, and all copying Dick and Timothy Gowers. They proposed organizing the investigation of the paper under the “PolyMath” wiki-based structure proposed in a paper by Gowers with Michael Nielsen, “Massively collaborative mathematics,” in *Nature* vol. 466, Oct. 2009, pages 879–881. After Dick signaled our co-operation, I replied:

I also agree, and will be happy to take advantage of the framework. I’ve been interested in the co-ordination of research discussion by Internet ever since the astounding Queen endgame of the Kasparov–World match in 1999.

This brought a flashback of pain and promise. In summer 1999 I was at the flood of my own concerted attempt to prove $NP \neq P$, or rather an algebraic analogue of it made famous by Leslie Valiant. My main graduate student in this work told me of a chess match sponsored by Microsoft to advertise their online “MSN Gaming Zone” in which Garry Kasparov was to take on the entire world voting on moves at the MSN site, at the rate of one move per 24 hours. I’ve never taken up playing chess online, and I ignored his repeated suggestions until well into July.

I found that MSN had set up a system where three young teenage players of about my playing strength (one division above Master but short of Grandmaster) would each suggest a move, and the World would vote for one of those moves or for a write-in. They were Elisabeth Paetz of Germany, Etienne Bacrot of France, and Irina Krush of Brooklyn. I picked up the game at Move 27 in a position that was *wild*, and found that the wildness had come from a fantastic new sacrificial move suggested by Krush for Move 10 of her side playing Black. In fact, early this year, 2013, world champion Viswanathan Anand won a brilliant game using Krush’s move when his opponent dared him to repeat it, so it’s a durably good move. The World players in 1999 were so appreciative that they voted for every move suggested by Krush subsequently, until the critical final phase in which the vote was apparently first hacked by a personage calling himself “Joe One-Two” in Spanish, and then a communications breakdown occurred between Krush and MSN on the final relevant move.

Two things about the match disquieted me. First, players of full Grandmaster strength were being organized to analyze and provide input on the forum provided by MSN, one from St. Petersburg, Russia, calling themselves the “GM School.” I felt there should have been a stated policy that no one above the level of the three

junior panelists should take part. Second, there were a number of people posting “flames” on the board, directed largely at Krush when she recommended a move other than what they favored, especially when she proposed trading Queens to enter a desperate-looking endgame. Krush was being managed by a childhood chess friend of mine, and suddenly this seemed personal. I looked at the position after the trade enough to tell it was still fully dynamic. So I waded in on the forum myself to say hey guys, let’s appreciate how amazing the game has been and still is, with every single one of the seven remaining pawns—two for Kasparov as White and five for the World as Black—being a so-called “passed pawn” readily capable of becoming a new Queen.

Vacation more than research had made me a less-involved onlooker in August 1999, but as the battle raged into September, a long looming unavoidable sequence emerged in which both sides would make a Queen. In this I saw an aspect that *is* research. The sequence would leave just the King, new Queen, and another pawn for Kasparov, versus King, new Queen, and two pawns for the World. But Kasparov’s other pawn would be a greater menace than our two combined, and I recognized that the World would still have to play with utmost care to earn the golden ticket of a draw. The GM-schoolers had peeled away, but amateur-level players on the forum started looking ahead at the sheaves of variations burgeoning from the two female coronations at the end of the forced sequence. There were to be only seven pieces, but the complexity would ramp up to levels unseen even in this game.

The endgame has always been my strength as a player, and I recognized general strategy principles to inform and organize the analysis efforts. I spent several days—that is nights—writing up what became a manifesto of *thirty* numbered points, titled “World Team Endgame Strategy Explained.” It had the same dozen-page length as a typical conference research paper submission. It was hailed by those doing the hard work on the forum. Someday I hope to tell the story fully, including how my strategy caused us to play a move now known to be losing, and an ingenious trap discovered by a player one division below Master that Kasparov’s own notes showed no inkling of. Alas the aforementioned communications breakdown prevented a crucial preparatory move for the trap from being posted as Irina’s choice, and the World voted for a superficial centering move that lost in short order.

Of Kasparov I greatly appreciate that even though the unexpected effort and duration of the game had derailed his preparations for his forthcoming title defense, he still cared enough to release thirteen pages of notes to claim that he would have won even without the mishap. I did, however, refute his claim within two days. Then the “postgame” entered a new interface of computer and chess spheres, in which exhaustive enumeration of possible positions took over from the brave human analysts. The resulting tabulation of perfect play (modulo ignorance of underpromotion which was later rectified without major change) upheld my refutation 100 %, but also revealed just before Thanksgiving 2010 that White has a winning line no human had thought of. Kasparov later published it without comment in his book on the match.

By this time I was well distracted from my student and our research drive, which was soon blunted anyway by a refutation of its workable hypothesis by a mathemati-

cian in Italy. I could say much more about my own chess analysis, but when I had written my strategy article in September I wasn't promoting my own chess—I was helping a host of others, and that was the story.

Come a decade later, and the larger story was recognized by Nielsen himself. While writing his 2011 book *Reinventing Discovery: The New Era of Networked Science*, he made Kasparov-World one of his main running examples, and drew upon my materials. Now I was graced with the opportunity, indeed the obligation, to do the same action regarding the most central problem in my professional field—and with some of the same people involved. I felt amazed and excited.

I also felt bulldozed—as I stared at the reasons why scrolling to the ends of our two posts still left the scrollbar a tiny little square down only an inch from the top of the window:

205 Comments

127 Comments

—numbers that would go over 250 and 300 in succeeding posts. Indeed when Dick started drafting a post announcing the PolyMath wiki page, he did so partly to enable a fresh comment thread.

1.6 The Group

Our Tuesday post on “The Group” also addressed three suggestions to Dr. Deolalikar himself, though in third person, on easier partial results that should be entailed by his methodology that would already be significant advances, and whose demonstration would do a lot to satisfy the community. Dick did the posting—I was not yet on the masthead—while I prepared my own contribution to the wiki page based on what I had gleaned from the paper the day before. I sent it as an e-mail internally to the five other group members.

I expected some internal discussion. That would be an *involution*—the way things had been discussed for centuries. What was happening, however, was *evolution* with heavy emphasis on the root stem *e-* or *ex-* meaning *out*—evolution meaning *turned outwards*, so onto the wiki it went, to see what response my writeup might get from outside. Outsiders, however, had plenty to investigate before getting to my hypotheticals. Nobody I know (including myself) has taken time since then to delve into them further, although they are still the only way I see to extract something solid and publishable from Deolalikar's work.

I read the initial forms of the wiki page, and then went over to the burgeoning comment threads, and read not just each for content but also to see that they were in sync with each other. From then on I did not contribute directly to the wiki page, but kept this indirect role, and I also started posting some comments to clarify and answer queries myself. Especially since Dick was occupied enough already by communications with ACM officials and other principals, I saw keeping abreast of the comment threads as my best purpose.

1.7 Internet Non-resistance

Overnight Tuesday there began some criticism of the whole event. Russell Impagliazzo, whom I might nominate as the most quietly powerful researcher pushing the field for a quarter century, commented at midnight Tuesday ET that he was “distressed to see so many people spending a lot of time on this,” and that the paper was not a serious attempt and had no original ideas. By the time I woke up Wednesday morning, there were a couple dozen replies agreeing and disagreeing, very respectfully, including a followup by Russell himself. But there was also a long anonymous comment by someone labeled “Concerned Citizen,” who began by quoting Russell’s words and went on:

Thank you Russell for saying something non-anonymously that the rest of us are thinking. Look at the paper. It is not a serious attempt. It is not a rigorously written mathematical argument. If any serious mathematician really believed they had an approach to P vs. NP, they would definitely seek to write it in a straightforward, easily verifiable way. And Deolalikar has a PhD in mathematics; he should know the standards of mathematical writing.

Read the paper. He is more interested in buzzwords than clarity. It is quite clear that there is something very wrong in the model theory part of the paper, which is where all the interesting action has to be happening. After having a very weak (and incorrect) characterization of P, the proof could end in a variety of ways. The author has chosen to reach a contradiction using random k -SAT. And this, apparently, is leaving some people (who are unfamiliar with the work on random CSPs) hailing it as a “new approach.”

I think that many bloggers have become quite enamoured with being part of the publicity storm, and are thus complicit in its continuation. The situation is very simple: If he believes he has a proof, then he should write a 10-page version without all the unnecessary background material, stream of consciousness rambling, and overly verbose ruminations. Just the math, please.

What you will find is not that he has discovered a shocking new technique. This is not very good work that happens to have a subtle flaw. This will not solve some special case like NL vs. NP. Instead, it is simply a misunderstanding caused by a lack of rigor and a lack of professional mathematical colleagues against whom he can check his ideas.

Let’s try to be responsible here, as a community. Let’s not say things like “Vinay’s use of conditional independence seems strikingly new.” How can you say this without understanding the proof, even at a very basic level? This reeks of bombastic pseudoscience.

I had written the line in the post causing the offense of this paragraph, based on the context of his Gibbs argument and my reading that Monday. I was gratified that my Sunday evening inkling about the characterization of “P” being too weak was seen as such. After again noting the unclarity in Deolalikar’s paper, the comment concluded:

The biggest mistake here is the response of the community. We should definitely NOT be supportive of such efforts. Think about it: Do you want to have such a reaction every week, to every new proof claiming to resolve a famous open question? We should reinforce the responsible action: To produce a clear, concise, carefully written proof and send it quietly to a few experts who can give it a first reading. After it has been revised and rewritten to their standards, humbly circulate it more widely; say that you are hopeful that your proof is correct, and you would be exceedingly grateful to address the comments of interested readers.

As currently written and announced, Deolalikar’s manuscript is disrespectful to the community. We are given a bad name so he can have 15 minutes of fame. Please stop propagating this madness. It’s irresponsible.

I woke up on Wednesday to find a few responses to this comment also: an anonymous “Amen,” and then “Hear, hear” from Hugo van den Berg of Warwick, who had also chimed in with support of Russell and scorn of the paper, and some longer defenses of Deolalikar and the situation. I imagined that people would be interested to see what the reaction of Dick and me would be. And I quickly reached a conclusion on exactly what that reaction should be:

Nothing.

My reason was based on what was happening just below this section of the thread. Kalai had posed a technical question on what another person examining the paper had commented and got a reply from him. Alberto Atserias stated his agreement with Russell but with his very next sentence helped us by saying exactly where he thought the flaw in the model-theory part of the paper was, and referenced two comments in previous posts (one in Aaronson’s item). Timothy Gowers sharpened the call for a synopsis of how the proof worked. A stream of further substantive notes leaning on previous ones was starting, including especially quickly perceptive ones by “vloodin.”

I decided it was of utmost importance not to disturb this discussion by doing anything to give oxygen to the “meta-discussion.” I called Dick first-thing about this and he agreed. Hence even when some others defended us, even about what we had opined was new, Dick and I just stayed silent. That afternoon, Tao gave what we still regard as the central view:

[Russell,] I understand your concern about the snowball effect, but actually I think this time around, the fact that we concentrated all the discussion into a single location made it more efficient than if everybody just heard rumours that everybody else was looking at it, and dozens of experts each independently and privately wasted hours of time doing the same basic groundwork of reading the entire paper before reaching their conclusions. I think the net time cost has actually been reduced by this method, though of course the cost is far more visible as a consequence.

I think there are a number of useful secondary benefits too. For instance, I think this experience has greatly improved our $P \neq NP$ Bayesian spam filter that is already quite good (but not perfect) at filtering out most attempts at this problem; and I think we are better prepared to handle the next proposed solution to a major problem that makes it past our existing spam filters. Also, I think the authors of such proposed solutions may perhaps also have a better idea of what to expect, and how to avoid unnecessary trouble (well, the best advice here is not even to try solving such problems directly), but somehow I think this advice will not be heeded [this came with a smiley emoticon]. Such authors often complain that their work gets ignored, but actually they may be lucky in some ways, when compared against those authors whose work gets *noticed*...

It is important to be reminded that Deolalikar himself had originally not wanted to be noticed. By Wednesday we understood fully that he had been “outed” by a leak from someone in the original distribution. Once that happened he had no choice but to go all the way public.

Now we hoped not only that he would respond to the three suggestions in our post, but also that he too would become a part of the public discussion. We did in fact get a private response from him, and he made a short statement that accompanied a revision of his paper.

1.8 Company...

Vinay Deolalikar's response, however, answered only a relatively easy question: how and why his paper distinguished between values of k in arguing structural properties of a space associated to the k -SAT problem that imply non-membership in P , so as to avoid the appearance queried by some that it would imply the false statement $2SAT \notin P$. It did not engage with suggestions and queries in our posts or the many comments which we and the wiki page had summarized. We were disappointed, but Dick went ahead and framed a post for Wednesday around his answer, our fourth post in four days.

Meanwhile, we had communications that brought into focus what is now regarded as the first of three main flaws in the paper, about the use of logic. Neil Immerman, whom we each regard as a personal friend as well as one of the foremost experts on finite model theory, addressed a two-page letter to Dr. Deolalikar in full detail, and gave us leave to release it in a fifth post, which we did on Thursday, August 12. We highlighted the following sentences from the letter:

Thus, your restriction to only have successor and to restrict to monadic fixed points is fundamental. In this domain—only monadic fixed points and successor—FO(LFP) does not express all of P !

This post, which was current for three days and remained on the WordPress “Top Posts” list even through the following Monday, attracted 327 comments. Robert Solovay piped up to say he had placed Immerman's letter on the wiki page. Anuj Dawar, perhaps Britain's leading finite model theory expert, concurred. Janos Simon and Mauricio Karchmer commented about possible satellite ideas. There was also a followup comment from “Concerned Citizen,” but this time the ‘meta-discussion’ was shorter with fewer siding with him. My favorite reply was from someone styled “Quantum Tennis Referee”:

As if the community needs protecting! ha! What a wild notion!

An interesting sequence near the top of that thread comes I believe closest to the sociological heart, though still from the cynical side. Amir Michail, the creator of DropZap for iPhone and other games, first imputed (by reference to “cynical people”) that the attention had been motivated “to promote the field, attract graduate students, and increase theory funding.” I hope this chapter makes clear that this opinion is wrong. But following up to a responder he said something closer:

Do you think the publicity from this will have an impact on complexity theory? If so, what sort of impact?

I find it hard to believe that those who commented on the proof gave no thought whatsoever to the fact that the world is watching. This has become a spectator sport.

To which an anonymous responder affirmed, “Amir: People are commenting here *precisely* because they know the world is watching them.”

Now here is my take: Most people who combine the terms “Internet” and “global village” attach an emotion of hope to their statements, and expectation that this is for good. There is a worldwide community at the juncture of computer science and

mathematics. When we come together for conferences or workshops there is a large component of *conviviality*, meant according to its roots of “together” and “live” more than drink and merriment. We cannot otherwise get the *con-* part, even by e-mails, which are largely individual. But the comment threads took a non-negligible step toward a virtual gathering. Various personal opinions amounted to statements of community values, and even when some of those values were opposed they jointly mapped community feelings.

In brief, the threads became a source of *company*. I myself noticed some people I’d been closer to in the past century, and sent short greeting e-mails. There is not space here to mention them, or many others I’ve known or known-of who made substantive contributions.

1.9 ...And a Crowd

There was more in that comment sequence. A commenter named “Random” whom I’d seen on the first day wrote:

Paradoxically, I think a lot of people who are still paying attention are non-specialists (like me). For people who know enough about barriers, it must have been easy to take a cursory glance, convince oneself that the strategy was flawed, and go back to one’s own research.

Alexander Razborov—he of the “Natural Proofs” barrier mentioned above—responded “yes” but added:

P.S. But I have to admit I am a little bit harassed by the Russian media these days (I am sure many colleagues have similar problems), and it is *extremely* handy to have this blog around as a pointer. Dick, the Group, et al.—thank you *very* much for investing effort into this.

Some media had indeed reached us. Lee Gomes of Forbes Online contacted me on the first Tuesday and kept pace with our posts and the comments:

- (1) “The Non-Flaming of an HP Mathematician” (August 10).
- (2) “A Beautiful Sausage” (August 11).
- (3) “How to Think Like a Mathematician” (August 12).
- (4) “Now It’s the Slow Roasting of an HP Mathematician” (August 17).

As the titles already make clear, the main subject was not the paper itself but rather the process of reviewing research collaboratively online, and of doing research (that way) to begin with. His first column remarked on the decorum of the review and the comment threads. In an update to it he quoted me defending Deolalikar’s actions and the autonomy of research at places like HP. In the second he quoted a comment by Tao:

Regardless of how significant or useful Deolalikar’s proof is (and how much of a “waste of time” it would be for so many people to be looking at it), it has provided a rare opportunity to gather a surprisingly large number of experts to have an extremely high quality conversation around $P \neq NP$. One may have strong feelings about the specific circumstances that created this opportunity, but I do not think that should be cause to squander the opportunity altogether.

This article went on to tell how aspects of the discussion were being made “accessible to anyone” in both the posts and the comments, even joking that mutual-fund managers could thereby “learn a lot about how to think clearly.” The last reflected the reality that our fifth post had the words “fatal flaws” in its title, as consensus grew that the jig was up with the paper. But the paper was less and less the story anyway, and the article by John Markoff in the Tuesday August 17 “Science Times” section of The New York Times led with the main subject.

The potential of Internet-based collaboration was vividly demonstrated this month when complexity theorists used blogs and wikis to pounce on a claimed proof for one of the most profound and difficult problems facing mathematicians and computer scientists.

Dick and I started another post to mark a week since the release of the paper, and to provide a fresh space for comments. Only rarely does the photo atop our posts have more than one person, but this time Dick chose to feature a *crowd*. We were grateful that the intent expressed at the beginning by Suresh on his own blog of “crowdsourcing” the review had worked out as well as it had. For the body of the post we worked on summarizing what had been established about the paper and its issues thus far.

1.10 The Issues Crystallize

I in particular was occupied that Saturday with the current long comment thread, besides also helping edit an item by Dick for the *Communications of the ACM* blog. Saturday afternoon I came again over comments regretting that someone like Tao—meaning a generic person of his talent—should “waste his time” on something like this. At the same time I noticed that the actual Tao was contributing some new comments, until by about 3pm he had 8 of the 15 most recent comments appearing in a sidebar to the blog.

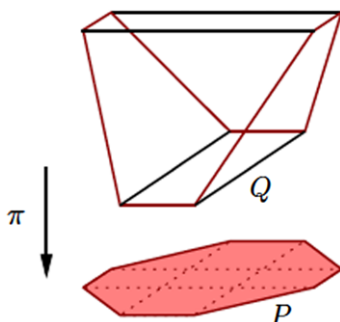
I was still catching up with observations Tao had made in the same thread the previous day, including pointing to a deep assessment by Gowers on his own blog, so I did not immediately pick up what Tao was doing. I helped Dick draft the body around what we called “The Two Main Issues”. Immerman’s point reappeared as the first issue, and then I wrote two sections giving larger shrift to the solution space issues which we had first noticed in the comments from Ryan Williams. I actually tweaked and shortcutted Ryan’s up-to-date argument a little, and wondered if I’d be corrected—there not being time to run my writeup by him first. I also referenced the beginning of the string of comments by Tao that I’d marveled at that afternoon, but by the time we hit send on the post early on Sunday the 15th, I hadn’t appreciated where they led.

What Tao had actually done was find a third issue, more fundamental than the other two. Commenter “vloodin” piped up about our omission right away, but Tao—who among all his talents has mastered arts of community—quickly put all three

into proper perspective, in a most genteel way. And this issue turns on a principle whose statement can be appreciated by all:

A shadow can sometimes be more complex than the object it is projected from.

Here is a visual example reproduced from a paper by Sebastian Pokutta and others in the case of polytopes:



Here “complexity” refers to counting the facets of the polytopes, which is the same as counting the number of half-spaces used to define them. The 3D object has 6 facets, but its 2D shadow has 8.

In formal mathematics, a shadow is defined by a *projection* operation. This notion applies formally even in complexity theory. By that notion, NP is a projection of P, exactly the one that Deolalikar himself had purposed to prove is more complex.

What Tao had ferreted out of the part of the paper dealing with physics and conditional probability distributions—the part least familiar to us complexity-theory people—was an implicit assumption that a certain complexity property would not increase under projections. In that context it was an easy intuition to have, but a false one. Tao phrased this in terms of what he called “ppp” not being equal to Deolalikar’s “pp”, with the extra ‘p’ standing for “projection.” Once Dick and I understood this Sunday evening into Monday, it was our greatest “aha” moment of the whole episode.

1.11 The Falling Action

Our Sunday post also had a three-page response by Deolalikar to “3 issues,” but while the third was on the subject of how his conditional-probability argument was parameterized, it did not yet see the projection-closure issue in its clear form. He promised that these issues would be addressed more fully in a “version being prepared, which will be sent for peer and journal review.” Thus he signalled his desire to revert to how he had originally intended his work to be appraised, not under the eyes of the whole world. Alas in a play one cannot go back from Act IV to Act II. We

quoted a paragraph that he had addressed to the first-order objection, but followed by writing:

The last sentence aims to answer Neil's objection, but does it? Recall Neil's contention is that the class X whose power is quantified by the finite-model theory section of the paper is not P , but rather a class already known to be properly contained in P . No one has disputed Immerman's counter-claim. We are, as we stated earlier, disappointed that this essence of Neil's communication has not been engaged.

This post attracted 264 comments and links over three days. Over that time Dick tried to draw more out of Deolalikar himself, but he instead removed the paper drafts from public access on his webpage, leaving only the three-page synopsis. Almost everyone I've mentioned and more contributed to the technical discussion, which crystallized exactly how the projection argument connected to the other components of the paper. The question of whether the space called SAT0 in the post could be the support of a 'ppp' distribution was hammered out in the affirmative, but this did not patch the paper. The thread then shifted over to Deolalikar's shutting his doors, while Dick and I answered 'no' about whether we or anyone else in the "Group" circle had received a revised manuscript.

Dick drafted a new post for the 18th. We felt it important to do a "rewind" and explain for the general public why a *proof* of $P \neq NP$ would be important. What is a proof itself good for? Why desire a proof of this inequality when most practitioners have latched on to the reality that most NP-complete problems have very many instances that fast programs *are* able to solve well? He then attached my roundup of the developments with Deolalikar and perspective of the status, the latter linking an opinion by Williams.

Meanwhile, I occupied myself with explaining the projection issue for the public. This became only my second full post for the blog—recall I was not yet on the masthead. For some time I had been intending to present a famous lower bound argument by Volker Strassen that my own effort toward $NP \neq P$ had focused on trying to extend. I was excited to realize that I could do both at once, because I could cast Strassen's argument as an instance where closure under projections *does* succeed. Thus it would present the intellectual impasse in Deolalikar's paper as a compare-contrast, while having value apart from the previous ten days' episode.

Dick looked at my incomplete first draft on the 18th, and drew from his knowledge of mathematical history an event that made a perfect lead-in. The great Henri Lebesgue had once made the error of assuming that the collection of sets defined by Émile Borel was closed under projection. That this was false came as a double surprise to me. First, I had forgotten it from my undergraduate topology and analysis courses. Second and more notably, I recalled Mike Sipser's analogy between Borel sets and the P , NP , $co-NP$, . . . hierarchy, under which projections *do* stay within the hierarchy.

After getting Dick's preamble that evening, I worked past 3am to finish the post. I did this and still managed to fulfill my fatherly duty of taking the kids to a 9:30am haircut appointment. This was in someone's house, and while playing with the family dogs and stepping over young kids' toys and sitting outside in the yard, I appreciated the material concreteness away from a glowing screen, while letting the

swirling projections of the eleven days subside into a dazed satisfaction. I also appreciated a supportive family in circumstances that made unusual demands at a difficult time. This post went out on the 19th and is reproduced here as Chap. 32.

Dick followed four days later with a short post looking back and referencing his article for *Communications of the ACM*, but there was no more news and only a little more discussion of the paper, as comments went under 50 in both posts. After a weekend preparing for the beginning of classes at Georgia Tech and the University at Buffalo, we reverted to an “easy” and “uncontroversial” topic: quantum computation.

1.12 What Did We Learn?

There have unfortunately not been any happy developments to report with Vinay Deolalikar himself in the three years since. His revised paper doubled in length to 200 pages, but we glean that it did not increase appreciably in content. He has not retracted his claim or otherwise responded about it in a public way. He continues to work at HP Labs but at press time his webpage has no research entries after 2010, and on this affair has only a statement that seems frozen in time:

BASIC RESEARCH

Vinay Deolalikar. $P \neq NP$. The preliminary version was meant to solicit feedback from a few researchers as is customarily done. It illustrated the interplay of principles from various areas, which was the major effort in constructing the proof. I have fixed all the issues that were raised about the preliminary version in a revised manuscript; clarified some concepts; and obtained simpler proofs of several claims. Once I hear back from the journal as part of due process, I will put up the final version on this website.

I stand by my belief that a publishable paper can be salvaged from the conditional-probability and first-order part, plus a scholium on hardness predicates, but with so much going on, including successful development of my model of human decision-making at chess and its employment in several cheating cases since then, I have not taken any step to further this. Nothing directly positive that we know has been traced back to this work.

As for what we learned about mass collaborative research, this is still awaiting the proverbial “proof of the pudding.” We have not reported on the blog any mathematical theorem that was obtained since then by open discussion by a dozen or more researchers. It is easier to tear potential proofs down than to build them up.

At least we showed that constructive discussion can be maintained in an open online forum. The hopeful relevance for scientific practice was hailed by Stephen Landsburg, an economist at the nearby University of Rochester and popular writer, in a post on his “Big Questions” blog titled “O Brave New World”—on August 16th in the fullest swing. It was picked up in the *New Scientist* magazine the following month, and has been referenced since, including being the sole subject of a May 4, 2012 article by Srinath Perur for India’s “Fountain Ink” e-magazine, titled “How the science works in science.” Well, there are many ways science works in science, and in an arena of survival of the fittest, we still need to see how this fits.

We have, however, been able to apply the most close-in lessons to our own practice. We have been slowly and carefully and privately mediating a potential approach and claim to prove the opposite, that $P = NP$. We expect that this claim will be public before this book appears, but our private contacts both pro and con recognize the need to find the best presentation of both a lemma–theorem–proof structure and fine details of notation, while keeping the paper well under 100 pages.

Most important, in 2012 we repeated the experiment in a different subject with two willing parties in a controlled fashion from the start, and stimulated some positive research. Yes, the subject was quantum computation. We hosted a debate on whether quantum computers will ever be feasible to build on a large enough scale to be useful. Gil Kalai, whom I've mentioned, who also publishes the blog "Combinatorics and More," took the "con" side, while Aram Harrow, recently hired to MIT's physics faculty, defended the consensus of the quantum computation community. What had been envisioned as a three-part debate became eight posts spanning much of the year, and several of these posts topped 200 comments with much technical discussion that never veered from decorum. Out of this so far has come a paper by Harrow with Steve Flammia showing that one of Kalai's conjectures fails for ternary quantum primitives in place of binary quantum bits, and two papers by renowned physicist John Preskill who also took part in the discussions. Preskill's papers, one updating the other, clarified the argument over proper noise models for quantum computers in a way supporting Harrow's position.

Dick and I ourselves intend to further this debate, I with an argument that Strassen's lower-bound predicate—the same one in my projections post—also furnishes an algebraic measure of multi-way entanglement in quantum circuits. This is still for the future as I write, as is so much else.

1.13 Notes and Links

The eight posts:

<http://rjlipton.wordpress.com/2010/08/08/a-proof-that-p-is-not-equal-to-np/>

<http://rjlipton.wordpress.com/2010/08/09/issues-in-the-proof-that-p≠np/>

<http://rjlipton.wordpress.com/2010/08/10/update-on-deolalikars-proof-that-p≠np/>

<http://rjlipton.wordpress.com/2010/08/11/deolalikar-responds-to-issues-about-his-p≠np-proof/>

<http://rjlipton.wordpress.com/2010/08/12/fatal-flaws-in-deolalikars-proof/>

<http://rjlipton.wordpress.com/2010/08/15/the-p≠np-proof-is-one-week-old/>

<http://rjlipton.wordpress.com/2010/08/18/proofs-proofs-who-needs-proofs/>

<http://rjlipton.wordpress.com/2010/08/19/projections-can-be-tricky/>

Greg Baker post:

<http://gregbaker.ca/blog/2010/08/07/p-n-np/>

Vinay Deolalikar's webpage at HP Labs:

http://www.hpl.hp.com/personal/Vinay_Deolalikar/

Michael Nielsen, *Reinventing Discovery*:

<http://press.princeton.edu/titles/9517.html>

My September 1999 chess strategy guide:

<http://www.cse.buffalo.edu/~regan/chess/K-W/wtstrategy.html>

Lee Gomes' columns in Forbes Online:

<http://www.forbes.com/sites/leegomes/2010/08/10/the-non-flaming-of-an-hp-mathematician/>

<http://www.forbes.com/sites/leegomes/2010/08/11/a-beautiful-sausage/>

<http://www.forbes.com/sites/leegomes/2010/08/12/how-to-think-like-a-mathematician/>

<http://www.forbes.com/sites/leegomes/2010/08/17/now-its-the-slow-roasting-of-an-hp-mathematician/>

Dick's CACM post:

<http://cacm.acm.org/blogs/blog-cacm/97587-a-tale-of-a-serious-attempt-at-p≠np/fulltext>

Scott Aaronson's \$200,000 offer:

<http://www.scottaaronson.com/blog/?p=456>

Steven Landsburg's blog post:

<http://www.thebigquestions.com/2010/08/16/o-brave-new-world/>

John Markoff's New York Times article:

<http://www.nytimes.com/2010/08/17/science/17proof.html>

Picture credits:

Polytope and projections: presentation slides by Sebastian Pokutta,

<http://spokutta.wordpress.com/2012/01/05/1311/>

Kenneth Iverson was a mathematician who is most famous for designing APL. This was the name of his programming language, and it cleverly stood for “A Programming Language.” The language is unique—unlike almost any other language—and contains many powerful and interesting ideas. He won the 1979 Turing Award for this and related work.

We will talk about notation in mathematics and theory, and how notation can play a role in our thinking.

When I was a junior faculty member at Yale University, in the early 1970s, APL was the language we used in our beginning programming class. The reason we used it was simple: Alan Perlis, the leader of the department, loved the language. I was never completely sure why Alan loved it, but he did. And so we used it to teach our beginning students how to program.

Iverson had created his language first as a notation for describing complex digital systems. The notation was so powerful that even a complex object like a processor could be modeled in his language in relatively few lines of code. The lines might be close to unreadable, but they were few. Later the language was implemented and had a small but strong set of believers. Clearly Alan was one of them, since he once said:

A language that doesn't affect the way you think about programming is not worth knowing.

APL was great for working with vectors, matrices, and even higher-order objects. It had an almost uncountable number of built-in symbols that could do powerful operations on these objects. For example, a program to find all the primes below R is:

$$(\sim R \in R \circ . \times R) / R \leftarrow 1 \downarrow \iota R.$$

This is typical APL: very succinct, lots of powerful operators, and no keywords. The explanation of this program is given by Brad McCormick on a wonderful [page](#) about APL.

One [comment](#) about APL is:

[The language was] famous for its enormous character set, and for being able to write whole accounting packages or air traffic control systems with a few incomprehensible key strokes.

Albeit not quite right and not quite nice, this comment does capture the spirit of Iverson’s creation. To make complex functions expressible tersely was an interesting idea. For example, Michael Gertelman has [written](#) Conway’s Game of Life as one line of APL:

$$\begin{aligned} \Phi' \square', \in N\rho \subset S \leftarrow' \leftarrow \square \leftarrow (3 = T) \vee M \wedge 2 \\ = T \leftarrow \supset + / (V\phi'' \subset M), (V\Theta'' \subset M), (V, \phi V)\phi'' (V, V \leftarrow 1^{-1})\Theta'' \subset M' \end{aligned}$$

This should make it clear that this language was powerful, perhaps too powerful.

When I had to learn APL in order to teach the beginning class, I decided to start a project on building a better implementation. The standard implementation of the language was as an interpreter, while one of my Yale PhD students, Tim Budd, eventually wrote a compiler for APL. We also proved in modern terms a theorem about the one-liners of the language:

Theorem 2.1 *Every APL one-line program can be computed in logspace.*

Proving this was not completely trivial, since the operators are so powerful. Perhaps another time I will discuss this work in more detail. Tim wrote a [book](#) on his compiler.

Let’s turn to discuss various notations used in math and theory.

2.1 Good Notation

It is hard to imagine that there was a time when mathematicians did not even have basic symbols to express their ideas. For the many who believe that notation helps to shape the way we think, clearly without basic symbols modern mathematics would be impossible—or at least extremely difficult.

- **Robert Recorde** is credited with introducing the equality symbol in 1557. He [said](#)

“... [T]o avoid the tedious repetition of these words, ‘is equal to,’ I will set (as I do often in work use) a pair of parallels of one length (thus =), because no two things can be more equal.

- **René Descartes** is known for the first use of superscripts to denote powers:

$$x^4 = x \cdot x \cdot x \cdot x.$$

François Viète introduced the idea of using vowels for unknowns and consonants for known quantities. Descartes changed this to: use letters at the end of the alphabet for unknowns and letters at the beginning for knowns.

One amusing part of this notation was that Descartes thought that x, y, z would be equally used by mathematicians. But it is x this and x that... The story—maybe a legend only—is that there is a reason that x became the dominant letter to denote an unknown. Printers had to set Descartes’ papers in type,

and they used many y 's and z 's, since the French language uses them quite a bit. But French almost never uses x , so Descartes' *La Géométrie* used x as the variable most of the time.

- **Isaac Newton and Gottfried Leibniz** invented the calculus. There is a controversy that continues to this day on who invented what and who invented what first. This is sometimes called the “calculus war.”

Independent of who invented what, they did use different notations, at least that is without controversy. Newton used the dot notation and Leibniz the differential notation. Thus Newton would write \dot{x} while Leibniz would write $\frac{dx}{dt}$ for the same thing. The clear winner, most agree, is the better notation of Leibniz. It is even claimed that the British mathematicians by sticking to Newton's poorer notation lagged behind the rest of Europe for decades.

- **Leonhard Euler** introduced many of the common symbols and notations we still use today. He used e and π for the famous constants and i for the square root of -1 . For summation Euler used Σ and also introduced the notation for functions: $f(x)$. One can look at some of his old papers and see equations and expressions that look quite modern—of course they are old, but look modern because we still use his notation.
- **Carl Gauss** introduced many ideas and proved many great theorems, but one of his most important contributions concerns the congruence notion. Leonhard Euler had earlier introduced the notion, but without Gauss' brilliant notation for congruences, they would not be so easy to work with. Writing

$$x \equiv y \pmod{m}$$

is just magic. It looks like an equation, can be manipulated like an equation—well almost—and is an immensely powerful notation.

- **Paul Dirac** introduced in 1939 his famous notation for vectors. His so-called *bra-ket* notation is used in quantum everything. It is a neat notation that takes some getting used to, but seems very powerful. I wonder why it is not used all through linear algebra. A simple example is:

$$|x\rangle = [a_0, a_1, \dots, a_n]^T.$$

The power of the notation is that x can be a symbol, expression, or even words that describe the value of a quantum state.

For a neater example, the outer-product of a vector R with itself, as used in the APL program for primes above, is written this way in Dirac notation:

$$|R\rangle\langle R|,$$

which flips around the inner product $\langle R|R\rangle$. Now to multiply the matrix formed by the outer product by a row vector a on the left and a column vector b on the right, we write

$$\langle a|R\rangle\langle R|b\rangle.$$

The bra-kets then associate to reveal that this is the same as multiplying two inner products. Another neat feature is that $\langle x|$ versus $|x\rangle$ captures the notion of a *dual vector*, and when x has complex entries, the $\langle x|$ notation implicitly complex-conjugates them.

- **Albert Einstein** introduced a [notation](#) to make his General Relativity equations more succinct. I have never used the notation, so I hope I can get it right. According to his rule, when an index variable appears twice in a single term, once as a superscript and once as a subscript, then it implies a summation over all possible values. For instance,

$$y = c_i x^i$$

is

$$y = \sum_{i=1}^3 c_i x^{(i)},$$

where $x^{(i)}$ are not powers but objects.

- **Dick Karp** introduced the notation $C/f(n)$ to complexity theory in our joint paper on the Karp–Lipton Theorem. Even though the notation was his invention, as his co-author I will take some ϵ amount of credit.

2.2 Good Notation?

Not all notation that we use is great; some may even be called “bad.” I would prefer to call these good with a question mark. Perhaps the power of notation is up to each individual to decide. In any event here are a few “good?” notations.

- **John von Neumann** was one of the great mathematicians of the last century, and helped invent the modern computer. He once introduced the notion

$$f(((x)))$$

where the number of parentheses modified the function f . It does not matter what they denoted, the notation could only be used by a brilliant mind like von Neumann’s. This notation is long gone as used by von Neumann, but in the theory of ideals, $((p))$ is different from (p) . Oh well.

- The letter π for pi is fine, but maybe it denotes the wrong number? Since 2π denotes the full unit circle and occurs all the time in physics, maybe we should have used the symbol π for that number? See a further discussion by Lance Fortnow and Bill Gasarch referenced in the end notes.
- Why is the charge of the electron negative? Evidently it is because Benjamin Franklin believed the flow of an unseen fluid was opposite to the direction in which the electron particles were actually going.
- Why has humanity been unable to establish that \subset means proper subset and only \subseteq means subset, by analogy to $<$ and \leq ? Hence for proper subset one often resorts to the inelegant \subsetneq notation.

- I will end with one example of notation that many feel strongly is only “good?": $f'(x)$ to denote the derivative of a function. This generated a lively [discussion](#) fifteen years ago on the long-running Drexel University Math Forum, which is also referenced in the end notes.

2.3 Open Problems

Does good notation help make mathematics easier? Are there some notions that are in need of some better notation? What do you think?

2.4 Notes and Links

Original post:

<http://rjlipton.wordpress.com/2010/11/30/notation-and-thinking/>

Brad McCormick’s APL page:

<http://www.users.cloud9.net/~bradmcc/APL.html>

Quotation on APL:

www-users.cs.york.ac.uk/~susan/cyc/p/prog.htm

Conway’s “Life” in APL:

<http://catpad.net/michael/apl/>

Book by Timothy Budd:

<http://web.engr.oregonstate.edu/~budd/Books/aplc/>

Equals sign reference:

http://en.wikipedia.org/wiki/Equals_sign

La Géométrie:

http://en.wikipedia.org/wiki/La_Geometrie

Leibniz–Newton controversy:

http://en.wikipedia.org/wiki/Leibniz_and_Newton_calculus_controversy

Gauss congruence notation:

http://en.wikipedia.org/wiki/Modular_arithmetic#Congruence_relation

Dirac bra-ket notation:

http://en.wikipedia.org/wiki/Bra-ket_notation

Einstein summation notation:

http://en.wikipedia.org/wiki/Einstein_notation

Pi discussion by Fortnow and Gasarch:

<http://blog.computationalcomplexity.org/2007/08/is-pi-defined-in-best-way.html>

Drexel Math Forum discussion on notation for derivatives:

<http://mathforum.org/kb/thread.jspa?threadID=33142&messageID=108756>

This post in *Gödel’s Lost Letter* had an exceedingly lively comment discussion.

Picture credits:

1. Conway’s Life: Screen capture from <http://catpad.net/michael/apl/>

Edmund Hillary was not a theoretician, but is still world-renowned for the first successful ascent and descent of Everest. He did this with Tenzing Norgay in 1953.

We will talk about the connection between mountain climbing and solving mathematical problems. By mountain climbing I mean climbing Everest or K-2 or one of the dozen other peaks that are over 8,000 meters high; by solving mathematical problems I mean solving hard open problems.

I believe that there are some interesting connections between these two vastly different tasks. Let me explain.

3.1 A Disclaimer

I want to start by saying I could never climb anything higher than a stepladder. Even that might be a stretch for me. Equally, I have solved some nice problems over the years, but I certainly am not one who has knocked off many great, or even lesser, open problems. However, I still think that I can say something about the relationship between these two. I hope you will agree.

3.2 Climbing

I would like to start by listing some facts about climbing that we believe are true. One of my favorite types of books to read are true stories of mountain climbing. If you like this genre, some of my favorites are:

- (1) *The Kid Who Climbed Everest*, by Bear Grylls.
- (2) *Into Thin Air: A Personal Account of the Mount Everest Disaster*, by Jon Krakauer.
- (3) *On the Ridge Between Life and Death: A Climbing Life Re-examined*, by David Roberts.
- (4) *Forever on the Mountain: The Truth Behind One of Mountaineering's Most Controversial and Mysterious Disasters*, by James M. Tabor.

- (5) *No Shortcuts to the Top: Climbing the World's 14 Highest Peaks*, by Ed Viesturs and David Roberts.
- (6) *K2: Life and Death on the World's Most Dangerous Mountain*, also by Viesturs and Roberts.
- **You cannot reach the top if you do not climb.** This sounds trivial but it still is true. If you do not try to climb the summit of K2, then you certainly will not reach it.
 - **Climbing requires a team.** Everest is a perfect example, as are most of the 8,000-meter peaks. There are some who have climbed solo, but the rule is that a great deal of work has to be done by a team to get even one person to the summit.
 - **Great climbers have multiple skills.** The best climbers need great technical skill, great mental toughness, careful planning, and luck. Ed Viesturs points out that mountains allow you to climb them; you cannot just attack and climb them if they do not wish to be climbed.
 - **Success requires reaching the summit and getting down.** George Mallory may have been the first to reach Everest's summit of 29,029 feet—but he never made it down.
 - **Know when to turn around and go down.** Great climbers know when the summit is out-of-reach. They know when they need to climb back down the mountain. Perhaps they will reach the summit another time, but they know it will not happen now.
 - **Never rely on another's rope.** The best climbers will check others' ropes very carefully before they use them. Many will almost never rely on any rope they have not placed themselves or by a trusted colleague.

3.3 Solving

- **You cannot reach the top if you do not climb.** *In solving this translates into:* you cannot solve an open problem if you do not think about the problem. One of the many themes that I have repeated many times is, in today's terms, get out there and try to climb a mountain.
- **Climbing requires a team.** *In solving this translates into:* the team in mathematics is those who have worked already on the open problem. Just as in climbing there are "solo" climbs, but even these have often relied on past climbers who proved that an approach could work. In problem solving we need to use the work that has come before. Also we need to work in teams: today many papers seem to have more and more authors. Even when the final paper is written by one mathematician, there are almost always others who helped. This was true with Andrew Wiles' solution of Fermat's Last Theorem; it was also true with Grigori Perelman's solution of the Poincaré Conjecture.
- **Success has several components.** *In solving this translates into:* clearly solving hard problems requires technical skill. It also requires mental toughness. Imagine the resolve that Wiles had in working alone on his problem for years—literally in his attic. The same for almost anyone who has solved an open problem. Finally,

just as in climbing there is, I believe, an element of “luck.” Some approaches that eventually worked on an open problem just make it—there may be a perfect cancellation of two terms, for example.

- **Success requires reaching the summit and getting down.** *In solving this translates into:* the lesson here is to solve the problem, but to also get “down.” That means—in my view—to get the solution written up and made public. There are many open problems that have been solved more than once. So by “getting down” I mean that you need to write up, and make your result accessible to others. There are many famous examples of the failure to do this. One of my favorites concerns the following pretty theorem from number theory that was conjectured previously by Carl Gauss:

Theorem 3.1 *There are only a finite number of imaginary quadratic fields that have unique factorization. They are \sqrt{d} for*

$$d \in \{-1, -2, -3, -7, -11, -19, -43, -67, -163\}.$$

This was proved first by Kurt Heegner in 1952, but the proof was thought to be flawed. The first accepted [proof](#) was by Harold Stark in 1967. Then, the Heegner proof was re-examined and discovered to be correct. Oh well.

- **Know when to turn around and go down.** *In solving this translates into:* know when to quit on a problem. Or at least when to put the problem aside. I have talked about mathematical diseases before: Good solvers know when their ideas are not going to work. Staying forever working on the same problem—that is, trying for the same summit—is a bad idea. In climbing it can get you killed, while in solving it will waste a great deal of time and energy.
- **Never rely on another’s rope.** *In solving this translates into:* if you are using another’s lemma or work, you need to understand their work. Of course if you are using Gauss’s Quadratic Reciprocity Theorem, then perhaps you can trust the result. But, if you are using a new result, or even a result that has been around for a little while, “checking the rope” is probably a good idea. I have personally wasted several months using a published result that was in a great journal, had been generalized by another, only to discover eventually that their “ropes” were frayed. In the end multiple papers had to be retracted—several papers were wrong. I wound up wasting a great deal of energy.

3.4 Open Problems

I hope you like the analogy to climbing. There are, of course, two big differences. First, when we fail, we do not get killed. Second, there are only fourteen mountains over 8,000 meters. In theory we seem to have an unending supply of great peaks to climb. Somehow as soon as a peak is scaled, another higher and harder one is unveiled.

Good luck, try and climb some high peaks, and be careful.

3.5 Notes and Links

Original post:

<http://rjlipton.wordpress.com/2010/02/02/climbing-mountains-and-proving-theorems/>

Stark–Heegner theorem:

http://en.wikipedia.org/wiki/Stark-Heegner_theorem

Leonardo da Vinci was not a complexity theorist, but according to our friends at [Wikipedia](#) he was just about everything else: a painter, a sculptor, an architect, a musician, a scientist, a mathematician, an engineer, an inventor, an anatomist, a geologist, a cartographer, a botanist, and a writer. If he were alive now, I wonder what brilliant things he would be doing.

We will discuss a question related to Leonardo as an artist: can we view great proofs as great art? I think we can, and I would like to explain why.

Connecting proofs and art reminds me of a legend I heard at Yale, many years ago, concerning the beginning programming class, CS 101. I do not believe the legend, but this will not stop me from repeating it. The legend goes there was once an exam given to a 101 class, containing this question:

Evaluate the following expression:

$$E + H \times E$$

where $E = 3$ and $H = 4$.

The correct answer was $15 = 3 + (4 \times 3)$, not $21 = (3 + 4) \times 3$, since the operator “ \times ” has higher precedence than the operator “ $+$ ”. But, the legend is one student “evaluated” the expression as follows:

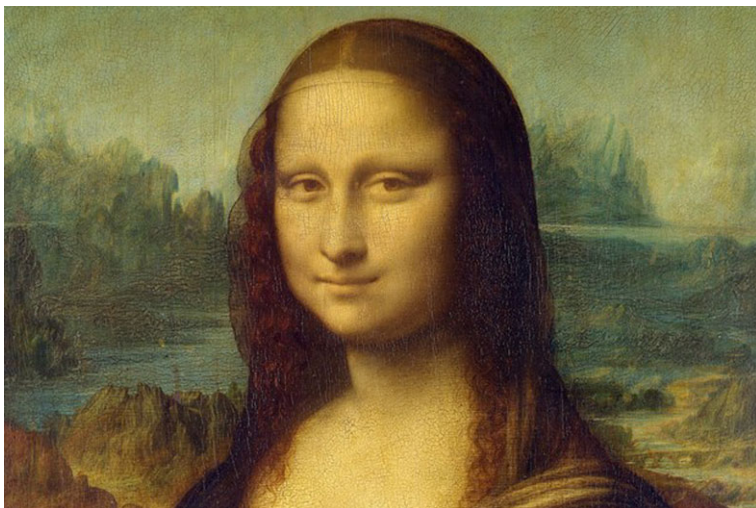
The expression “ $E + H \times E$ ” has a pleasing horizontal symmetry, and a near left-to-right central symmetry, and is also...

You must remember Yale was filled with very bright students, but their main interests were often in the liberal arts, not in mathematics. The student was right, the expression does have some nice symmetries. . .

4.1 Studying Great Art

How do we study great art? Consider the Mona Lisa as an example—perhaps the best-known painting in the entire world. We may, if in Paris, go to the Louvre and wait in a long line, and eventually get a glance at the famous painting. Or we may

study reproductions of the Mona Lisa in art books, in other works, or even online today.



Most of us look at the Mona Lisa and see different things. We may like it, we may love it, or we may prefer more modern paintings, but most are amazed at Leonardo's achievement.

Our goal in looking at and studying a great work of art like this is not to reproduce it ourselves—that is impossible. Our goal is not even to try to understand how Leonardo achieved his great painting, how he captured her famous smile—that also is probably impossible.

Instead our goal is one of enjoyment, of pleasure, of the excitement in seeing the work of a master. We come away with an impression, a general feel, a holistic view of what he did. We do not understand each brush stroke, we do not understand how he made his paints, we do not understand how he captured her look, we do not understand how he created the masterpiece. But, we do leave with something.

4.2 Studying Great Proofs

I have wondered if we can do something similar with the great proofs. Many read and study great proofs in an attempt to really understand them. If there is a masterpiece in your area of research you may need to understand the proof line for line—you may need to be able to reproduce the proof to be considered an expert in your area.

However, most of us do not even look at the masterpiece proofs, ever. We may know their statements, but most of us—I claim—have no idea how they work. Nor do we need to understand them line-for-line. But my thesis is this:

We can learn from a study of the great proofs, even if we do not follow the proofs in detail.

We should study the great proofs, not to understand them completely, but in the same way people study the Mona Lisa. For enjoyment, for enrichment, for seeing a master at work, and for getting something out of the study. Not to be able to reproduce the Mona Lisa. What we learn may be intangible, but still invaluable. Again, we can leave with something.

Ken adds that a similar consideration is well attested for improving one's chess. It is worthwhile to play over good games played by masters, even without stopping to analyze the crucial moves and alternative moves that were rejected. One can play over the games fairly quickly, letting the patterns and strategy sink in. Playing over lots of master games is thought to give more benefit than studying just a few of them deeply.

4.3 Some Great Proofs

Here are some sample great proofs—there are many others, these are just a few to make our discussion concrete. The same comments will apply to many other of the great proofs.

One property of many of the great proofs—not all—is they are long. The opposite is not true: there are plenty of long proofs of relatively unimportant results. Length is no measure of greatness, just as the length of a book or the size of a painting is not a measure of its greatness. However, many of the great proofs are long, indeed some are very long. One reason is that a great proof often requires several new ideas, and it is reasonable to expect that weaving these ideas together can be difficult to explain. This accounts for the substantial length of the great proofs.

- *Feit–Thompson's Theorem*: Walter Feit and John Thompson proved in 1962: Every group of odd order is solvable.
- *Cohen's Theorem*: Paul Cohen proved in 1962 that the Axiom of Choice and the Continuum Hypothesis are independent from ZF set theory. Obviously, 1962 was a very good year for great proofs.
- *Szemerédi's Theorem*: Endre Szemerédi proved in 1975: Every dense enough set of natural numbers has arbitrarily long arithmetic progressions.
- *Wiles' Theorem*: Andrew Wiles proved in 1994: Fermat's Last Theorem. Ken adds that I could have dated this 1987–1994, including the premature announcement in June 1993. However, the keystone insight by which Wiles bridged the gap in September 1994 is one of the great moments in the history of proofs. Wiles realized that the circumstances under which a previous attempt of his failed supplied the conditions needed for his present attempt to succeed. One or the other had to go through, and that was enough for a proof.

4.4 What Can One Learn from This Study?

I would like to run a class on the study of great proofs as great art. I would have students “read” each of the great proofs, and we would have discussions about them. I would not expect even the top students to be able to understand the proofs fully, or

even partially. I would not expect anyone to understand the proof at any deep level at all. They may, even should, understand parts of the proof completely—a lemma here, or an argument there.

I would expect students to learn some appreciation for what it takes to create a masterpiece—a great proof. I believe we can learn a great deal from reading great proofs, even without understanding them in the usual sense. Let's look and see what we might learn from reading these proofs in this way.

Previous Work Great proofs usually do not arise out of a complete vacuum. They are almost always based on previous work: they may extend an earlier argument, they may use techniques developed by others, they arise in a complex context.

Feit–Thompson: They used a great deal of machinery of group theory. One of the great previous results was due to William Burnside, who had proved every group of order $p^a q^b$ with p and q both prime is solvable.

Cohen: He used Kurt Gödel's famous proof of the relative consistency of the AC with ZF.

Weave Many Ideas Together Great proofs often need to use much of the existing machinery. This is similar to the last point, but a bit different. Here the point is the creation of a great proof usually requires the mastery of many tools and techniques.

Feit–Thompson: They used almost all aspects of group theory: local analysis, character theory, and definitions of groups via relations. They also needed to use non-trivial number theory. At the very end of their proof they could have saved a large amount of work if they could have proved this conjecture: there are no distinct primes p and q so that

$$\frac{p^q - 1}{p - 1} \text{ divides } \frac{q^p - 1}{q - 1}.$$

Since they could not, they needed to use even more group theory tricks to get the contradiction they needed.

Wiles: He used much of modern number theory, especially the theory of modular forms and Galois representation theory. The famous number theorist Enrico Bombieri said that he would need at least a full year to understand Wiles' achievement—this is because of the diverse tools used in the proof.

Golden Nuggets Great proofs often contain new ideas and methods, sometimes these can be more important than the theorem being proved. Note that I have talked all through this discussion about proofs not theorems. This is one reason for this: often the proof is much more important than the theorem itself.

Cohen: Cohen created an entire new way of constructing models of set theory. This method, called *forcing*, is now a standard tool used by set theorists everyday. When Alfred Tarski heard about Cohen's new method he said:

They have a method, now they will get everything.

Szemerédi: Szemerédi needed to prove a certain lemma to make his great proof work. This lemma is the now-famous Regularity Lemma.

Strategic Plan Great proofs almost always have a plan of attack. If someone tells you they have solved a long-standing open problem by “just an induction”—it is pretty unlikely to be true. Usually, great proofs have a complex strategic plan for the attack of their proof.

Feit–Thompson: Their proof is divided into six chapters: each has its own abstract, and they give a coherent overview of what each chapter does and how they all fit together.

CONTENTS

Chapter I, from Solvability of groups of odd order, Pacific J. Math., vol. 13, no. 3 (1963) 775–787

Walter Feit and John Griggs Thompson

Chapter II, from Solvability of groups of odd order, Pacific J. Math., vol. 13, no. 3 (1963) 789–802

Walter Feit and John Griggs Thompson

Chapter III, from Solvability of groups of odd order, Pacific J. Math., vol. 13, no. 3 (1963) 803–843

Walter Feit and John Griggs Thompson

Chapter IV, from Solvability of groups of odd order, Pacific J. Math., vol. 13, no. 3 (1963) 845–942

Walter Feit and John Griggs Thompson

Chapter V, from Solvability of groups of odd order, Pacific J. Math., vol. 13, no. 3 (1963) 943–1010

Walter Feit and John Griggs Thompson

Chapter VI, from Solvability of groups of odd order, Pacific J. Math., vol. 13, no. 3 (1963) 1011–1027

Walter Feit and John Griggs Thompson

Bibliography, from Solvability of groups of odd order, Pacific J. Math., vol. 13, no. 3 (1963) 1029

Walter Feit and John Griggs Thompson

Szemerédi: His proof had a road map: there is a figure showing the various lemmas and the inter-relationships.

4.5 Open Problems

What are the other great proofs that you view as works of art? What features, other than those mentioned, do you find often in such proofs?

Is this idea reasonable? Would you like to take such a course? Should I give this course sometime?

4.6 Notes and Links

Original post:

<http://rjlipton.wordpress.com/2010/04/14/great-proofs-as-great-art/>

Wikipedia da Vinci bio:

http://en.wikipedia.org/wiki/Leonardo_da_Vinci

Feit–Thompson paper:

[http://projecteuclid.org/DPubS?service=UI&version=1.0&verb=Display
&handle=euclid.pjm/1103053941](http://projecteuclid.org/DPubS?service=UI&version=1.0&verb=Display&handle=euclid.pjm/1103053941)

Cohen’s forcing method:

[http://en.wikipedia.org/wiki/Forcing_\(mathematics\)](http://en.wikipedia.org/wiki/Forcing_(mathematics))

Szemerédi’s Theorem:

http://en.wikipedia.org/wiki/Szemeredi's_theorem

Fermat’s Last Theorem:

http://en.wikipedia.org/wiki/Fermat's_Last_Theorem

Regularity Lemma:

http://en.wikipedia.org/wiki/Szemeredi_regularity_lemma

Picture credits:

Mona Lisa:

<http://www.wired.co.uk/news/archive/2013-01/18/mona-lisa-in-space>

http://www.nasa.gov/mission_pages/LRO/news/mona-lisa.html

Feit–Thompson: Screen capture from Pacific Journal of Mathematics website.

Michael Atiyah, actually Sir Michael Atiyah, is one of the great mathematicians in the world. He has received awards from the Fields Medal to the Abel Prize, for his seminal work in many aspects of algebraic geometry, topology, and operator theory. Besides his deep and beautiful results he has some striking insights into the nature of proof. For example, one of his [quotes](#) is:

I think it is said that Gauss had ten different proofs for the law of quadratic reciprocity. Any good theorem should have several proofs, the more the better. For two reasons: usually, different proofs have different strengths and weaknesses, and they generalise in different directions—they are not just repetitions of each other.

We will talk about the $P = NP$ question, and not even mention the major and minor claims to prove them different—or equal—that arose during 2010.

At the time of the major proof claim by Vinay Deolalikar, Ken and I both spoke to various reporters. Some were from the online media and some from the print media. Both types of reporters asked good questions, and in general we had an interesting discussion. But one or two asked a question that I really had trouble answering. The question was not an obvious one like: what is polynomial time, or what is NP? The question was:

Why is proving $P \neq NP$ an important result?

The trouble I had is simple: if most believe that $P \neq NP$ is true—perhaps obviously true—then why care if it gets proved? Yes, it is a famous problem, with a large monetary prize. No doubt whoever first proves the result will be showered with many awards and honors. But, still why the huge interest in proving something that we know is correct?

I have thought about this quite a bit, and have some insights that I would like to share with you about their question. I wonder if you have other thoughts.

5.1 Does Any Proof Matter?

So why do we really care if there is suddenly a proof that $P \neq NP$? As many of you know, I am less sure than most that $P \neq NP$. So perhaps the answer *for me* is it would be important because I have great doubts about it. But, I have already discussed a number of times and in different ways why I am skeptical about the conventional wisdom.

I can think of several different reasons why a proof of $P \neq NP$ would potentially really matter. The first has to do with the nature of proofs in mathematics. Let me explain.

I had the honor of meeting Atiyah a few years ago, and we had a chance to talk about the nature of proofs. One story he told me in private was quite telling. He told me that he had, many years earlier, proved a technical theorem about finite groups. He felt very sure the proof was correct, but sometimes late at night he would lie awake with some doubts. The proof was not a high-level proof; instead it relied on a detailed analysis of the group structure. Since the proof was so technical and filled with case analysis he never felt that he really knew why the theorem was true.

Years later he found *the proof*. He realized that his theorem was true for a much larger class of groups than finite: it was true for compact algebraic groups. Further, the proof there was high-level, was clear to an expert, and relied on no magic calculation, nor any case analysis. He said now he *knew* the original theorem was correct, and he could sleep better at night.

He further said that the role of proof, in his opinion, is not to “check-off” that a statement is correct. The role is to give insight into *why* the statement is correct. As you might imagine he was not very interested in machine proofs—at the time we discussed Thomas Hales’ [proof](#) of the Johannes Kepler Conjecture. While he understood the potential need for such computer proofs, he really wanted to know why it was true.

5.2 Does a Proof of $P \neq NP$ Matter?

Yes it does. Here are my three foremost reasons to think that such a proof could be very important.

- **A Proof Would Tell Why:** Even those who are sure that $P \neq NP$ would like to know why this is so. This is exactly Atiyah’s point. A proof would give us insight into why there can be no efficient search for SAT.
- **A Proof Could Give Us New Methods:** Perhaps the best reason is the hope that a proof that $P \neq NP$ would have to use new tools in the proof. These tools would hopefully shed light on computation in general. They could yield insights into the fundamental nature of computation. This is the best reason, in my opinion, for wanting a proof.

There are many examples in mathematics where this is exactly what has happened. The proof of a great result has many times created new tools and techniques that have raised the curtain and allowed us to see for the first time new insights into

other problems. Certainly it would be wonderful if this were the case with a proof of $P \neq NP$.

For example, Andrew Wiles' proof of Fermat's Last Theorem and Grigori Perelman's proof of the Poincaré Conjecture have changed their respective fields. Wiles' proof of Fermat has led to other fundamental advances in number theory, well beyond solving the one famous family of Diophantine equations.

However, not all mathematical proofs of great conjectures use new techniques. There have been many solutions of longstanding open problems that used *no* new techniques. These were often very clever results, but the provers did not need new methods and tools. They were able to resolve the conjecture using well-known methods—their proofs may have been very clever, but they did not change the landscape.

I have no idea how often the latter happens, but it does happen. Some friends who are experts on the Riemann Hypothesis once told me their greatest fear is: what if someone came along with a clever proof, but one that “just” used known technology in a clever way? They would then know that the Riemann Hypothesis is true, yet they would be quite disappointed.

Hopefully, this will not happen with $P \neq NP$. Hopefully.

- **A Proof Helps With Goals of Security:** In many cases this means *only* that the authors of a paper have proved that breaking their system implies that some hardness assumption, such as on factoring, is wrong. Even a proof that $P \neq NP$ would not rule out that such assumptions are false: recall that factoring is not known to be NP-complete. I think, however, that a proof of at least $P \neq NP$ would be of some comfort to cryptographers. Their special hardness assumptions might still be unproved, but a proof would move us closer to perhaps one day really having provable security.

5.3 Open Problems

Are these good reasons? What are your reasons?

5.4 Notes and Links

Original post:

<http://rjlipton.wordpress.com/2010/08/18/proofs-proofs-who-needs-proofs/>

Kepler Conjecture:

http://en.wikipedia.org/wiki/Kepler_conjecture

See Chap. 1 for more on Vinay Deolalikar's claimed proof. The first post on our blog about it was:

<http://rjlipton.wordpress.com/2010/08/08/a-proof-that-p-is-not-equal-to-np/>

Subhash Khot is one of the great young complexity theorists in the world. He is perhaps best known for his wonderful conjecture—the Unique Games Conjecture. But also he has worked in other areas, and solved many other important problems. He won the 2010 Alan T. Waterman Award. It is named after the first Director of the NSF, and was created in 1975. Subhash joins an array of famous mathematicians winning this award, including:

- Charles Fefferman, 1976;
- William Thurston, 1979;
- Harvey Friedman, 1984;
- Herbert Edelsbrunner 1991;
- Gang Tian, 1994;
- Emmanuel Candes, 2006; and
- Terence Tao, 2008.

This is a wonderful list.

We must visit Khot’s Unique Games Conjecture, and survey some recent results on his conjecture.

John Cherniavsky, Senior Advisor for Research at NSF and a longtime friend of mine, told me that Subhash’s talk at NSF for his prize was stellar. John added,

“It might be appropriate to blog about his Unique Games Conjecture work—which seems very relevant to your blog.”

I agree. So here is my view of the conjecture in three acts.

6.1 Act I: The Unique Games Conjecture

Khot’s conjecture is remarkable on many levels. It is simple to state, yet it has created a wealth of important ideas and results. Perhaps stating a great conjecture—whether true or false—is one of the best ways to advance any field. Steve Cook and Dick Karp have the $P \neq NP$ question, Juris Hartmanis the Isomorphism Conjecture, Leslie Valiant the Permanent Versus Determinant Conjecture, and so on. Now we have the Unique Games Conjecture of Khot.

Many of you probably know the [Unique Games Conjecture](#), but I will give an informal definition of it anyway. I think of it as a generalization—a very clever one—of the simple problem of 2-coloring a graph. Suppose G is an undirected connected graph. There is a linear-time algorithm for testing whether the graph is 2-colorable—you probably saw it in Computing 101:

- (1) Pick any starting vertex v .
- (2) Color v **red**.
- (3) Select any vertex u that is yet uncolored and is adjacent to a vertex that is colored. If u is adjacent to a **red** vertex, then color it **green**; if it is adjacent to a **green** vertex, then color it **red**.
- (4) Continue until all the vertices are colored.
- (5) The graph is 2-colorable if and only if there are no conflicts when all vertices are colored.

Khot's idea was to change the notion of 2-colorable in several simple ways. First, he allows a fixed but arbitrary set of colors—finite of course. This sounds like it might become the general coloring problem, but his brilliance is to pull back and make it closer to 2-coloring. He notes that the essence of 2-coloring is the rule:

If a vertex v is **red**, then any neighbor is **green**.

Khot allows each edge to have its own rule, provided it obeys the following deterministic condition. If v and u are adjacent, then the color of one must **uniquely** determine the color of the other.

Generally a graph with rules restricting allowed values for the vertices on each edge is called a *constraint graph*—but the uniqueness makes these graphs special. It is still easy to tell whether or not such a graph can be colored so that all edge rules are followed. Just take the 2-coloring algorithm and modify it slightly:

- (1) Pick any starting vertex v and a color c .
- (2) Color v with the color c .
- (3) Select any vertex u that is yet uncolored and is adjacent to a vertex v that is colored. Use the edge rule to color u . Note, there is **no** choice here once v is selected.
- (4) Continue until all the vertices are colored.
- (5) If all the edge rules are satisfied the graph is satisfiable. If not go back to step (1) and try another color c' . If all the colors have been tried, then the graph is not satisfiable.

This still takes only linear time, and still lies within Computing 101.

So far we have an easy problem—what is all the excitement? Why is this so important that Subhash was awarded the Waterman Prize? Khot adds one final ingredient: *approximation*. Suppose G is again a unique-constraint graph, but now I *promise* that one of two situations is true:

- there **exists** an assignment of colors to the vertices of G so at least $1 - \varepsilon$ of the edges are satisfied; or
- there is **no** assignment of colors to the vertices of G so more than ε edges are satisfied.

Telling which case is true is the nub of the Unique Games Conjecture (UGC). Khot conjectured it is NP-hard to tell which is true, for large enough sets of colors and all constraint graphs. Specifically, his conjecture states that for all $\varepsilon > 0$ there is an $R > 0$ such that if some polynomial-time algorithm distinguishes the above two situations for all unique-constraint graphs with R colors, then $P = NP$. The point is that allowing some edges to fail destroys the above linear-time algorithm. Indeed, tolerating εn failures could lead to $2^{\varepsilon n}$ amount of backtracking.

6.2 Act II: The Conjecture's Applications

The beauty of the problem is the power that goes with its simplicity. The ability to let each edge have its own rule allows many problems to be encoded into a UG problem. Since Khot made his [conjecture](#) in 2002 there have been many papers proving theorems of the form:

Theorem 6.1 *Obtaining a Y -approximation to problem X is as hard as solving the Unique Games problem.*

One of the best examples is the famous [algorithm](#) of Michel Goemans and David Williamson for maximum cut of a graph. If Khot's UGC is true, then their algorithm would be essentially the best one can hope for. That connection alone is a pretty neat result.

6.3 Act III: Is It True?

As you probably know I am unsure of the answer to $P \neq NP$, so I have similar thoughts about the Unique Games Conjecture. Perhaps my doubts are stronger since it is unknown whether or not solving unique-games instances is NP-hard. I do believe whether it is eventually shown to be false or not changes little about the brilliance of the conjecture. Mathematics is moved forward by great conjectures, and Khot has certainly done this for computational complexity. It takes insight, creativity of the highest order, and a bit of "guts" to make such a conjecture.

However, in recent years there have been a series of results showing that unique-games problems *may* not be so difficult. The ideas in these beautiful papers would not have existed without the conjecture, and the techniques used may be used to solve other problems—no matter what eventually happens to the conjecture.

I have neither the expertise nor time right now to give an exhaustive list of the results that have been chipping away at Khot's original hardness assertion. I would like to mention one direction in detail, and then just state the other more recent ones.

The first direction was a series of results by many to show that solving unique-games problems on expander graphs is easy. This is work of many, including Sanjeev Arora, Subhash Khot, Alexandra Kolla, David Steurer, Madhur Tulsiani, and Nisheeth Vishnoi. I apologize for not listing all, but a later expansion of the story may do justice to all.

A sample [result](#) is due to Konstantin and Yuri Makarychev:

Theorem 6.2 *There exists a polynomial-time approximation algorithm that, given a $(1 - \varepsilon)$ -satisfiable instance of Unique Games on a d -expander graph G with $\varepsilon/\lambda_G \leq c_1$, finds a solution of cost*

$$1 - c_2 \frac{\varepsilon}{h_G},$$

where c_1 and c_2 are some positive absolute constants.

Note that h_G is the normalized edge expansion constant, while λ_G is the smallest positive eigenvalue of the normalized Laplacian of the graph G . Alexandra Kolla had similar results, but as I said earlier I cannot state and compare all the known results.

There is even more recent work due to Arora, Boaz Barak, and Steurer (ABS) that could be the start of the end of the UGC. They show, roughly, that any unique-games problem can be solved in time

$$2^{n^\varepsilon}$$

for any $\varepsilon > 0$. This does not rule out unique-games instances being NP-hard, but it certainly shows if the conjecture is true there would be great consequences.

The ABS paper shows that if the conjecture is true then either some famous NP-hard problems have subexponential time algorithms, or any reduction showing NP-hardness of some general set of unique-games instances must take time with a polynomial whose exponent depends on the ε parameter, which seems to defeat local-gadget reductions used for other approximation problems. Absolutely ABS shows that there are smarter ways of solving unique-games problems than simple backtracking—the result shows that a general divide-and-conquer paradigm can be employed. Their result contains a new insight into the structure of any graph, one that could have far-ranging consequences beyond the important application to the Unique Games Conjecture.

6.4 A Comment on Expanders

I believe the bounds for playing Unique Games on expanders are a bit misleading. They are right, they are deep, and they are important. But, unless I am confused—always a possibility—we need to be careful in reading too much practicality into these results.

This is my concern. Let G be a d -regular graph. Then, there is a trivial method to always find a solution to a unique-games problem with $1/(d + 1)$ fraction of the edges satisfied. By Brooks' [Theorem](#) such a graph has a $d + 1$ edge coloring: pick the most common color and these edges can be satisfied. Thus, if we are trying to

separate a $1 - \varepsilon$ from ε fraction of the edges, the promise problem is meaningful only for $\varepsilon > 1/(d + 1)$.

Thus, if we are trying to separate a $1 - \varepsilon$ from ε fraction of the edges, $\varepsilon > 1/(d + 1)$. The theorem by Makarychev and Makarychev has a large constant $c_2 \approx 100$, and

$$1 - c_2 \frac{\varepsilon}{h_G} > \varepsilon$$

forces the degree d to be very large. Unless I am wrong, if the graph is a spectral expander, then d must be order 10,000—and in any case it must be order 100.

These bounds are not extremely large by theory standards, but they do leave open the question of what happens on expander graphs of modest degree d . This is perhaps a small point, but one I find interesting.

6.5 Open Problems

What happens when we try to solve the UG problem on degree-3 graphs? What values of ε versus $1 - \varepsilon$ are distinguishable? What happens on other graphs of low degree? The last word may be due to [Oded Goldreich](#):

I'm happy to see yet another application of the paradigm of decomposing any graph into parts that are rapidly mixing, while omitting relatively few edges. Let me seize this opportunity and mention the application of this paradigm in the bipartite tester for bounded-degree graphs.

He was referring to a paper by Jonathan Kelner and Aleksander Mądry on generating random spanning trees, but this could apply to the Arora–Barak–Steurer breakthrough. Perhaps graph property testing techniques can shed light on the Unique Games Conjecture.

6.6 Notes and Links

Original post:

<http://rjlipton.wordpress.com/2010/05/05/unique-games-a-three-act-play/>

Survey by Khot:

<http://cs.nyu.edu/~khot/papers/UGCSurvey.pdf>

Wikipedia reference on the conjecture:

http://en.wikipedia.org/wiki/Unique_games_conjecture

Semidefinite programming algorithm:

http://en.wikipedia.org/wiki/Semidefinite_programming#Examples

Paper by Makarychev and Makarychev:

<http://eccc.hpi-web.de/eccc-reports/2009/TR09-021/Paper.pdf>

Brooks' Theorem:

http://en.wikipedia.org/wiki/Brooks'_theorem

Goldreich source:

<http://www.wisdom.weizmann.ac.il/~oded/MC/044.html>

Arno van den Essen is one of the world experts on the Jacobian Conjecture. He has written a *book* on the problem—really **the** book, titled *Polynomial Automorphisms*. Besides this he has made many contributions to the understanding of this great conjecture.

We will talk about a paper Essen put into the archives on the Jacobian Conjecture (JC), and which we are discussing further at press time. His paper is amazing—really amazing. Its title is “The Amazing Image Conjecture” (referenced in the end notes).

Years ago Essen, Evangelos Markakis, and I wrote a simple paper on an approach to the JC, titled “Some Remarks on the Jacobian Conjecture and Connections with Hilbert’s Irreducibility Theorem” (referenced in end notes). I still believe the [paper](#) is a viable approach to the problem. But we could only prove a very weak result about the JC problem.

Since we could not make our ideas work, let’s turn to Essen’s new paper.

7.1 The Jacobian Conjecture

I thought you might like to go through a possible scenario of how the JC was created, and see some of the rationale behind the conjecture. This is not a “real” history, but one that I hope helps expose the ideas that go into the creation of a great conjecture.

Imagine that you are studying mappings F from the two-dimensional Euclidean plane to itself. A natural question is: when is F one-to-one? There are many properties that mappings may or may not have, but being one-to-one is a basic property. Such functions are called *injective*—this name is due to [Nicolas Bourbaki](#).

The first thing you notice is that F being injective is a “global” property, since F is one-to-one, or injective, provided there are no two distinct points a and b in the plane so that

$$F(a) = F(b).$$

Determining whether or not there are such distinct points is equivalent to asking for the solvability to the two equations

$$f(a_1, a_2) = f(b_1, b_2),$$

$$g(a_1, a_2) = g(b_1, b_2)$$

where $(a_1, a_2) \neq (b_1, b_2)$ and $F(x, y) = (f(x, y), g(x, y))$.

In general deciding whether such equations have a solution can be very difficult. In order to have any hope of solving this problem you decide to restrict the function F to be a polynomial mapping: this means that $f(x, y)$ and $g(x, y)$ are both polynomials.

Even with the restriction to polynomial maps the difficulty is that a and b can be anywhere in the plane; they need not be “near” each other. This global nature is the key reason that determining whether or not F is injective is hard.

You realize one way to simplify the problem is to ask a much weaker question: is the mapping F “locally injective”? The mapping F is *locally injective* if for each point p there is a disk around p so that for all a and b in the disk if $F(a) = F(b)$, then $a = b$.

There are two reasons you think this is a good move. First, the mapping F cannot be injective unless it is locally injective at every point p . This is a simple idea, but a powerful one. When facing a tough question, often relaxing the question can yield insight.

Second, you recall a classic theorem that has a sufficient condition for a smooth map to be locally injective at a point. The theorem is the famous *Inverse Function Theorem*. In one dimension it says that a function h is locally injective at point p , provided

$$f'(p) \neq 0.$$

For example, $h(x) = x^2$ is locally injective at any point $p \neq 0$.

The Inverse Function Theorem generalizes this simple idea from elementary calculus to two dimensions and beyond. The critical question is what takes the role of the derivative of a function? It turns out the role is played by the *Jacobian matrix* of the mapping F —clearly we are close to why the conjecture is called the JC.

Associated with any mapping smooth F is the Jacobian matrix defined as

$$J_F(x, y) = \begin{bmatrix} f_x(x, y) & f_y(x, y) \\ g_x(x, y) & g_y(x, y) \end{bmatrix}.$$

Note, $f_x(x, y)$ is the partial derivative of the function f . In the one-dimensional case this would degenerate to the 1-by-1 matrix of $f_x(x) = f'(x)$. If the determinant of this matrix is non-zero at the point p , then by the famous Inverse Function Theorem, the map F is locally injective at p .

Thus to tell if the mapping F is locally injective at every point you need only look at the determinant of the matrix $J_F(x, y)$. Note this is a polynomial in the variables x and y . It can be a high degree polynomial, and over the reals it could be quite difficult to tell if this polynomial is or is not ever 0. For example, does

$$17x^3 - 348xy^6 + x^7 - 89x^2y^4 + 12 = 0$$

have a solution over the reals?

Now you make a move that is brilliant. Rather than try to figure out whether such polynomials have zeroes or not you realize that the polynomial c , where c is a non-zero constant, has no zeroes. So you make the “simplifying” conjecture:

Conjecture 7.1 *If F has $\det(J_F(x, y)) = c$ where c is a non-zero constant, then F is globally injective.*

This is the *Jacobian Conjecture* due to Ott-Heinrich Keller in 1939, later given its modern name by Shreeram Abhyankar.

You might ask why not replace the condition $\det(J_F(x, y))$ is a non-zero constant by the condition $\det(J_F(x, y))$ is never zero for any real numbers. The answer is surprising—it is false. This was the so-called “Real Jacobian Conjecture,” which is now known to be false. Sergey Pinchuk refuted it in his paper, “A Counterexample to the Strong Real Jacobian Conjecture” (referenced in the end notes).

I will not go into more details, but the intuition I believe is that many real analysis problems are only understood when complex numbers are allowed. This happens in understanding the singularities of rational functions. For example the reason the power series of $1/(1+x^2)$ has radius of convergence only 1 is there is a complex singularity. In the reals this function has no poles, while over the complex numbers it has a pole at i .

7.2 JC Approaches

There have been many claims of proofs of JC. Some are still active claims. The incorrect proofs of JC—I think all have been proofs not counterexamples—have been by amateurs and professionals alike. The problem is slippery. There seem to be many difficulties in understanding the problem. Yet the problem’s statement only requires basic calculus.

JC seems to be one of those problems that should be resolvable. Yet so far it has withstood the attacks of mathematicians for over 70 years. Since then there have been many partial results and reductions, but no proof yet.

- **Equivalent Problems:** Like many problems in mathematics there are numerous equivalent statements. See Essen’s book for many examples. One of the most famous is to reduce the degree of mapping F . In order to do this the problem is generalized to higher dimensions: the map F is allowed to be

$$F(x) = (f_1(x), \dots, f_n(x))$$

where $x = (x_1, \dots, x_n)$. Hyman Bass, Edwin Connell, and David Wright proved the following famous theorem in their 1982 paper, “The Jacobian Conjecture: Reduction of Degree and Formal Expansion of the Inverse” (referenced in the end notes):

Theorem 7.1 *If the generalized JC is true for all n and polynomial maps of degree at most three, then JC is true.*

As often is the case, two is special. The general JC is easily proved for mappings of degree at most two.

- **Direct Proofs Attempts:** One “obvious” idea is to try and prove an inductive statement. If $F = (f, g)$ is the mapping, then prove that there is a transformation of F that preserves injectivity and lowers the degree. A claim a few years ago was based roughly on this type of approach, but it fell apart.
- **Advanced Attempts:** There were approaches to the JC problem based on advanced ideas of algebra. I am the last one to discuss these, but while they look possible they have not yet worked.
- **Algebra or Analysis:** There are other approaches based on analysis rather than algebra. Of course all methods are allowed, but I have often thought the problem is closer to number theory than algebra, again who knows.

7.3 Amazing Conjectures

Essen’s [amazing paper](#) describes a beautiful connection recently discovered by Wenhua Zhao between the JC and some striking questions that seem interesting on their own. These are a new set of extremely interesting conjectures. I will explain one of them—see Essen’s well-written paper for the full set of conjectures.

The generic class of open problems concern particular linear functionals L on spaces of polynomials. Call L “nice” if the only polynomial f such that $L(f^m) = 0$ for all $m \geq 1$ is the zero polynomial. The amazing fact is that there exist particular functionals L such that:

if L is nice, then JC is true.

These functionals are still fairly complicated to define, so here I will follow Essen’s paper by starting with three simpler functionals for which niceness is non-trivial—and still open for the third:

$$\begin{aligned}
 A(f) &= \int_0^1 f(x) dx \\
 B(f) &= \int_0^\infty f(x) e^{-x} dx \\
 C(f) &= \int_0^\infty \cdots \int_0^\infty f(x_1, \dots, x_n) e^{-(x_1 + \cdots + x_n)} dx_1 \dots dx_n
 \end{aligned}$$

Each of A, B, C sends a polynomial to a scalar. The questions are:

- (1) If $A(f^m) = 0$ for all integers $m \geq 1$, must $f = 0$?
- (2) If $B(f^m) = 0$ for all integers $m \geq 1$, must $f = 0$?
- (3) If $C(f^m) = 0$ for all integers $m \geq 1$, must $f = 0$?

Essen can prove the first two are true for univariate polynomials, but the third about $C(f^m) = 0$ is open for multivariate polynomials.

I will give a bit of the flavor of how he proves such theorems, but as always see his paper for the details.

Consider the polynomial

$$f(x) = c_n x^n + \cdots + c_l x^l$$

where $l > 0$ and each coefficient is a Gaussian integer. I am assuming that the constant term is zero, and plan on showing the lowest term is also zero. This will inductively prove that f is zero.

The key is to raise the polynomial to a special power m , which is selected so all the terms but the last “disappear” when B is applied. Let $m = (p - 1)/l$ where p is a prime and l divides $p - 1$. Then, f^m is congruent to

$$g(x) = h(x) + c_l^m x^{ml}$$

modulo p . Define d to be the minimum degree of the terms of $h(x)$. Then d is at least

$$lm + 1 = p.$$

Thus $B(f^m)$ is congruent modulo p to

$$c_l^m (p - 1)!,$$

since all the terms in $h(x)$ have degree at least p . It is easy to prove that B maps

$$x^r \rightarrow r!.$$

It follows that c_l^m is divisible by p , and that $c_l = 0$. The latter uses that c_l is a Gaussian integer.

7.4 Open Problems

Of course *the* open question is the JC. Is it true? This problem is not a Clay problem—perhaps it should be added to replace the Poincaré Conjecture. The JC problem is on Steve Smale’s noted [list](#) of open problems as Problem 17.

7.5 Notes and Links

Original post:

<http://rjlipton.wordpress.com/2010/07/17/an-amazing-paper/>

Arno van den Essen: *Polynomial Automorphisms*, Birkhäuser, Basel, 2000:

<http://www.infibeam.com/Books/info/Arno-Van-Den-Essen/Polynomial-Automorphisms/3764363509.htm>

Arno van den Essen, “The Amazing Image Conjecture,” arXiv.org 2010:

<http://arxiv.org/abs/1006.5801>

Arno van den Essen, Evangelos Markakis, and Richard Lipton, “Some Remarks on the Jacobian Conjecture and Connections with Hilbert’s Irreducibility Theorem,” 2005, available at:

<http://www.cs.toronto.edu/~vangelis/research/jac14.pdf>

Bourbaki entry:

<http://jeff560.tripod.com/i.html>

Inverse function theorem:

http://en.wikipedia.org/wiki/Inverse_function_theorem

Sergey Pinchuk, “A counterexample to the Strong Real Jacobian Conjecture,” *Mathematische Zeitschrift* **217**(1) (1994), 1–4, available at:

<http://www.springerlink.com/content/t6q81m94056v367m/>

Hyman Bass, Edwin Connell, and David Wright, “The Jacobian Conjecture: Reduction of degree and formal expansion of the inverse,” *Bull. Amer. Math. Soc.* **7**(2) (1982), 287–330, available at:

<http://projecteuclid.org/DPubS?service=UI&version=1.0&verb=Display&handle=euclid.bams/1183549636>

Smale’s problems:

http://en.wikipedia.org/wiki/Smale's_problems

Richard Hamilton is not a computer theorist, but is a senior mathematician at Columbia University. He is a leading expert in geometric analysis, and the creator of the program that helped solve the Poincaré Conjecture. He is a winner of numerous awards including the Veblen Prize in Geometry, a Clay Research Award, and most recently the AMS Steele Prize for Seminal Contribution to Research.

We shall discuss a question that has been raised here and elsewhere: should researchers share their ideas? Especially ideas that are not “complete.”

Every time you talk to a friend about a result, give a talk, or post a paper—let alone publish one—you are sharing your ideas. Most would argue that this is the key to advancing science in all areas. Yet there is the issue of *when* do you share the ideas. Do you wait until they are completely worked out—and what does that mean anyway? No matter what your result is, it is likely not to be the last word on your area. At least you should hope not. The last paper is a bit like: would the last person to leave . . . please turn out the lights.

On the other end of “when” is this advice from Harvard’s great computing pioneer, Howard Aiken:

“Don’t worry about people stealing an idea. If it’s original, you will have to ram it down their throats.”

8.1 Fermat’s Last Theorem

The proof of this great theorem, after the problem was unsolved for over 350 years, is due to Andrew Wiles—with some key help from Richard Taylor. But Wiles would have never been able to even start working on the proof without an insight of Gerhard Frey.

Frey gave a public lecture—are there private lectures?—where he pointed out some strange consequences of a non-trivial solution to

$$x^p + y^p + z^p = 0$$

for p an odd prime. Frey stated that such a solution would “likely” violate known conjectures. The “likely” was made in a precise statement by Jean-Pierre Serre, which was then proved by Ken Ribet. The missing piece had been called the “epsilon conjecture,” but the “epsilon” moniker was not because it was a trivial or small step.

The key moral I draw is that the community needed to solve the problem. Yes Wiles worked alone for almost a decade on his solution. But without Frey and then Serre and Ribet, he probably would not even have started. Frey’s ideas were a perfect example of the type of **partial results** that can drive us all toward the solution of great problems.

8.2 Applying the Idea for a Partial Result

I cannot even attempt here, or anywhere, an outline of Wiles proof. I think I can give an idea of how Frey’s idea can be used in a more modest way. The essential idea will start out like his idea: if there is a solution, then we will create a strange mathematical object. This object in our case will be a simple quadratic equation—the same kind you probably studied in high school. But the equation is “strange” in a way that will eventually lead to a contradiction.

Suppose that there is a non-trivial solution to the cubic case of Fermat’s Last Theorem. Let

$$a^3 + b^3 = c^3.$$

Then there are rational $x = a/c$ and $y = b/c$ so that

$$x^3 + y^3 = 1,$$

and xy is non-zero. The simple idea is to construct a mathematical object that is too strange, in some sense, and then prove that it cannot exist. The object will be just a quadratic equation: consider the polynomial

$$(z - x^3)(z - y^3).$$

Define $t = xy$. Then it is easy to check that the quadratic equation

$$z^2 - z + t^3 = 0$$

has solutions $\{x^3, y^3\}$. Therefore $\sqrt{\Delta} = \sqrt{1 - 4t^3}$ must be a rational number too. In other words the curve $u^2 = 1 - 4t^3$ has a rational point. Moreover, $U = u/2$, $T = -t$ gives that $U^2 = T^3 + 1/4$, which must also have a rational point.

Okay so what does this do for us? The point is that the equation

$$U^2 = T^3 + 1/4$$

is an **elliptic curve**: a class of curves that are very well [studied](#). There is a well-known (to the experts) theory that allows one to prove that the only rational points are (U, T) in $(1/2, 0)$ and $(-1/2, 0)$, which translates to give $xy = 0$. Thus this proves FLT in the cubic case.

8.3 Poincaré Conjecture

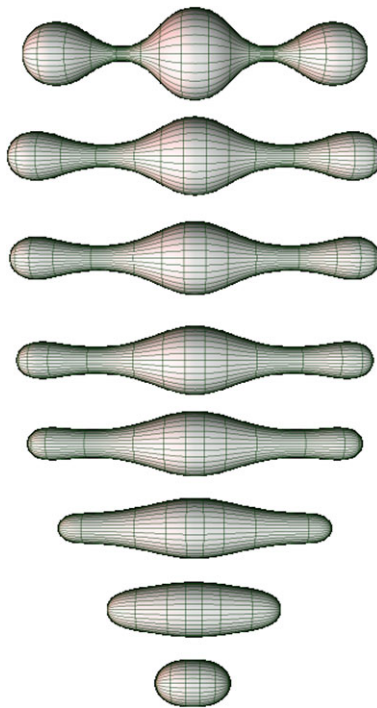
The proof of the Poincaré Conjecture by Grigori Perelman was one of the great results of mathematics. He carried forward to a conclusion a program that had been started by Richard Hamilton in 1982. Hamilton had proved a number of beautiful partial results, but the full proof needed Perelman.

Hamilton's brilliant idea was to replace a static problem by a dynamic one. The standard statement of the conjecture is:

Every simply connected, closed 3-manifold is homeomorphic to the 3-sphere.

The "obvious" approach is to take such a manifold and try to prove that it is a sphere. All earlier attempts failed.

What Hamilton did was to introduce a flow, the so-called Ricci flow, which changes a manifold over time. The idea of his method was then to prove that this flow made the manifold move toward a sphere over time. He could not prove this in general—he knew the behavior was more complex. But Hamilton did prove some important special cases. See the following figure for an [example](#) of the flow in action:



Note how the manifold changes over time, and becomes closer to a sphere.

8.4 Unique Games Conjecture

In Chap. 6 we discussed the famous Unique Games (UG) Conjecture of Subhash Khot. A 2010 paper of Sanjeev Arora, Boaz Barak, and David Steurer (ABS) has given new reason to doubt it. The rough notion is that a UG instance is a certain graph edge-coloring problem, of a kind that Subhash conjectured cannot even be approximately solved in polynomial time. The usual statement is: given a UG instance that has a solution that satisfies $1 - \varepsilon$ edges, even finding an edge coloring that satisfies $\delta > 0$ fraction of the edges is difficult. The conjecture has been used to prove that many other interesting problems also have no good approximation.

The paper of ABS proves that there is a sub-exponential-time algorithm for every UG instance. This does not rule out Subhash's conjecture, but does raise some issues. One possibility is that the conjecture is wrong; another is that the conjecture is true, but the reductions that would establish it are more complex than usual. We will see in time what happens.

Just as with FLT and Poincaré there were papers and ideas that helped guide ABS to their terrific result. Since this area is still in-progress, even three years later, I cannot say anything as definitive as I could for the other problems. However, it is clear that ABS built their paper and proof on previous work of a number of researchers. In their paper they thank, among others, Alexandra Kolla for an early copy of her manuscript. She proved some results that seem to have pointed the way for ABS.

Kolla's main theorem is roughly the following: For every $1 - \varepsilon$ satisfiable UG instance that obeys certain technical assumptions, one can find an assignment that satisfies more than 90 percent of the constraints in time that depends on the spectral profile of the game. She also shows how spectral methods can solve in quasi-polynomial time game instances that were previously thought to be hard, and how to solve Unique Games on expander graphs.

8.5 Open Problems

Should researchers be encouraged to share ideas earlier than we do now? What mechanisms are needed to be sure that proper credit is given? Would you publish a partial result?

I must say that one thing I have tried here is to state approaches, conjectures, and ideas about many problems. I hope that this is helpful for the field, but would like to know what you think. Although you are reading this in the book, you can still comment in the blog, both at the original item or in a relevant newer one. Comments never close.

8.6 Notes and Links

Original post:

<http://rjlipton.wordpress.com/2010/09/25/who-gets-the-credit-not-facebook-credits/>

Text on elliptic curves:

<http://www.amazon.com/Elliptic-Curves-Cryptography-Mathematics-Applications/dp/1584883650>

Poincaré conjecture:

http://en.wikipedia.org/wiki/Poincare_conjecture

Referenced post on Unique Games conjecture:

<http://rjlipton.wordpress.com/2010/05/05/unique-games-a-three-act-play/>

Picture credits:

http://en.wikipedia.org/wiki/Ricci_flow.png

Grigori Perelman, the solver of the Poincaré Conjecture in three dimensions, has [declined](#) to accept the first awarded Clay Prize. I may not understand him, but I think he is one of the most colorful mathematicians ever. Perhaps the award should have been offered to both him and Richard Hamilton—recall from Chap. 8 that Hamilton created the Ricci-flow approach used by Perelman. Perhaps together they would have accepted. Oh well.

We will speculate about what the [Clay Mathematics Institute](#) should add to their list to get it back to the magical number *seven*. I do not know whether they plan to do this, I doubt they will ask me for advice even if they do, but let's discuss what they should add anyway.

Seven is a magical number. One example is the work of George Miller, who is one of the world's great psychologists. When I was at Princeton I was honored to meet him. He has done many things, but he is famous for the "[rule of seven](#)." He discovered that most people can remember in their "working memory" only 7 ± 2 things. I probably am on the lower end of this bound these days, but his paper is one of the most cited in all of science. Perhaps his rule is one reason the Clay Institute should get back to seven problems.

I once worked with George on a large project, although we never wrote a joint paper together. The large project was called the *Massive Memory Project*. Our thesis was that large computer memory was as important as having many processors, sometimes even more important. It was under consideration for funding by NSF, but first we needed to "pass" a site visit. It went well until I was asked a certain hard question by a visitor X. I tried to answer X's question, but was not convincing. Then George Miller answered why memory was the answer to X's question:

The brain uses memory to do this, so I think we should use memory too.

This convinced X. We were soon funded by both NSF and DARPA.

9.1 Problems

What problem should Clay add to their list if they decide to get back to seven problems? Here are seven they may wish to consider.

- **Jacobian Conjecture:** We have discussed this in Chap. 7. I think it is one of the great open problems.
- **Hilbert’s Tenth Over Rationals:** Suppose $P(x_1, \dots, x_n)$ is an integer polynomial. The question is do there exist x_1, \dots, x_n *rational* numbers so that

$$P(x_1, \dots, x_n) = 0?$$

Of course if x_1, \dots, x_n have to be integers, then this is in general an undecidable problem. This follows from the beautiful work of Martin Davis, Yuri Matiyasevich, Hilary Putnam, and Julia Robinson on Hilbert’s Tenth Problem. However, when the x_i ’s can be rational numbers the problem is still open.

- **Graph Isomorphism:** The current Clay list does not have any graph theory problems. One that I might suggest is the famous—infamous?—problem of deciding in polynomial time whether or not two graphs are isomorphic. This problem has some beautiful partial results, but the question is unresolved.
- **Quantum Computing:** The question I would suggest is: Can quantum computers be simulated in polynomial time by classical ones? This seems to be one of the outstanding open questions in this area. An answer would have many consequences, and would shed light on both physics and computational complexity.
- **Goldbach Conjecture:** Recall this simple-to-state conjecture asks: can every even number greater than 2 be written as the sum of two primes? I think this is one of those “how can it be open?” questions, ones I have called “[mathematical embarrassments](#).”
- **An Explicit Quadratic Boolean Circuit Lower Bound:** The problem is to give an example of a Boolean function family $f_n(x_1, \dots, x_n)$ so that two things are true:
 - (1) The family can be computed in polynomial time by an explicit program.
 - (2) The general Boolean circuit complexity of f_n grows as $\Omega(n^2)$.
 I selected quadratic growth, but of course even $100n$ would be a major result.
- **Invariant Subspace Problem:** The invariant subspace problem is the question: “Does every bounded operator on a separable Hilbert space over the complex numbers have a non-trivial closed invariant sub-space?” B.S. Yadav wrote a survey [paper](#) giving some history, while Terence Tao [discussed](#) it on his blog.

9.2 Open Problems

Do you like any of my suggestions? What problem do you think should be added to the Clay list?

9.3 Notes and Links

Original post:

<http://rjlipton.wordpress.com/2010/07/21/a-new-million-dollar-prize/>

New York Times article:

<http://www.nytimes.com/2010/07/02/science/02math.html>

George Miller's work:

http://en.wikipedia.org/wiki/The_Magical_Number_Seven,_Plus_or_Minus_Two

Referenced post on Jacobian conjecture:

<http://rjlipton.wordpress.com/2010/07/17/an-amazing-paper/>

Referenced post on “embarrassing” open problems:

<http://rjlipton.wordpress.com/2009/12/26/mathematical-embarrassments/>

Comment by Paul Homer:

<http://rjlipton.wordpress.com/2010/07/01/can-amateurs-solve-ppn/#comments>

Invariant subspace problem, survey by B.S. Yadav:

<http://www.nieuwarchief.nl/serie5/pdf/naw5-2005-06-2-148.pdf>

Post by Terence Tao:

<http://terrytao.wordpress.com/2010/06/29/finitary-consequences-of-the-invariant-subspace-problem/>

Eric Allender is one of the world's experts on computational complexity. He is especially a master at the care and feeding of the complexity classes that are at the weaker end of the spectrum. These include classes like ACC^0 or TC^0 among many others. Just because these classes seem to be weak does not mean they give out their secrets easily. Proving almost anything new about them seems well beyond all the techniques that are available to us.

We will talk about a pretty result of Eric's, and how it connects to Chapter 8's discussion on partial results.

I have never had the honor of working on a project with Eric, even though he was at Rutgers and I was nearby at Princeton for many years. We did however have an interesting situation arise where Eric and his colleagues wiped out Anastasios Viglas and me.

It was back in 1999, and for some reason the question of what is the computational cost of deciding whether a given number is prime occurred to both groups, one at Rutgers and one at Princeton. Proving lower bounds on general Turing machines is too hard, so both groups asked: could it be proved that testing whether a number is prime or not at least cannot be too easy. In particular, could we prove that primality did not lie in some low-level complexity class.

The answer from Rutgers was "yes"; the answer from Princeton was "maybe." Eric, along with Michael Saks and Igor Shparlinski, [proved](#) that primality could not be in $\text{ACC}^0[p]$ for any prime p . Viglas and I proved a slightly weaker result, and what was worse, our result needed an unproved assumption about the structure of primes. It was one of those assumptions that must be right, but is probably hard to prove.

Unfortunately for us both papers were submitted to exactly the same conference, the 14th Annual IEEE Conference on Computational Complexity. Theirs got in, while ours did not. It was clearly the right decision, but I have wondered what would have happened if we had submitted alone. Oh well.

10.1 Allender on the Permanent

Now I will talk about another particularly pretty result, an even more notable theorem of Eric's from the late 1990s. The actual results proved in his [paper](#) are stronger, but for our purposes the following simple statement is sufficient:

Theorem 10.1 *The permanent is not in uniform TC^0 .*

Recall that TC^0 is a constant-depth circuit model that allows arbitrary threshold gates. In the theorem the circuits must be *uniform*: this means roughly that there is a simpler-than-logspace computation that tells which wires connect to which gates. There is some care needed in getting the model just right, so please look at Eric's paper for all the details.

The proof uses a number of facts from complexity theory, but shows that a contradiction would follow if the permanent were in the threshold class. In all it is a very pretty and clever argument. I think the result is well known, but I do not think Eric gets as much credit as he should for it. It is definitely one of the strong results about the complexity of permanent, and also one of the Mason–Dixon lines in the frontiers of lower bounds.

10.2 A Result to Dream of

I decided that after the previous discussion on partial results I would put out a partial result. The goal is to state a conjecture—actually a family of conjectures—so that if one is proved, then we can prove the following statement:

Conjecture 10.1 *The complexity class NC^1 is properly contained in the class $\#P$.*

The high-level issue is whether there is a loophole in the work by David Barrington and others that an efficient group-algebra simulation of Boolean gates **requires** a non-solvable group. The issue involves the so-called *commutator subgroup* $[H, H]$ of a subgroup H of a group G . This is the group generated by the *commutators* $[a, b] = aba^{-1}b^{-1}$ where $a, b \in H$. The commutators themselves need not form a group. The question is, what group I do they generate, and is it all of H ?

Well, if H is an Abelian subgroup then $I = \{e\}$, the trivial identity subgroup, and we've figuratively "hit bottom." If I is not $\{e\}$, and not all of H , then one can iterate and ask the same question about $[I, I]$, and so on. A group G is solvable provided you always hit bottom. Put another way, G is *non-solvable* provided it has a subgroup $H \neq \{e\}$ such that $[H, H] = H$. Barrington's proof employs such a subgroup H , indeed where H is simple.

Of course, most of us probably believe that solvable groups cannot replace the simple groups in Barrington's theorem, but that is an open problem. Suppose they can. Then the later work by Barrington and others on the solvable-group case would make NC^1 equal to (a subclass of) TC^0 . Allender's theorem would then make NC^1

properly contained in #P (which is equivalent for our purposes to the language class PP). The partial work has thus been to establish a connection between a family SOLVE of existential questions about solvable groups and results that would then fit together to prove an open separation theorem.

Moreover, if you don't believe $NC^1 = TC^0$, i.e. if you believe Barrington's characterization is tight, then you can read the connection the other way: results in complexity theory imply the impossibility of some pure statements in group theory. Maybe you can refute the particular statements we'll propose by direct algebra, but if they stay open then we gain a meaningful dialogue between complexity and algebra.

10.3 SOLVE

I will state one of the questions in the family SOLVE. They all imply as I stated that

$$NC^1 \subsetneq \#P.$$

The following is the question. Some group theory notation is needed. If x, y are in a group, then

$$yxy^{-1}$$

is the conjugate of x by y . Also $\langle a, b \rangle$ denotes the group generated by the elements a and b . Finally, the order of an element x in G is denoted by $o(x)$.

- Suppose that G is a solvable group. Are there four elements a, b, c, d in the group so that the following are true?
 - (1) Let K be the subgroup $\langle a, b \rangle$. Then we require that some conjugate of c and some conjugate of d lie in K .
 - (2) Let L be the subgroup $\langle c, d \rangle$. Then we require that some conjugate of c and some conjugate of d lie in L .
 - (3) And the following two numbers are relatively prime: $o(a)o(b)$ and $o(c)o(d)$.

The last clause may force that $H = \langle a, b, c, d \rangle$ must equal its commutator subgroup, which would violate G being a solvable group. That would shoot down this member of SOLVE, though we have others that might survive.

10.4 Open Problems

Prove SOLVE. Or disprove it. If you do that then I will put out some even weaker conditions. Let me know.

10.5 Notes and Links

Original post:

<http://rjlipton.wordpress.com/2010/09/28/can-group-theory-save-complexity-theory/>

Eric Allender, Michael Saks, and Igor Shparlinski, “A lower bound for primality,” *Journal of Computer and Systems Sciences* **62**(2), 2001, 356–366. Also available at:

<http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.53.55>

In comments, as we tell in Chap. 11, Colin Reid turned aside the idea. The similarly-titled Chap. 43 gives more technical background, while we moved this and the next chapter up to highlight the social process.

Enrico Bombieri is one of the world leaders in many areas of mathematics, including number theory, algebraic geometry, and analysis. He has won many awards, and is a Fields Medalist.

We will discuss the notion of intuition in mathematics. I am curious what it is, how to get it, and how to use it.

One story, perhaps an urban legend, is that a senior expert in real analysis was once sent a paper that “proved” a surprising theorem. The expert looked at the proof, and was immediately skeptical. The “theorem” seemed to be too surprising. His intuition, based on his great experience, was that the theorem could not be true. Yet even after hours of studying the proof he could not find any mistakes. But his intuition continued to bother him. He finally looked even more carefully, and found the problem. The author of the proof had used a lemma from a famous topology book. He had used the lemma *exactly* as it was stated in the famous textbook. But there was a typo in the book. Somehow the words “closed” and “open” had been exchanged in the statement of the lemma. This made the lemma false, caused a gap in the proof of the surprising theorem, and left the poor author with a buggy paper.

Proving theorems is not mechanical. Yes it does require formal manipulation. Yet proving theorems also requires the use of intuition, the ability to see what is reasonable or not, and the ability to put all components together. Blindly using a lemma from even the most famous textbook can be dangerous, as the story shows.

I once lost several months of hard work trying to use a published theorem to solve an open problem. I almost had a proof, but eventually—as in the story—I found a bug in the published result. The result was the sole result of a friend’s PhD thesis. Oh well. This is not an urban legend; I was there, but I will leave it to another time.

For now let’s turn to the discussion of intuition in mathematics.

In mathematics you don’t understand things. You just get used to them. John von Neumann

11.1 Number Theory

I think that it is not too hard to have a reasonable intuition in number theory, especially concerning the behavior of prime numbers. A pretty fair approximation is to try the distribution of primes as “random.” That is primes in the range $[N, N + \Delta]$ are roughly randomly selected odd numbers with the correct density: the number of primes in such an interval for Δ is about $\Delta / \ln N$. This of course follows from the prime number theorem.

A classic [example](#) of this is the conjectured behavior of twin primes. If primes are random, then one would expect that there are about

$$CN / \ln^2 N$$

twin primes in $[1, N]$ where C is a constant. Godfrey Hardy and John Littlewood made an even more detailed conjecture, which included the value of the constant C . They guessed that C is

$$\prod_{p \geq 3} \frac{p(p-2)}{(p-1)^2} \approx 0.6601 \dots$$

Of course there are local properties that need to be added to the model. The number of primes p so that $p + 2$ and $p + 4$ are also primes is not

$$CN / \ln^3 N,$$

but one. This follows since one of $p, p + 2, p + 4$ is divisible by 3: so p must be equal to 3.

I once had a proof that needed only a lemma about the structure of the primes to be complete. It was about a communication complexity lower bound that I was working on at the time with Bob Sedgewick. We could not prove the lemma, nor could we find any references to anything like it. So we made an appointment to see the famous number theorist, Enrico Bombieri. He is a member of the Institute for Advanced Study, and was there at the time. So Bob and I went over to ask him about our lemma.

Bombieri listened very politely, asked a question or two for clarification, and said, “Yes, that lemma is surely correct.” We were thrilled, since this would make our proof complete. I then asked him for a reference. He looked at me and said:

Of course the lemma about primes is true, but it is completely hopeless to prove it.

He had great intuition about primes, but proving certain results was then and still is today completely beyond anything anyone can do.

11.2 Geometry

My intuition in geometry, especially in high dimensions, is very poor. I have proved basic geometry theorems that hold in n dimensions, but I have terrible geometric intuition.

I am not sure why. I am not sure what makes geometry so hard for me, but it is very different from the study of prime numbers. There are plenty of geometric questions that I would not have any idea how to conjecture what is true. Perhaps it is just a defect in my abilities, but geometry seems to be orders of magnitude trickier than number theory.

John Moller of the University of Utah writes a blog titled “On Topology,” and gave some helpful insights into high-dimensional geometry in a March 3, 2009 post titled “Reasoning in Higher Dimensions: Hyperspheres” (referenced in the end notes).

11.3 Groups

My intuition about finite groups is even worse than my geometric intuition. No, that is not quite right. In a sense my intuition about groups is really very good. Over the years I have hit upon a rule in thinking about groups. I figured out that if I thought that X was a reasonable theorem that should hold for finite groups, then X was likely to be false.

Of course, this is a bit silly. It is like having a really poor sense of picking stocks—if you were always wrong, then there would be a great strategy. But, somehow I do believe there is something to what I am saying. My intuition is so bad that after a while I just started to guess the opposite of whatever I first thought.

Chapter 10 is a perfect example. At the time, I had a series of conjectures about solvable groups. The conjecture I listed took an expert, Colin Reid, a few minutes to disprove. He is a group theorist. I should have known better, as my intuition about groups is terrible, even though groups may play an important role in our understanding of computer science theory.

11.4 Reid’s Proof

Colin Reid posted his proof that a problem I had called *SOLVE* is impossible in the comments section of the post on which Chap. 10 is based. The parts of his comments using angle brackets were snipped as HTML-style tags, so the following is an expanded version fixing the glitches. Ken and I have also replaced his quotient-subgroup notation by homomorphism notation.

Let G be a non-trivial solvable group—some say soluble group. The *composition series* is defined by $G_0 = G$, and for $i \geq 1$, $G_i = [G_{i-1}, G_{i-1}]$ = the subgroup generated by the *commutators* $[u, v] = uvu^{-1}v^{-1}$ for $u, v \in G_i$. Solvability of G means that some G_i is the trivial subgroup $\{1\}$, whereupon the series terminates. The

important facts are that not only is every G_i normal in its predecessor, it is normal in the entire group G , and that the immediate quotients G_{i-1}/G_i are Abelian. This means that we can define a homomorphism π on all of G whose kernel is G_i , and for all $c, d \in G_{i-1}$, $\pi(c)$ and $\pi(d)$ commute. The reason for the latter is that

$$dc = cdk \quad \text{where } k \text{ is the commutator } [d^{-1}, c^{-1}],$$

so $\pi(d)\pi(c) = \pi(dc) = \pi(cdk) = \pi(c)\pi(d)\pi(k) = \pi(c)\pi(d)$, since k is in the kernel G_i .

Now recall that $SOLVE$ asserted the existence in G of elements a, b, c, d such that some conjugates xax^{-1} of a and yby^{-1} of b are in $\langle c, d \rangle$, where $x, y \in G$, and likewise some conjugates of c and d are in $\langle a, b \rangle$, with a third condition on the orders of these elements in G . The condition is that $o(a)o(b)$ be relatively prime to $o(c)o(d)$, and this is what finally prevents the existence of a, b, c, d .

For contradiction, suppose a solvable group G with such elements exists. Then in the composition series, there is some i such that G_{i-1} contains all of a, b, c, d , but G_i does not. By symmetry, without loss of generality, we can suppose G_i does not have a . Take $a' = xax^{-1}$ as above, so $a' \in \langle c, d \rangle$. It does not matter whether c or d belongs to G_i ; that both belong to G_{i-1} is enough. Since a' is a conjugate, $o(a') = o(a)$. Now take the homomorphism π with G_i as kernel, and observe:

- (1) $\pi(a') \in \langle \pi(c), \pi(d) \rangle$.
- (2) $o(\pi(c))$ divides $o(c)$, $o(\pi(d))$ divides $o(d)$, and $o(\pi(a'))$ divides $o(a') = o(a)$.
- (3) Since c and d are in G_{i-1} , $\pi(c)$ and $\pi(d)$ commute.
- (4) Hence $\pi(c)$ and $\pi(d)$ can generate at most $m = o(\pi(c))o(\pi(d))$ different elements.
- (5) Put more strongly, the subgroup $\langle \pi(c), \pi(d) \rangle$ they generate is also a subgroup of the Abelian group generated by $\pi(c)$ and $\pi(d)$, which has exactly m elements. Hence by Lagrange's theorem the order of $\langle \pi(c), \pi(d) \rangle$ divides m .
- (6) Since $\pi(a')$ is in $\langle \pi(c), \pi(d) \rangle$, it follows that $o(\pi(a'))$ divides m , which divides $o(c)o(d)$.
- (7) However, since $o(\pi(a'))$ divides $o(a)$ which is relatively prime to $o(c)o(d)$, the only way this can happen is $o(\pi(a')) = 1$.
- (8) That means $\pi(a') = 1$, so $a' \in G_i$ since G_i is the kernel of π .
- (9) But G_i is normal in G , not just in G_{i-1} , so $xa'x^{-1}$ is in G_i . This puts a into G_i , which yields a contradiction.

To someone with good algebraic intuition this comes trippingly off the tongue, which is why it is a service to communicate in public. So we thank Colin Reid—and we will see if the insight gained works against our more-complicated stratagems of this kind. At the time he was a graduate student at Queen Mary College, University of London.

It may be that deterministically simulating each level of a Boolean circuit simply must bump you one step along a composition series, which in a solvable group G is a finite, non-renewable resource.

However, we also have other ideas. Perhaps we can get mileage out of choosing different solvable groups G for different input sizes n . This relates complexity questions to ones about the possible lengths of composition series in groups of certain

sizes—although what we know about such sizes is not so promising. But perhaps a randomized simulation can possibly avoid these limitations.

11.5 Complexity Theory

I think we have good intuition here, but I have seen many surprises in my career:

- Linear programming is in polynomial time.
- Nondeterministic space is closed under complement.
- Polynomial-size bounded-width branching programs are powerful.
- Permanent has random self-reduction.
- Quantum polynomial time can factor integers.
- Random walks for undirected graphs can be de-randomized.
- Proofs can be checked in constant time.
- Zero-knowledge protocols exist for natural problems.
- ...

I have reasonable intuition, yet all of these were surprising to me—even some that I worked on and contributed to their development.

11.6 Open Problems

Is intuition simply built up by learning more and more about an area? Or is intuition something that is separate from just being an expert in an area? Can you be quite strong in an area and still have weak intuition, or is that impossible?

11.7 Notes and Links

Original post:

<http://rjlipton.wordpress.com/2010/10/01/mathematical-intuition-what-is-it/>

Twin primes:

http://en.wikipedia.org/wiki/Twin_prime

Blog item on higher-dimensional geometry:

<http://ontopo.wordpress.com/2009/03/03/reasoning-in-higher-dimensions-hyperspheres/>

Colin Reid's homepage:

<http://qmul.academia.edu/ColinReid>

Fred Hennie was a professor of Electrical Engineering at MIT for many years, and did seminal early work in the theory of computation. He is famous for a number of beautiful results on lower bounds, as well as work on algorithms and even discrete mathematics.

We will talk about attempts over the years to get lower bounds on SAT. Of course if $P \neq NP$ is ever proved then the following weak results will all become uninteresting. However, until that happens these are the best results we have for proving lower bounds on Turing-type machines.

The lower bounds on SAT use connections between old ideas and new ideas—between problems once studied for their own sake and problems that now are considered central.

12.1 The Models

The models we are interested in are all single-worktape Turing Machines (TMs). Well, that is a lie: the models we are *really* interested in are multiple-tape Turing Machines, but no one seems to have a clue how to get even an $n \log \log n$ -time lower bound on such powerful models, for SAT or any “natural” NP-complete problem. So we do what we can. Even one-tape machines are non-trivial.

Here are the three models of Turing Machines that we will discuss:

Basic model This is a TM with one tape. The tape starts with the input, and is a read–write tape.

One-way input This differs from the basic model in only one aspect: the input is on a separate one-way tape. Such a tape may as well be read-only.

Two-way input This also differs from the basic model in only one aspect: the input is on a separate two-way read-only tape.

Sometimes the basic model was called “off-line,” and other times the last model was called this too. This has confused me—so be careful when reading any papers in this area. For some reason the names shifted years ago, and this shift can cause lots of confusion. I wonder if the confusion might have been driven by the interest years ago in *real-time* computations.

A computation that computes a function in real-time must output a bit every step of the computation. This type of computation was interesting because it is possible to actually prove lower bounds on them. Whereas it is still impossible to prove non-linear lower bounds for multi-tape TMs for natural problems that are in polynomial time. Fifty years. No progress. None. However, Mike Paterson, Michael Fischer, and Albert Meyer were able in 1974 to [prove](#) strong lower bounds on an even more inclusive model of computations for integer multiplication. Their proof used a method called the [overlap method](#) created by Steve Cook and Stål Aanderaa in 1968.

12.2 Hennie's Result

I believe that Hennie proved the first quadratic lower bounds on the basic model. Note, his result shows that the bound is also tight.

Theorem 12.1 *Recognizing the language PAL of palindromes requires $\Omega(n^2)$ time in the basic model.*

Recall a string w is a palindrome if it satisfies $w = w^R$ where w^R is the reverse of the string: w^R is $w_n w_{n-1} \dots w_1$. In English the rules are relaxed a bit to allow spaces and punctuation to be ignored. Thus, the following is a “palindrome.”

“No, it never propagates if I set a gap or prevention.”

More than the result, what was interesting was the proof method. Hennie used what is called a **crossing sequence** argument. Roughly, he showed that in testing whether a string is a palindrome the tape head would have to move across the middle part of the tape order- n times. This yields the lower bound of order n^2 .

Ken adds that the proof can be viewed as like a game of “Battleship.” In that game a screen between two players hides their respective layouts of ships on a grid. In our case the screen divides a string x of the form $u0^m v$ where $|u| = |v| = m$ between the players, and can be positioned anywhere in the 0^m part. Wherever it is, the information carried by the basic Turing machine M over the screen allows the second player to infer u by trying all $v \in \{0, 1\}^m$, since the unique “yes” answer from M will come when $v = u^R$. Let m_0 be the number of bits required to specify u . It follows that m_0 bits must cross over each of $m + 1$ possible screen positions, so the total information ferried by M on input $u0^m u^R$ must be at least $m_0(m + 1)$. Since there are incompressible u with $m_0 \geq m$, and $m = \Theta(n)$, we have the order- n^2 time lower bound.

12.3 Since Hennie

There have been many results since Hennie's pioneering paper. Many of these papers use crossing sequence arguments, sometimes variations of the original method. I will just summarize some of the main results—I apologize if I have left out your favorite results.

- (1) Wolfgang Maass [proved](#) a similar lower bound on the basic model with a one-way input tape.
- (2) Dieter van Melkebeek and Ran Raz [proved](#) a pretty result on a model that is even stronger than the basic model plus two-way input. They allowed random access to the input and still proved a lower bound of order n^a where $a < \sqrt{1.5}$.
- (3) Ryan Williams proved related and even stronger results in a paper mentioned below and referenced in the end notes.

The results by van Melkebeek and Raz extend to Turing machines with one d -dimensional worktape, provided $a^2 < (d + 2)/(d + 1)$. But as soon as you add a second worktape, even a 1-dimensional one, all bets are off. Why is that? An explanation is given by another great result of Hennie, with Richard Stearns:

Theorem 12.2 *A Turing machine with any number of worktapes can be simulated by a two-worktape machine with only a log-factor overhead in time.*

Thus an n^a lower bound like theirs with $a > 1$ for a two-worktape machine would essentially imply an n^a lower bound for TMs with any number of tapes.

12.4 Toward Even Stronger Bounds

The stronger results by Ryan Williams are based on the fact that SAT is NP-complete, which is something that Hennie did not have available to him back in the 1960s. Also at the time it was interesting that relatively simple languages could be used to get lower bounds. Today we are more interested in SAT than in any artificial problem.

Ryan in his paper also gives evidence that getting much stronger lower bounds on SAT via these methods will require new ideas. Perhaps they are already out there, perhaps not.

One question I believe is open: what happens here for probabilistic one-tape machines? I am especially interested in these since they are related to getting lower bounds on practical SAT algorithms such as GSAT. Recall that algorithms like GSAT operate as follows: They select a random assignment for the variables, and then look for an unsatisfied clause. If there are none, then they have found a satisfying assignment. If there is an unsatisfied clause, then they randomly pick such a clause and flip a variable.

The key point is this: any algorithm of this type can be implemented efficiently on the basic model plus a random access input. Thus, suppose van Melkebeek and Raz's theorem could be improved to work for probabilistic TMs. Then it would follow that any algorithm like GSAT would take at least \sqrt{n} flips. This is a pretty small number of steps, but at least it would be a bound.

12.5 Open Problems

One problem is to check the details that GSAT-style algorithms can be implemented as claimed. The much harder problem is to show that the lower bounds can extend to randomized TMs.

12.6 Notes and Links

Original post:

<http://rjlipton.wordpress.com/2010/09/02/old-ideas-and-lower-bounds-on-sat/>

Paterson–Fischer–Meyer paper:

<http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.17.4249>

Cook–Aanderaa paper:

<http://www.ams.org/journals/tran/1969-142-00/S0002-9947-1969-0249212-8/S0002-9947-1969-0249212-8.pdf>

Maass paper:

<http://portal.acm.org/citation.cfm?id=808706>

van Melkebeek–Raz paper:

<http://portal.acm.org/citation.cfm?id=1132646&dl=ACM&coll=portal>

Williams paper:

<http://www.cs.cmu.edu/~ryanw/automated-lbs.pdf>

GSAT:

<http://en.wikipedia.org/wiki/WalkSAT>

In comments, Emmanuele Viola noted a paper of his with further work (<http://www.ccs.neu.edu/home/viola/papers/BPvsE.pdf>). Paul Beame contributed further analysis of the problems.

Volker Strassen has won many prizes for his terrific work in computational complexity—including the Cantor Medal, the Paris Kanellakis Award, and most recently the Knuth Prize. He is perhaps most famous for his brilliant work on matrix multiplication, but he has done so many important things—others may have their own favorites.

We should talk about some amazing results. Two are classic results due to Strassen, and the third is a recent result related to Boolean matrix products.

The topic of “Amazing Results” may merit a longer treatment. I need to think, first, how I would define them. For now I will just give three examples.

13.1 Fast Matrix Product

Strassen’s 1969 paper “Gaussian Elimination Is Not Optimal” showed a way to multiply two $n \times n$ matrices in time order $n^{2.807}$, instead of the classic cubic-time method. This is certainly an amazing result—it came as a shock to everyone that the classic method was not optimal. Strassen’s fast matrix algorithm launched a thousand results; today many of the best algorithms for X are obtained by reducing X to matrix products. As a consequence, it is quite common to see algorithms that run in time $O(n^{\omega+2})$, for instance, where ω is the exponent of the fastest known matrix multiplication algorithm.

The best matrix product algorithm for long was due to Don Coppersmith and Shmuel Winograd, and runs in time $O(n^{2.376})$. I must mention the brilliant [work](#) of Henry Cohn, Robert Kleinberg, Balázs Szegedy, and Christopher Umans who reduced the search for faster matrix algorithms to certain group theory questions.

There is a story—I believe it is true—of how Strassen found his terrific algorithm. He was trying to prove that the naive cubic algorithm is optimal. A natural idea was to try and prove first the case of two-by-two matrices—one-by-one matrices are trivial. One day he suddenly realized, if there were a way to multiply two-by-two matrices with 7 multiplications rather than 8, there would be a sub-cubic recursive method for n -by- n . He immediately sat down and quickly discovered his now famous formulas. At least this is the story.

13.2 All for the Price of One

Strassen also has another amazing result, in my opinion, on arithmetic circuits—this is joint work with Walter Baur. They used this [result](#) to get some tight lower bounds on arithmetic computations. In 1983 they proved their result about the arithmetic straight-line complexity of polynomials: let $L(f)$ denote the minimum number of basic arithmetic operations needed to compute the multivariate polynomial f , and let

$$L(f, g, h \dots)$$

denote the minimum number of operations needed to compute the polynomials f , g , h , and so on, *all at the same time*. As usual $\partial f/\partial x$ is the partial derivative of f with respect to the variable x . The amazing result they proved is:

Theorem 13.1 *For any polynomial $f(x_1, \dots, x_n)$,*

$$L(f, \partial f/\partial x_1, \dots, \partial f/\partial x_n) \leq 3L(f).$$

Thus, the cost of computing f is almost the same as the cost of computing f and all n of its first-order partial derivatives. Getting n functions for the “price” of one seems surprising to me.

This result has been used to get lower bounds—this was why Baur and Strassen proved their result. The general method is this:

- Argue the cost of computing f_1, \dots, f_n is $\Omega(S)$.
- Then, reduce computing these n polynomials to computing g and all its partial derivatives, for some g .
- This implies, by the Baur–Strassen theorem, that the cost of $L(g)$ must be at least $\Omega(S)$.

The critical point is that proving a lower bound on the cost of **one** polynomial may be hard, but proving a lower bound on the cost of n polynomials may be possible. This is the magic of their method.

13.3 Finding Triangles

Virginia Vassilevska Williams and Ryan Williams have proved a number of surprising results about various matrix products and triangle detection.

One of the long-standing open questions is, what is the complexity of determining whether a graph has a triangle? That is, does the graph have three distinct vertices a, b, c so that $a - b$, $b - c$, and $c - a$ are all edges?

One way to detect this is to use matrix product—that is to invoke Strassen, or rather the newer algorithms. Another approach is combinatorial. It is easy to see that there is an algorithm that runs in time order

$$\sum_k \deg(v_k)^2.$$

Of course, this can be as large as cubic in the number of vertices.

What Williams and Williams consider is the following question: is there a relationship between detecting triangles and computing Boolean matrix products? They also consider the same type of question between Boolean matrix verification and Boolean matrix products.

The surprise is that the ability to *detect* triangles or the ability to *verify* Boolean products, can also be used to compute Boolean matrix products. What is so surprising about this is these both return a single bit, while matrix product computes many bits. How can a single bit help compute many bits? The simple answer is that it cannot—not without some additional insight. Their insight is that the algorithms can be used over and over, and in this manner be used to speed up the naive Boolean matrix product.

13.4 One Bit, Two Bits, Three Bits, n Bits

Let me give a bit more detail on what they prove, not on how they prove it. See their [paper](#) for the full details. Since they prove many theorems, I will state just one.

Theorem 13.2 *Suppose Boolean matrix product verification can be done in $O(n^{3-\delta})$ time for some $\delta > 0$. Then graph triangle detection on n -node graphs can be done in $O(n^{3-\delta})$ time.*

Note that matrix product verification is simple with randomness, but Boolean matrix product seems to be much harder to check. Recall that to check that $A \cdot B = C$ for n by n matrices, pick a random vector v . Then, check

$$(A \cdot B)v = A \cdot (Bv) = Cv.$$

This can be done in $O(n^2)$ time.

13.5 Open Problems

The main open problems are of course to finally resolve the exponent on matrix product—is it possible to multiply in $n^{2+\varepsilon}$ for any $\varepsilon > 0$? Also the Williams' results show there are very interesting connections between checking and computing. I think more can be proved in this direction.

13.6 Notes and Links

Original post:

<http://rjlipton.wordpress.com/2010/03/27/fast-matrix-products-and-other-amazing-results/>

Baur–Strassen paper:

<http://portal.acm.org/citation.cfm?id=382836>

Paper by Cohn et al.:

<http://arxiv.org/abs/math.GR/0511460>

Triangle-detection paper:

<http://www.cs.cmu.edu/~ryanw/tria-mmult.pdf>

Later in 2010 and in 2011, Andrew Stothers and Virginia Vassilevska Williams gave successive minor improvements to the Strassen constant. We posted about this work at:

<http://rjlipton.wordpress.com/2011/11/29/a-breakthrough-on-matrix-product/>

and

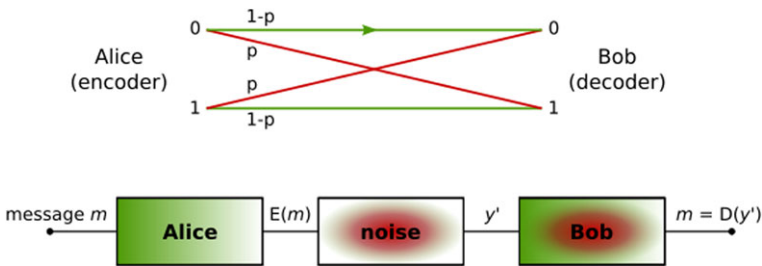
<http://rjlipton.wordpress.com/2011/12/03/the-meaning-of-omega/>

Adam Smith is a complexity theorist who works in the area of coding theory, and especially the interface between information theory and cryptography.

We describe a result of his with Venkatesan Guruswami. They both are masters of using cryptography and information theory, and they use these tools to prove a pretty theorem in information theory. This result appeared at the FOCS 2010 conference. Guruswami gave a detailed talk on it at the 2010 Eastern Great Lakes Theory of Computation Workshop, which was held on the University at Buffalo campus. The workshop was sponsored by NSF and the Transborder Regional University Network.

The central idea is one that I find extremely appealing. In their paper they study, among other things, the consequences of limiting the power of the communication channel. This is not a new idea, information theory has long studied binary symmetric channels. These are channels that have a probability p of “flipping a bit,” and each bit is flipped with this probability and **independently**. The reason these channels are so popular is two-fold: (i) they are the simplest non-trivial channels and usually are the easiest to understand, and (ii) they sometimes can be used to solve even more complex situations.

Our figure taken from Wikipedia shows Alice sending bits to Bob over such a channel.



I discussed the idea of limiting the power of the communication channel in a [post](#) from 2009.

An aside: are you all getting tired of Alice and Bob, they seem to be everywhere? I recently was told to stop using them to explain things. What do you think? Here is a great [comment](#) about them from John Gordon:

So you see Alice has a whole bunch of problems to face. Oh yes, and there is one more thing I forgot to say—Alice doesn't trust Bob. We don't know why she doesn't trust him, but at some time in the past there has been an incident.

Now most people in Alice's position would give up. Not Alice. She has courage which can only be described as awesome. Against all odds, over a noisy telephone line, tapped by the tax authorities and the secret police, Alice will happily attempt, with someone she doesn't trust, whom she cannot hear clearly, and who is probably someone else, to fiddle her tax returns and to organize a coup d'etat, while at the same time minimizing the cost of the phone call.

A coding theorist is someone who doesn't think Alice is crazy.

14.1 Computationally Limited Channels

A major idea in their paper is to restrict the communication channel between Alice and Bob—yes I will stay with them—to be a logspace-limited computation. Actually they study a number of other results, but this is the one that I will discuss. The restriction on the *space* used by the channel to decide which bits to change is pretty reasonable in my opinion.

But I am biased. Almost twenty years ago I worked on a related problem and got weaker results. I did however introduce—I believe—the idea of limiting the computational power of the channel. My results were for the case where the channel was polynomial-time limited, which is of course more powerful than a logspace-limited one. At least we believe that is true: recall we still do not know whether logspace equals polynomial-time.

Shortly after my paper was out, Zvi Galil, Moti Yung, and Xiangdong Yu wrote a paper on limited channels, changing my assumption to logspace. I made a terrible mistake. They asked me to join them and we eventually wrote a joint paper which we never published. It was titled: “Computational Error-Correcting Codes Achieve Shannon's Bound Explicitly.”

My terrible mistake was joining them. Their paper without me had a line in the introduction about the “great idea of Lipton.” But once I was added to the paper that line was removed, as it should have been. Rule:

Never join a paper that gives you a great reference.

14.2 The New Results

Adam and Venkatesan consider more generally space-limited channels. They showed that as long as the channel is restricted to logspace they have a coding method that is efficient and optimal up to the channel rate of $1 - H(p)$ and recovers a short list that contains the correct message with high probability.

I will not attempt to give any details of how they prove their theorem, take a look at their [paper](#). I will make a few high-level comments about their wonderful paper. It makes use of Noam Nisan's famous pseudorandom [generator](#). This is one of the great results in complexity theory: Noam shows that there is a pseudorandom number generator that “fools” logspace computations, and does this unconditionally. Noam's proof uses no unproved assumptions, no need to assume that one-way functions exist.

Their paper also extends the previous work in several ways. The major one, in my opinion, is they get rid of the assumption of a shared secret that was used in the previous work. In my paper and in the paper of Galil, Yu, and Yung the assumption was that Alice and Bob had a shared secret. Clearly removing this is extremely important. They call these “setup” assumptions. The proof that they can do this is quite a subtle one. The “obvious” idea is to send the shared secret over the channel, but this must be done with great care. Recall they are making no assumptions about the existence of public-key systems. Take a look at how they make all these work—it is quite impressive.

14.3 Open Problems

Suppose that you have a result that uses an error-correcting code somewhere. In order to make your result work against worst-case adversaries, you would likely need to use a classic code that can handle an all-powerful adversary. What would happen if you replaced the worst-case code by the code of Adam and Venkatesan? Of course your result would now likely *fail*. There might be a situation that would invalidate your result. The open question is: can your result still be proved to be true if the adversary has limited computational power? I would love to see such a result.

Another question is a version of this for the SAT problem, my favorite problem. Suppose that a logspace machine M must create instances of SAT problems. If we allow the machine M to be random it is unclear why the generated instances would be any easier than “general” ones. However, suppose that we require that the machine M must know the answer to the instances it generates:

- (1) If M generates a satisfiable formula ϕ , then M must also generate a satisfying assignment.
- (2) If M generates an unsatisfiable formula ϕ , then M must also generate a short proof of this.

What is the complexity of SAT for formulas generated in this way?

14.4 Notes and Links

Original post:

<http://rjlipton.wordpress.com/2010/10/14/strong-codes-for-weak-channels/>

Guruswami–Smith paper:

<http://arxiv.org/abs/1004.4017>

FOCS 2010 conference page:

<http://www.egr.unlv.edu/~larmore/FOCS/focs2010/>

Talk by Guruswami:

<http://www.cse.buffalo.edu/events/theory-III/slides/venkat.pdf>

Eastern Great Lakes (EaGL) workshop:

<http://www.cse.buffalo.edu/events/theory-III/>

Transborder Regional University Network:

<http://wings.buffalo.edu/intled/trun/>

Binary symmetric channel:

http://en.wikipedia.org/wiki/Binary_symmetric_channel

Referenced post on channels:

<http://rjlipton.wordpress.com/2009/05/06/worst-case-complexity/>

John Gordon on Alice and Bob:

<http://downlode.org/Etext/alicebob.htm>

Nisan generator:

http://en.wikipedia.org/wiki/Pseudorandom_generator#Pseudorandom_generators_for_logarithmic_space

Picture credits:

Binary symmetric channel:

http://en.wikipedia.org/wiki/Binary_symmetric_channel

[http://upload.wikimedia.org/wikipedia/commons/thumb/8/8e/Binary_symmetric_channel_\(en\).svg/600px-Binary_symmetric_channel_\(en\).svg.png](http://upload.wikimedia.org/wikipedia/commons/thumb/8/8e/Binary_symmetric_channel_(en).svg/600px-Binary_symmetric_channel_(en).svg.png)

Georg Cantor discovered his famous diagonal proof method, which he used to give his *second* proof that the real numbers are uncountable. It is a curious fact that Cantor's first proof of this theorem did not use diagonalization. Instead it used concrete properties of the real number line, including the idea of nesting intervals so as to avoid overlapping a given countable sequence.

This brings us to discuss the famous second proof: the diagonal method of Cantor.

In my teaching experience, students find it hard to believe Cantor's diagonal method. Perhaps it is my fault, but I have talked to others who teach the same result, and I hear the same comments. The diagonal method is elegant, simple, and deep. Students usually follow the method line by line, but I am sure that many really fail to get it. Perhaps that it is a proof by contradiction makes it hard to follow? But, they seem to get other proofs by contradiction. Or is the key problem that it is about infinities?

Here is an interesting quote by the logician Wilfrid Hodges:

I dedicate this essay to the two-dozen-odd people whose refutations of Cantor's diagonal argument have come to me either as referee or as editor in the last twenty years or so. Sadly these submissions were all quite unpublishable; I sent them back with what I hope were helpful comments. A few years ago it occurred to me to wonder why so many people devote so much energy to refuting this harmless little argument—what had it done to make them angry with it? So I started to keep notes of these papers, in the hope that some pattern would emerge. These [pages](#) report the results.

You might enjoy his essay—it is a careful treatment of some of the issues that people have in following Cantor's famous argument.

Let's turn to prove the famous result.

15.1 Proofs

I will give two different proofs that the reals are not countable. Actually, I will prove the statement that no countable list of infinite sequences of 0–1's can include all such sequences.

This is enough because of two observations. First, it is enough to show that the interval $[0, 1]$ is uncountable. Second, the reals in the interval have the same cardinality as the set of all of the infinite 0–1 sequences.

The first proof is essentially the famous diagonal proof, with a slight—very slight—twist. The second is a proof based on probability theory.

15.2 A Variant of the Classic Proof

Consider the following triangular array of bits that has an infinite number of rows:

$$\begin{array}{l} s^1(1) \\ s^2(1) \quad s^2(2) \\ s^3(1) \quad s^3(2) \quad s^3(3) \\ \vdots \end{array}$$

The i th row is

$$s^i(1) \ s^i(2) \ \dots \ s^i(i)$$

where each $s^i(j)$ is a 0 or a 1.

Our plan is to construct an infinite sequence $t(n)$ that is different from each row. Let's construct t . We need that t is different from $s^1(1)$ so there is no choice: set $t(1)$ equal to $\neg s^1(1)$. Note, there was no choice here: often the lack of choice is a good thing. In a proof if there is no choice, then you should be guided to the right choice. Henry Kissinger, as given by the website "Brainy Quote," once said:

The absence of alternatives clears the mind marvelously.

Next we must make t different from $s^2(1)s^2(2)$. We could be lucky and

$$t(1) \neq s^2(1).$$

But, we must be prepared for the worst case. So we set $t(2)$ equal to $\neg s^2(2)$. This forms a pattern: the simple rule is to set $t(i)$ to $\neg s^i(i)$.

Look at the triangular array again and we see that t is just equal to the negation of the **diagonal** elements:

$$\begin{array}{l} s^1(\mathbf{1}) \\ s^2(1) \quad s^2(2) \\ s^3(1) \quad s^3(2) \quad s^3(3) \\ \vdots \end{array}$$

This is why we call it the diagonal method. What does this have to do with the reals being uncountable? Suppose that now we have an array where each row is infinite too:

$$\begin{array}{lll} s^1(1) & s^1(2) & s^1(3) \dots \\ s^2(1) & s^2(2) & s^2(3) \dots \\ s^3(1) & s^3(2) & s^3(3) \dots \\ \vdots & & \end{array}$$

We want to construct a t so that it is different from each row. Just forget about the extra part of the array: use only one from the first row, two from the second row, and so on. The above just becomes our old friend:

$$\begin{array}{lll} s^1(1) & & \\ s^2(1) & s^2(2) & \\ s^3(1) & s^3(2) & s^3(3) \\ \vdots & & \end{array}$$

But, we just constructed a t that is different from each row. I claim that t works with the array that has rows of infinite length. The key observation is trivial: if t differs from the start of a row, it certainly is different from the whole row. That is it.

15.3 A Probability-Based Proof

In this proof we use the probabilistic method. We just pick a random 0–1 sequence t , and claim with positive probability that it is not equal to any sequence in the list s^1, s^2, \dots . Thus, such a t must *exist*.

Let $E_{n,i}$ be the following event:

$$t(1), \dots, t(n) = s^i(1), \dots, s^i(n).$$

Clearly,

$$\text{Prob}[E_{n,i}] = 2^{-n}.$$

The key is the event E defined as

$$E_{2,1} \vee E_{3,2} \vee E_{4,3} \vee \dots$$

The probability of E is at most

$$\text{Prob}[E_{2,1}] + \text{Prob}[E_{3,2}] + \dots$$

which is equal to

$$1/2 = 1/4 + 1/8 + 1/16 + \dots$$

Thus, the probability of the complement event \overline{E} is $1/2$.

But, \overline{E} is true provided t is not equal to any s^i . For suppose that t was equal to s^i , then it must be the case that

$$t(1), \dots, t(n) = s^i(1), \dots, s^i(n)$$

for any n . In particular, the event $E_{i+1,i}$ must be true, which is a contradiction since

$$\overline{E} = \overline{E}_{2,1} \wedge \dots \wedge \overline{E}_{i+1,i} \wedge \dots$$

Even though the methods look different, if you look closely you would notice that they both have Cantor's diagonal method at their heart.

15.4 Open Problems

There are many other papers on alternative approaches to proving the reals are uncountable. One is by Matthew Baker, titled "Uncountable Sets and an Infinite Real Number Game" (referenced in the end notes). Baker gives a great explanation of his method, which is close to the first proof that Cantor found.

Did you always believe the classic proof that the reals are uncountable? Or did this discussion help? I hope it increased your understanding, rather than decreased it.

15.5 Notes and Links

Original post:

<http://rjlipton.wordpress.com/2010/01/20/are-the-reals-really-uncountable/>

Previous post on Cantor's "first proof" of his theorem:

<http://rjlipton.wordpress.com/2009/04/18/cantors-non-diagonal-proof>

Wilfrid Hodges' observations:

<http://www.math.ucla.edu/~asl/bsl/0401/0401-001.ps>

Kissinger quote:

http://www.brainyquote.com/quotes/authors/h/henry_a_kissinger_2.html

Matthew Baker, "Uncountable sets and an infinite real number game," *Mathematics Magazine* **80**(5), December 2007, 377–380. Also available at

<http://people.math.gatech.edu/~mbaker/pdf/realgame.pdf>

This post had an especially lively comment discussion. Fields Medalist Terence Tao remarked:

Cantor's theorem is part of a general family of results that show that the class of all "potential" solutions to some problem is far larger than the class of "explicitly describable" or "actually solvable" solutions. . .

and gave seven further examples.

Raymond Smullyan is a great writer of popular books on logic—especially books on various forms of the [liar paradox](#). He is also a first-rate logician whose [book](#) on set theory with Melvin Fitting was just republished by Dover. This book gives a very readable and somewhat unusual approach to set theory, especially the independence results of Kurt Gödel and Paul Cohen.

We will use an example from their book to explain Georg Cantor’s famous diagonal argument. I have an idea why some people have difficulty with Cantor’s beautiful proof. If you know it well, you may wish to read on anyway; if you do not know it or do not believe it, then please read on and see if I have something to say. I have discussed the argument before, but still feel there is something to add to the discussion.

I have never had the pleasure of meeting Smullyan, but have heard a number of stories about him. One concerns the etiquette at Yale University, where I had my first job in the Computer Science department. One day I wrote a letter and gave it to a secretary to type up on our CS letterhead. My letter started

Dear Dr. X:
I am sending you my referee report on . . .

The letter came back slightly changed

Dear Mr. X:
I am sending you my referee report on . . .

I immediately asked why the change from “Dr.” to “Mr.”? The answer was: “This is Yale, we assume that all professionals have doctorates and call them ‘Mr.’ or ‘Ms.’ and so we never use any formal titles.” Okay I was new, I was young, so the letter went out as changed.

Later on I heard a story about Smullyan, who had given a talk at Yale, in the 1950s, before he had his PhD. After his talk there was tea in his honor, and his host introduced Smullyan around to all.

Mr. Blue, Mr. Green, Ms. Red, Mr. Yellow, and so on, I would like to introduce you to *Smullyan*.

Note, the introduction was to “Smullyan” not “Mr. Smullyan.” I do not know firsthand if the story is true or not, but I like it very much. Oh well.

Let’s turn to the classic argument of Cantor that proves the reals are uncountable. Somehow his argument is subtle, and many do not “get it.” There are some who are puzzled when they first see the proof, there are others who never understand it, and there are some who do not believe it at all. I would like to use Alice and Bob to help explain what the roadblock may be to understanding Cantor.

16.1 Alice and Bob Play Some Games

Consider the following simple game that Alice challenges Bob to play. Alice picks a natural number X , which she hides from Bob. Bob then must try and guess the number. This is a pretty tough game for Bob, but Alice is quite fair. She allows Bob to make an infinite number of guesses. It should be clear that Bob has a simple winning strategy. He just announces his strategy; he will say

$$1, 2, 3, 4, \dots,$$

Eventually he will say X and he will win the game. Alice agrees.

She then challenges Bob to another game. This time Alice picks a secret X that is an integer. Again Bob has an infinite number of guesses. Bob thinks for a moment and says his strategy now is to say

$$0, -1, 1, -2, 2, \dots,$$

They both agree that he wins again.

So far the games have been pretty easy, and Bob is feeling pretty good. Then Alice says she will pick a *pair* (X, Y) of natural numbers. She smiles and asks Bob what is his strategy now? This time Bob has to think a bit. He notices he cannot use a strategy like

$$(1, 1), (1, 2), \dots, (1, k), \dots$$

and then switch to the pairs

$$(2, 1), (2, 2), \dots, (2, k), \dots,$$

The problem is that if Alice picked $(2, 10)$ he will never get to it; all of his infinite number of guesses will be of pairs of the form $(1, Y)$. Then Bob sees the “trick”: guess first $(1, 1)$ and then $(1, 2)$ and $(2, 1)$. In general guess all the pairs that sum to k before all the pairs that sum to $k + 1$. If Alice selects (X, Y) , then Bob will guess it when $k = X + Y$. He wins again.

Alice has many games like this, but here is the last one she decides to ask Bob. She said now my secret is a *finite* set of natural numbers X . She again challenges Bob to guess her secret. Bob gets this one pretty fast—he is catching on to the

games. He realizes he can just generalize the pair idea. He breaks his guesses up into groups. In the k -th group he will guess all finite sets of natural numbers that sum to at most k .

Alice gives up, for now, since Bob has handled all her challenges so well.

16.2 Let's Look at Bob's Strategies

In set theory terminology Alice's games prove:

- (1) The natural numbers are countable.
- (2) The integers are countable.
- (3) The pairs of natural numbers are countable.
- (4) The finite sets of natural numbers are countable.

The (3) game can be viewed as proving the rational numbers are countable.

Note there was something very special about Bob's strategies. He did not need to keep them secret. Even if Alice knew his strategy for the second game, for example, she still would lose. No matter what number she selected, positive or negative, he would eventually guess the number. This is true of all of Bob's strategies. They win whether or not Alice knows them.

This is a very special situation in playing games, even playing infinite games. Since Bob's strategies are so powerful, I believe there should be no argument with the conclusions. For example, the pairs of natural numbers are countable. Do you agree?

16.3 Is Diagonalization Cheating?

Alice now challenges Bob to another game. Now Alice selects an infinite set of natural numbers X . Bob's task is the same as before: he must guess an infinite series of sets of natural numbers

$$S_1, S_2, \dots$$

Bob wins if, as before, one of his sets $S_k = X$. Otherwise, Alice wins.

The classic proof of Cantor shows the following: if Bob selects a strategy and Alice **knows his strategy**, then Alice can produce an X that makes Bob lose. Recall how she does this. She looks at each of his sets S_i in turn, and decides whether or not to put i into her set X so that $X \neq S_i$.

Perhaps the point that many find unfair is **Alice gets to see Bob's strategy**. This seems like a huge advantage. In the previous games, Bob's strategies were so strong that he could allow her to see his strategies. Now Alice can only be sure of winning if she knows exactly what Bob's strategy is. If Alice does not know his sequence of sets

$$S_1, S_2, \dots$$

then she **cannot** guarantee that she will always win. Whatever she picks for X it *might* just happen to be S_1 . If that is true, then Bob wins instantly.

16.4 Can We Fix This?

Cantor's theorem is correct, there are more infinite sets of natural numbers than natural numbers. But there is no strategy for Alice that always wins the game without "cheating"—without her seeing exactly what Bob's strategy is.

This seems unfair, it seems like Alice is cheating. Is this the reason you might have been puzzled the first time you saw Cantor's proof, or is this the reason you still are worried there is something wrong with it?

Alice has a fix. She tells Bob that she will let him keep his strategy secret, but she will now use a mixed strategy. That is Alice will pick her secret set X randomly. Now she claims she will win with probability 1. She asserts that this means that there is no strategy

$$S_1, S_2, \dots$$

such that Bob always wins. She further argues that this means the set of all sets of natural numbers is not countable. If it was, then Bob would have a winning strategy.

Bob thinks about this last statement. In all the previous games he won because the sets were countable, he agrees that if she can win even some of the time, then Alice is right. Alice says here is how she picks X : each i is in her set with probability exactly $1/2$.

The claim, from Alice, is that the probability $X = S$ for any set S Bob picks is 0. Thus, the probability that her secret set X is in his list is bounded above by the union bound by

$$0 + 0 + \dots$$

Alice thus wins with probability 1, and Bob agrees that he has no infinite list of **all** the sets of natural numbers. He sees that if there were a *complete* list of all infinite sets of natural numbers, then mixed strategy or not, Alice would always lose. Bob would eventually hit Alice's set. Do you agree?

16.5 Open Problems

This discussion is related directly to Chap. 15 on various proofs that the reals are uncountable. There was strong discussion at the time on whether or not the random proof is valid. I think the point here is that the randomization is needed to make the game between Alice and Bob "fair." What do you think?

16.6 Notes and Links

Original post:

<http://rjlipton.wordpress.com/2010/06/11/does-cantors-diagonalization-proof-cheat/>

Liar paradox:

http://en.wikipedia.org/wiki/Liar_paradox

Smullyan and Fitting book:

<http://www.amazon.com/Theory-Continuum-Problem-Dover-Mathematics/dp/0486474844>

<http://rjlipton.wordpress.com/2010/01/20/are-the-reals-really-uncountable/>

William Tutte was one of the founders of modern graph theory. He left his mark on almost all aspects of graph theory, and both solved hard problems and created much of the foundations of graph theory. He was a Fellow of the Royal Society, and won many other awards for his seminal work.

I present a result of mine that is joint with Atish Das Sarma. It concerns a famous open problem of Tutte, and a new approach to the problem.

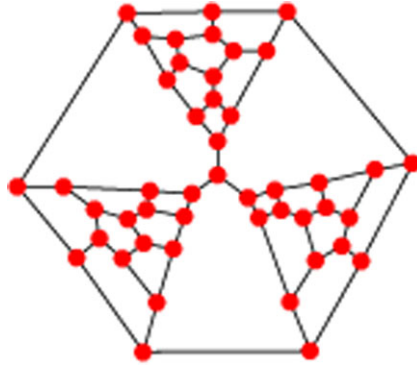
Peter Tait thought he had proved the [Four-Color Theorem](#) in 1880. He did prove that the theorem was equivalent to proving that every cubic bridgeless planar graph was three-edge-colorable. He then “proved” that there was always a three-coloring of the edges by asserting the following: every such graph has a Hamiltonian Cycle. If this were true—it is not—then there is an easy way to three-color the edges. Color the edges on the cycle alternately green and blue. Then, color all the remaining edges yellow. This is a legal three-coloring.

The cycle must have an even number of edges; otherwise, the alternation of colors will fail. I once got a negative referee report back on a paper from an unnamed conference, saying this was a problem—what if the Hamiltonian Cycle had an odd number of edges? Well, the number of edges on such a cycle is the same as the number of vertices. And, of course, every cubic graph has an even number of vertices. The proof of this is simple: the degree of each node is 3 so the n vertices have

$$3n/2$$

edges. Clearly, this forces n to be even, since we cannot have graphs with a fractional number of edges.

The statement of Tait stood for decades, until Tutte thought about it. In 1946 he discovered the following [graph](#):



This graph is easily seen to be cubic, planar, and bridgeless. Also a simple case argument shows there is no Hamiltonian Cycle. Tait was wrong, and his proof of the Four-Color Theorem was thus incorrect. It would be exactly thirty years until a proof of the Four-Color Theorem would be found. It was a heavily computer-dependent proof, but a [proof](#) nonetheless, by Kenneth Appel and Wolfgang Haken.

17.1 Flow Conjectures

Suppose that $G = (V, E)$ is an undirected graph. Direct the edges of the graph in any manner at all; it will turn out not to make any difference. Then a *flow* is a mapping ϕ from the edges E to the integers, so that *Kirchhoff's Rule* is satisfied: the amount flowing into each vertex is the same as the amount flowing out of the vertex. Formally, for each vertex v , let A_v^+ be the edges directed to v and A_v^- the edges directed from v . Then

$$\sum_{e \in A_v^+} \phi(e) = \sum_{e \in A_v^-} \phi(e).$$

A flow is a k -flow if the values $\phi(e)$ are all in the range

$$-(k - 1), \dots, 0, \dots, k - 1.$$

It is called a [nowhere-zero](#) k -flow provided $\phi(e)$ is never 0.

Tutte made a number of conjectures about nowhere-zero flows on graphs, motivated by the coloring theorems for planar graphs. His conjectures have been partially solved, but some are still open. In particular he conjectured:

Theorem 17.1 *Every bridgeless graph has a nowhere-zero 5-flow.*

The best known results to date is François Jaeger's 8-flow theorem and Paul Seymour's 6-flow theorem. In the end notes we list course [notes](#) by Emo Welzl that give a nice survey of what is known about flows.

17.2 Tensor Trick

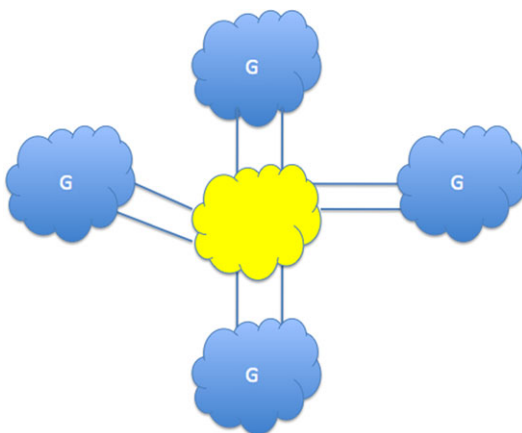
One of the powerful ideas in science and mathematics is the notion of amplification. I discussed this [before](#) in 2009. Sometimes amplification is called the **Tensor Trick**. It is a simple idea, but can yield surprisingly powerful consequences.

Theorem 17.2 *Suppose all bridgeless graphs on n vertices have 5-flows with $o(n)$ zeroes. Then the nowhere-zero 5-flow conjecture is true.*

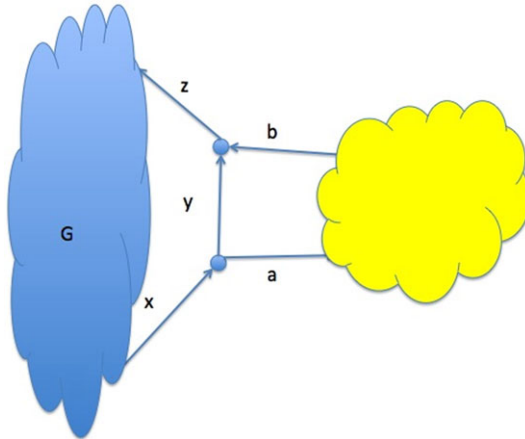
17.3 Sketch of Proof

The proof is based on the notion of amplification. Suppose that G is a counter-example graph. That is, suppose every 5-flow on G has at least one zero. Then we construct another graph H that contains many copies of G . Moreover, if there is a flow on H that has $o(n)$ zeroes, then there is a way to recover a perfect nowhere-zero flow for G .

More precisely, assume that G is a bridgeless graph without a nowhere-zero 5-flow. Create many copies of G and attach them via two edges to a cycle. In the picture below, the blue graphs are copies of G , and the yellow cloud is a cycle.

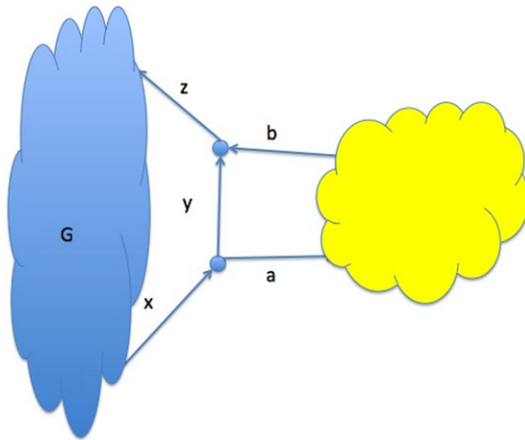


Call this resulting graph H . It is important to see how we do this attachment. We take an edge e of G and split it into three edges x, y, z and add edges a, b to the cycle. See the picture below.



For enough copies of G there must be a 5-flow of H with so few zeroes that one copy of G and its extra edges are all non-zero. The last step of the proof is to argue that we can use these values on G and the extra edges to create a valid nowhere-zero flow for G .

Recall we added the extra two edges to G as in the following figure. We are using the edge labels to also stand for their values, to avoid excessive notation.



The flow must satisfy

$$\begin{aligned} x &= y + a, \\ z &= y + b, \\ a &= b. \end{aligned}$$

The last equality holds since the net flow into a subgraph must be 0. Then it is easy to see that $x = z$. This shows we can collapse and join the edge x and z into one, since they have the same value. This collapse forms the original graph G , since this

replaces the edge we split into three pieces. Note that the resulting flow is non-zero, and this yields a nowhere-zero 5-flow for G .

17.4 Open Problems

Can these tensor results actually be used to prove the nowhere-zero 5-flow conjecture?

Can the same tensor ideas be used on other open questions concerning graphs? We believe it can also work on the Tutte 4-flow conjecture, but there are some details to be checked. The above method does not depend on the flow being a 5-flow. Tutte's nowhere-zero 4-flow conjecture has an additional constraint on the family of graphs, since some bridgeless graphs do not have 4-flows.

The tensor method may apply, we think, to other graph conjectures such as the *Double Cover Conjecture*, which is covered in a [paper](#) by Melody Chan.

17.5 Notes and Links

Original post:

<http://rjlipton.wordpress.com/2010/06/02/the-tensor-trick-and-tuttles-flow-conjectures/>

Previous post on four-color theorem:

<http://rjlipton.wordpress.com/2009/04/24/the-four-color-theorem/>

Tait's conjecture:

http://en.wikipedia.org/wiki/Tait's_conjecture

Nowhere-zero flow:

http://en.wikipedia.org/wiki/Nowhere-zero_flow

Notes on flows by Emo Welzl:

<http://www.ti.inf.ethz.ch/ew/courses/GA09/NZF.pdf>

Referenced post on amplifiers:

<http://rjlipton.wordpress.com/2009/08/10/the-role-of-amplifiers-in-science/>

Paper by Chan:

<http://math.berkeley.edu/~mtchan/cdc.pdf>

We note that Kenneth Appel passed away as this book was entering final production.

We wrote a tribute to him and his daughter at

<http://rjlipton.wordpress.com/2013/05/28/kenneth-and-laurel-appel-in-memorial/>

Picture credits:

Tutte Graph:

http://en.wikipedia.org/wiki/Tait's_conjecture

except Wikipedia has changed its figure.

The other three figures are original.

Basil Rathbone is not a theorist, but was one of the best—in my opinion—actors ever to portray Sherlock Holmes. Like many, I love his “version” of Holmes—just perfect in my opinion.

Our subject is how to claim a major result: how to write it up, and how to release it to the world.

Holmes will play a role in this discussion, of course. Besides being a wonderful character, he is a great problem solver. In the movies Holmes often plays his violin while thinking, much to the annoyance of Dr. John Watson. The good doctor, as he is often called, hates the “noise” of the playing. Sometimes it is at these moments when the good doctor makes a passing remark that unlocks the problem. Perhaps we should all play the violin or something like it when thinking. Or perhaps the message is that working alone is hard—we all need our Watson.

18.1 My Suggestions

You have just proved the greatest result of all time. Okay, say you have really proved a good result. Here are a few suggestions that you may wish to consider in thinking about your result, in writing it up, and in announcing it to the world. They are just my suggestions—feel free to disagree or to add your own ideas to my list.

- **Be Humble:** The Fundamental Theorem of Algebra states that every non-trivial polynomial over the complex numbers has at least one root. The famous Carl Gauss’ doctoral thesis was one of the first “almost correct” proofs of this theorem. It is noteworthy that even the great Gauss entitled his [thesis](#):

A New Proof of the Theorem that Every Integral Rational Algebraic Function of One Variable can be Decomposed into Real Factors of the First or Second Degree

Note the word “new” in his title. He later gave correct proofs; today there are perhaps hundreds of papers published, each with a slightly different proof. Some use algebra, some use analysis, some topology, some complex function theory, and on and on.

- **Be Humble:** Okay it's a repeat, but it is a key point.

If a problem is a well-known open problem it is likely that some new ideas are needed to solve it. You may think you have them. That is fine. Perhaps you do. But you may wish to step back and ask:

What is the *trick* that I have seen that eluded everyone else?

If your answer is, “there is no new idea; it just all works,” then you should be extremely skeptical.

I once heard a story that I cannot confirm, but it is so cool that I have to repeat it. Today we have just about every type of liquid sold to us in plastic containers. There was a time when milk was sold in plastic containers, like today, but soda was still sold in glass bottles. My understanding is that a clever engineer at a large company that made plastic had the “brilliant” insight: why not sell soda in plastic containers? It would be cheaper—plus it would make money for his company, thus he would get a raise.

The engineer told his co-workers his brilliant idea. He could not understand why no one had done it before. They told him: soda in plastic was impossible. No explanation, just that it was impossible.

He went home and decided he would prove them wrong. So he took a plastic milk container, removed all the milk, cleaned it out, and poured in his favorite soda. Then he placed it in his refrigerator and went to bed.

In the morning he went over to his refrigerator and opened the door. He was shocked. They were right. There was his soda container, only it had increased in size many-fold. Clearly, the gas pressure in the soda had made the plastic container expand to fill all the voids. It was like some sci-fi movie—the container had forced its way into every spare inch of space in the refrigerator.

I can only imagine the mess it made getting the container out. Yuck. The problem of putting soda into plastic containers was not trivial—there was a reason that the “obvious” idea did not work.

The story has a great ending, however. Unlike his co-workers this engineer started to work seriously on the problem. He asked “Why did the plastic container fail?” Eventually he was able to create a new type of plastic that had the right properties so that soda would not destroy the container.

- **Be Recursive:** One way is to divide the great paper up into pieces. Each piece should stand on its own, and together they make the whole proof. But each piece by itself is easier to explain, easier to write up, and easier to check. This method of dividing the “secret” will also damp down any popular press issues.

In “Sherlock Holmes and the Secret Weapon,” a 1943 film starring Rathbone, this very trick is used to prevent the theft of a secret bombsight. A bombsight is a device used by planes to decide when to release their bombs. This is a real issue: the plane is moving rapidly, there usually is wind, the bomb falls a great distance before it hits the ground, and so on.

Holmes saves Dr. Franz Tobel and gets him, with his invention, to England. There Tobel divides the invention into four parts. Each part does not reveal the secret of how his new bombsight works.

Of course Professor Moriarty gets involved. There are codes to be broken, escapes to be made since Holmes is captured by his arch rival, and finally a trapdoor that takes the professor by surprise. This was one of the best of the Holmes movies made during the war.

- **Be Partial:** Okay you can prove a huge result. Perhaps a good idea might be to check that your methods at least extend our current knowledge. For example, if you prove that $P \neq NP$, perhaps you could first write a paper that proves that SAT cannot be done in linear time. This is open. It would be a great result, and would show off your new methods. But it would be more credible a claim, and would still be a major result. It would likely cause fewer distractions from the press—that can wait for your full paper.

This happened to some extent in the recent resolution to the Poincaré Conjecture. Grigori Perelman proved much more than “just” the Poincaré, but held out the details on all his results. Experts quickly realized that he might have proved more, but even just proving the conjecture was an immense achievement.

- **Be Computational:** I have proved theorems about n -dimensional space, even though my geometric intuition is close to zero. Once I read a claimed proof resolving an open problem about n -dimensional space. Unfortunately, some key lemma failed even for the case when $n = 1$: the simple line.

The suggestion is to try examples of your results. Not all proofs can be checked in this manner, but many can. Many of the great mathematicians have been tremendous calculators. They tried cases, they computed examples, they made tables of values. Some of these calculations were used to discover patterns, but they can also be used to see if your ideas make sense.

One of the neat examples of this is the story of how Percy MacMahon discovered the rough growth of the partition function. Recall $p(n)$ is the number of ways of writing n as a sum, if we do not count order. Thus $p(4)$ is 5:

$$\begin{aligned} 4 &= 1 + 1 + 1 + 1, \\ 4 &= 1 + 1 + 2, \\ 4 &= 1 + 3, \\ 4 &= 2 + 2, \\ 4 &= 4. \end{aligned}$$

Apparently he kept lists of the number of partitions of numbers and eventually noticed that the number of *decimal digits* in the numbers formed a parabola. This suggested that

$$p(n) \approx e^{c\sqrt{n}}.$$

He was right: although the correct approximate [formula](#) is a bit more complicated.

$$\frac{\exp(\pi\sqrt{2n/3})}{4n\sqrt{3}}.$$

- **Be Clear:** Writing mathematics is not easy. But you must do a reasonable job if you hope to be able to see if it works yourself. And of course also if you do this it will help your readers.

Gary Miller, the “Miller” in the Miller–Rabin primality test, has told me his theory of writing mathematics. He says it is definitions, definitions, and definitions—just like “location, location, location” in real estate. What he means is that one should concentrate on getting the definitions right. They should be clear and crisp. The more precise they are, the better the chances that your proof can be understood. The other advantage of this insight is that even if one of your lemmas has a bug, you might be able to get help fixing it. If the statement of what you tried to prove is clear, then this is possible. If your definitions are murky, or even worse not stated at all, there is no hope.

- **Be Uneven:** What I mean is a more concrete notion of how to be clear in writing down your ideas. A common issue with many wrong proofs is that they spend too much precious time working on the easy part. This is what I mean by being “uneven.” Spend little time on the standard facts, and spend lots of time on the new ideas, on the parts of the proof that are tricky, on the places where you do something new.

I have seen many papers that fell apart after having spent huge space and time working out details to either things that are known, or on things that follow routinely from known theorems. It is a shame the writers do this—sometimes they spend huge amounts of energy on detailed L^AT_EX tables or figures of facts that I would not contest. Then when the big step occurs there is little or no detail.

- **Be Google-Smart:** Know the literature. If you work on any problem you should try to find out as much as you can on what is known. Almost no major work is done in a vacuum. Use the search engines, talk to experts, send friends and colleagues email, and try to see what is known about your problem.

I must admit that this is harder and harder to do these days. There is so much work going on across the world that certainly you may miss something. But you can find out a huge amount of information from the Web and related sources.

When I was a graduate student things were really quite different. There was a time when I flipped through every paper in all of Computer Science. Every one. I sat in the CMU library for months and just looked at every bound journal volume that had anything to do with computing. I will not say I read every paper, but I did scan them all. I loved having the time to do this. Also I loved that I could do it. I suspect today this is hopeless, even trying to do it for a subfield is hard. But the more you know the better your chances will be to succeed.

- **Be . . .:** Add your own suggestion here.

18.2 Open Problems

Good luck and I hope you can indeed prove some new wonderful theorems. I hope these suggestions have helped. What do you think? Also check out Lance Fortnow’s [view](#) on the same subject.

18.3 Notes and Links

Original post:

<http://rjlipton.wordpress.com/2010/09/12/how-to-present-a-big-result/>

Gauss' thesis:

<http://www.robertnowlan.com/pdfs/Gauss,%20Carl%20Friedrich.pdf>

“Sherlock Holmes and the Secret Weapon”:

http://en.wikipedia.org/wiki/Sherlock_Holmes_and_the_Secret_Weapon

Partition function:

[http://en.wikipedia.org/wiki/Partition_function_\(number_theory\)](http://en.wikipedia.org/wiki/Partition_function_(number_theory))

Lance Fortnow on presenting results:

<http://blog.computationalcomplexity.org/2010/09/how-to-write-up-major-results.html>

Elwyn Berlekamp is a famous mathematician who has made seminal contributions to many aspects of both theory and practice. His work on decoders for Reed–Solomon codes, with James Massey, is both beautiful and practical. Without this work many consumer products would be impossible—for example, Reed–Solomon codes are used in disc storage and in wireless networks. Elwyn has also worked on subjects of potentially less practical application, such as combinatorial game theory. He coauthored with John Conway and Richard Guy the book *Winning Ways for Your Mathematical Plays*. He also has the same birthday as I do: September 6th—perhaps the least important fact.

We will talk about work related to combinatorial game theory. When the post was written the 2010 chess world championship match was in progress, and I was inspired by the match to think about mathematical games.

I vividly recall the first time I met Elwyn. I was visiting Berkeley in 1979 on a job interview for a faculty position in the Department of Computer Science. My first appointment was with him, since he was then the chair of the department. I recall sitting down in his office, and before I could say a word he said:

“Dick, I want to know what is the probability that you will accept an offer, if we make you one.”

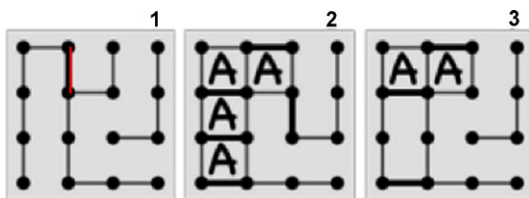
It was a job interview and I had planned answers for all the usual questions: what did I do, what were my reasons for leaving Yale, and so on. But this question surprised me. I thought for a second and said “I would accept with probability $1/3$.” He said fine, and started to ask a more standard question, when I interrupted and said “Was my answer okay?” Elwyn said it was fine—if it had been too high he would have worried why I was so anxious to leave Yale, and if it was too low he would not have wasted any more time on the interview. I never knew if I had said “ $1/10$ ” would he really have terminated the interview. Oh well.

After a while we switched to a favorite topic of his: playing games and in particular playing *dots and boxes*. I almost agreed to play a game, when he told me he had not lost in over ten years. I thought playing a game with the chair—a game I was sure to lose—might not be the best way to impress him. Instead I asked him for

a pointer on how to play the game. I knew this simple game: each player adds lines, if you make a box you go again, and the player with the most boxes wins. I never had thought the game was more than just a kid's game. I was very wrong.

Elwyn knew many strategies, and he explained a basic one to me. He said it would allow me to beat most people, and he was right.

Here is the trick. The thicker edge in figure (1), second vertical from upper left, was just added. A beginner would fill in all the boxes to form the middle figure (2). This loses. A better player would only fill in some of them, see the rightmost figure (3), and would win. Very clever. Elwyn told me to master the game there were many more sophisticated tricks, but even this simple one is quite neat.



19.1 A Colorful Game

There are many combinatorial games. These games by definition are like chess: each player has full knowledge of the state of the game and there is no randomization. A game like [backgammon](#) is not allowed, since it uses dice; a game like checkers is allowed.

A wonderful combinatorial game is one first popularized by the famous Martin Gardner. There has been quite a bit of recent work on this game and its connection to other problems. There are two players, Alice and Bob—perhaps I should use other names, but Alice and Bob seem to be everywhere these days. They fix a planar graph G and a number of colors k . Suppose Alice goes first. She can pick any uncolored vertex and color it with any of the k colors. Bob goes next. He also can pick any uncolored vertex and color it with any color, provided his choice of color is **legal**. He is not allowed to pick a color that would lead to an edge with both endpoints colored the same. The game is continued with the players alternating until the graph is completely colored or until some player is stuck. Alice wins if she gets the graph colored completely, and Bob wins if someone gets stuck.

One way to think about the game is there is an [algorithm](#) to four-color every planar graph. The problem for Alice is that Bob is not trying to help her. Thus the game is like trying to color a planar graph with an adversary making every other color choice. Given this view it is surprising to me there is a winning strategy for Alice when the number of colors is fixed—my guess would have been Bob could always mess up the coloring even with $O(\log n)$ colors. The best known [bound](#), to my knowledge, is due to Xuding Zhu:

Theorem 19.1 *Every planar graph G has $\chi_g(G) \leq 17$.*

Here $\chi_g(G)$ is the least number of colors Alice needs to win the game against any play by Bob. Earlier Hal Kierstead and William Trotter had proved a bound of 33 using previous insights on playing the game on forests. These results are quite surprising in my opinion, and I wonder if they can be used in complexity theory.

19.2 Packing Graphs

There is a beautiful result of Norbert Sauer and Joel Spencer, from 1978, about graph packings. Suppose G and H are two graphs on n vertices. Say they **pack** provided their respective vertex sets can be numbered $1, 2, \dots, n$ so that no (u, v) is an edge in both graphs. Their theorem is:

Theorem 19.2 *Suppose G and H are graphs so that*

$$\deg(G) \deg(H) < n/2.$$

Then, the graphs G and H pack; moreover, the packing can be found in polynomial time.

In the above theorem, $\deg(G)$ denotes the maximum degree of the graph G .

I mention this wonderful result because lately there has been quite a bit of work on extensions. Some look at other restrictions, some at extremal graphs, some at weaker notions of packing. See the [paper](#) “Efficient Graph Packing via Game Colouring” by Hal Kierstead and Alexandr Kostochka (referenced in the end notes) for more details.

One of the most interesting results in their paper is a connection between packing and the coloring game I just talked about. Roughly, Sauer and Spencer showed a connection between game playing and packing:

Theorem 19.3 *If G and H are n -vertex graphs and*

$$(\text{gcol}(G) - 1) \deg(H) + (\text{gcol}(H) - 1) \deg(G) < n$$

then G and H pack.

Here the *game coloring number* $\text{gcol}(G)$ is an upper bound on $\chi_g(G)$ —see their paper for the full definition. Their theorem is stronger than the classic Sauer–Spencer theorem, since

$$\text{gcol}(G) - 1 \leq \deg(G).$$

Another nice discussion of packing problems was given by Hemanshu Kaul in slides from a talk (referenced in the end notes). Kaul stated the longstanding problem called the **Bollobás–Eldridge Graph Packing Conjecture**: if

$$(\deg(G) + 1)(\deg(H) + 1) \leq n + 1,$$

then G and H pack. An example of how packing problems relate to complexity theory lower bounds is a theorem by Péter Hajnal giving an $\Omega(n^{4/3})$ lower bound on the randomized complexity of certain graph properties.

19.3 Open Problems

There are many interesting open problems. An obvious one is to get the exact number of colors needed to win the coloring game on planar graphs, and on other families of graphs. Are there complexity theory applications of the coloring game? Also the packing problem is quite interesting. Are there other sufficient conditions implying that two graphs pack together?

19.4 Notes and Links

Original post:

<http://rjlipton.wordpress.com/2010/05/01/playing-games-and-packing-graphs/>

Reed–Solomon codes:

http://en.wikipedia.org/wiki/Reed-Solomon_error_correction

Berlekamp’s book:

<http://www.amazon.com/Winning-Ways-Your-Mathematical-Plays/dp/1568811306>

Dots and Boxes:

http://en.wikipedia.org/wiki/Dots_and_Boxes

Backgammon:

<http://en.wikipedia.org/wiki/Backgammon>

Xuding Zhu, “Refined activation strategy for the marking game,” *Journal of Combinatorial Theory, Series B*, **98(1)**, 1–18, also available at

<http://portal.acm.org/citation.cfm?id=1321785.1321969>

H.A. Kierstead and A.V. Kostochka, “Efficient graph packing via game colouring,” *Journal of Combinatorics, Probability, and Computing* **18** (2009), 765–774.

<http://www.math.uiuc.edu/~kostochk/docs/2012/cpc09k.pdf>

Slides by Hemanshu Kaul:

<http://www.math.iit.edu/~kaul/talks/GraphPackingTalk-short.pdf>

Péter Hajnal, “An $\Omega(n^{4/3})$ lower bound on the randomized complexity of graph properties,” *Combinatorica* **11(2)** (1991), 131–143, also available at

<http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.30.5982>

Picture credits:

1. Dots and Boxes figure from Wikipedia:

http://en.wikipedia.org/wiki/Dots_and_Boxes

<http://upload.wikimedia.org/wikipedia/commons/thumb/b/b7/Dots-and-boxes-chains.png/300px-Dots-and-boxes-chains.png>

David Johnson is a renowned computer theorist who was awarded the 2009 Knuth Prize for his seminal contributions to theory. He is famous for his many great papers on various aspects of algorithmic theory, his classic book *Computers and Intractability: A Guide to the Theory of NP-Completeness* with Mike Garey, and his continuing interest in the practical aspects of computing. He has been the driving force behind many [experimental](#) studies of how well algorithms perform on “real” data.

We discuss a counterpart to this kind of studies: the issue of galactic algorithms. A **galactic algorithm** is an algorithm that is wonderful in its asymptotic behavior, but is never used to actually compute anything.

The power of asymptotic analysis is that we can predict the behavior of an algorithm without ever running it. This ability is one of the great achievements of modern complexity theory. That we can predict, often with great accuracy, how fast an algorithm would run, if executed, is indeed a major triumph. Yet, often algorithms are created that cannot be run because their running time is too large, or because there are less sophisticated ones that would outperform them on realistic data.

The name “galactic algorithm” is due to Ken Regan—I suggested the concept, but he came up with the name—my original name was terrible. Not actually *terrible*, but not as good as Ken’s. Perhaps there is a better name, but for now this is the one I will use.

The thought behind it is simple: some algorithms are wonderful, so wonderful that their discovery is hailed as a major achievement. Yet these algorithms are never used and many never will be used—at least not on terrestrial data sets. They do, however, play several important roles in our field, which I will discuss in the next section.

20.1 Galactic Algorithms

The danger in giving examples is that I may make some people upset—this is not my intent. If they are the proud creators of an algorithm they may be offended if I point out that their prize creation is galactic—that it will not get used in practice. I think

this misses several key points. There is nothing wrong with galactic algorithms, they abound throughout complexity theory. They are still of immense value for several reasons:

- A galactic algorithm may contain new techniques that can be used to create other algorithms that can be used. This happens all the time.
- A galactic algorithm may eventually become a real algorithm as computer architectures change.
- A galactic algorithm may lead to a series of research advances. The initial algorithm may run in a huge polynomial, but further improvements may yield practical algorithms. This has happen a number of times.
- A galactic algorithm may show us that lower bounds that were conjectured are wrong. This alone could be important and often is a great reason for finding such algorithms. For example, if there were tomorrow a discovery that showed there is a factoring algorithm with a huge but provably polynomial-time bound that would change our beliefs about factoring. The algorithm might never be used, but would certainly shape the future research into factoring.

20.2 Some Examples

Here are some examples of galactic algorithms. Again please keep all of the above in mind. This is **not** any criticism.

- **Nash Equilibrium:** I thought I would start with an algorithm of my own. Not because it is the most important, but to make it clear that I too have worked on galactic algorithms. The [result](#) is the best current bound for non-zero sum games. The [paper](#) was by Evangelos Markakis, Aranyak Mehta, and myself. It proves that every such game has a strategy that is an ε Nash solution, and it finds this solution in time at most

$$n^{c \log n / \varepsilon^2}.$$

Note, this is clearly a galactic algorithm: even for modest approximations the exponent is huge, and the algorithm cannot be used. Yet researchers in algorithmic game theory would be thrilled if one could prove a bound of the form

$$n^{100/\varepsilon},$$

where the dependence on ε is reduced. Note, this would still be a galactic algorithm.

- **Quantum Factoring:** The famous quantum [factoring](#) algorithm of Peter Shor may or may not be a galactic algorithm. It is of course one of the great results in theory, ever. It has sparked funding for research centers on quantum computation that have promoted many other advances.

If and when practical quantum computers are built Peter's algorithm will be one of the first algorithms run on them. Right now it is a galactic algorithm. But, perhaps it is the best example of the importance of galactic algorithms. Peter's

great breakthrough in the discovery of his algorithms has led to the creation of thousands of new results and papers.

- **Graph Minors:** Neil Robertson and Paul Seymour proved their famous graph minor [theorem](#). This theorem proves that for every minor-closed family \mathcal{F} , there is a polynomial-time algorithm to check whether a graph contains a forbidden minor from \mathcal{F} . The algorithms that arise from this general technology unfortunately contain huge constants that make the algorithms galactic. There has been recent [work](#) trying to make the results potentially practical. In any event, David Johnson famously once said:

For any instance $G = (V, E)$ that one could fit into the known universe, one would easily prefer $|V|^{70}$ to even constant time, if that constant had to be one of Robertson and Seymour's.

- **Matrix Product:** One of the most beautiful pairs of results are the initial brilliant breakthrough of Volker Strassen on fast matrix product, and the subsequent deep work of Don Coppersmith and Shmuel Winograd on an even faster matrix product algorithm. These results are used in hundreds—perhaps thousands—of papers. Clearly, these are galactic algorithms. Even the initial one due to Strassen was not used in actual computation for many years. Apparently today as computers have become able to perform huge matrix products it is unclear whether Strassen's idea is truly useful. The Wikipedia [article](#) on it claims that even for matrices of size 1,000 by 1,000 the improvement is marginal.
- **Sorting:** Miklós Ajtai, János Komlós, and Endre Szemerédi [proved](#) the existence of a sorting network of $O(\log n)$ depth and $O(n \log n)$ size. This solved a long-standing open problem, but due to the hidden constants it is a galactic algorithm.

20.3 Open Problems

Galactic algorithms can be of immense importance. They can lead to new methods, new directions, and sometimes can be made practical. If you found a SAT algorithm that runs in time bounded by

$$n^{2^{100}}$$

or proved that any SAT algorithm requires time

$$n^{1+1/2^{100}},$$

then you would be famous. The first would prove $P = NP$ so that would be huge, the latter would prove a non-linear lower bound on SAT. That would also be huge—at least to those of us who have tried to prove such theorems.

Is this classification of algorithms useful? What is your favorite galactic algorithm? What do you think?

20.4 Notes and Links

Original post:

<http://rjlipton.wordpress.com/2010/10/23/galactic-algorithms/>

Johnson's experiments:

<http://dimacs.rutgers.edu/Workshops/EAA/slides/johnson.pdf>

Referenced post on Nash equilibria:

<http://rjlipton.wordpress.com/2009/11/08/rumors-and-playing-games/>

Paper on Nash equilibria:

<http://portal.acm.org/citation.cfm?id=779928.779933>

Shor's algorithm:

http://en.wikipedia.org/wiki/Shor's_algorithm

Robertson–Seymour theorem:

http://en.wikipedia.org/wiki/Robertson-Seymour_theorem

Paper applying the R–S theory:

<http://www-math.mit.edu/~hajiagha/wagnercontraction.pdf>

Strassen's algorithm:

http://en.wikipedia.org/wiki/Strassen_algorithm

Sorting network paper:

<http://portal.acm.org/citation.cfm?id=808726>

Warren Hirsch was a mathematician who started his career in the US Army, and much later retired as a professor emeritus of NYU's Courant Institute of Mathematical Sciences. He is well known for many important results in diverse fields, from combinatorial algorithms, to mathematical models of epidemiology, to his famous geometric conjecture. The last became widely known as the *Hirsch Conjecture*.

Let us discuss how good we are at guessing the truth of mathematical conjectures. Hirsch's famous one was recently disproved by the discovery of a counterexample. For a summary of the state of the art until the recent result look at the paper "An Update on the Hirsch Conjecture" by Edward Kim and Francisco Santos, which is listed in the end notes.

I often have conversations with colleagues on how well mathematicians do at guessing the truth of an open problem. My colleagues often claim that they are pretty good guessers. I think that is not quite true; they certainly do not guess perfectly. One of my main beliefs is that $P = NP$ could be possible. Does that make everyone else a better guesser than myself? I guess so.

In one of Tom Cruise's movies, "A Few Good Men," in the climactic courtroom scene, he yells "I want the truth!" to his opponent, played by Jack Nicholson, who answers, "You can't handle the truth!"

I am interested in how well we can guess the truth—I am sure we can handle it.

21.1 Guesses and the Truth

- **The Guess:** The great David Hilbert thought in 1900 that resolving whether the numbers

$$2^{\sqrt{2}} \text{ and } e^{\pi}$$

are transcendental would take a very long time. He thought such questions might be harder than the Riemann Hypothesis. This is his seventh problem, from his famous list of twenty-three problems.

- *The Truth:* These numbers and more were proved to be transcendental in 1934; the proofs were **found independently** by Aleksandr Gelfond and Theodor Schneider. The methods they used were clever, but were based on already-known technology used by Charles Hermite in 1873 to prove that e is transcendental.
- **The Guess:** David Hilbert also believed there might be a proof that arithmetic is consistent. He proved partial results, and it seemed that the problem might be resolved in the near future. This is his second problem, again from his famous list.
 - *The Truth:* Kurt Gödel’s results, especially his Second Incompleteness Theorem, showed that this was impossible. Consistency is unprovable in arithmetic unless arithmetic is actually inconsistent.
- **The Guess:** Warren Hirsch first stated his conjecture in a letter, dated 1957, to George Dantzig, the inventor of the simplex method for linear programming. Hirsch conjectured a sharp bound on the diameter of the edge-vertex graph of an n -facet polytope in d -dimensional space. The bound was conjectured to be $n - d$. For a very long time this was believed to be true, and most work was on getting better upper bounds. The best known to date is a beautiful quasipolynomial **bound** due to Gil Kalai and Daniel Kleitman.
 - *The Truth:* In May 2010, Francisco Santos, from the University of Cantabria, **announced** he had found a counterexample to the conjecture. Roger Brockett, a famous control theorist at Harvard, once told me:

It is hard to argue with a counterexample.

Santos’s result was big news, and hopefully will lead to further insights into the structure of polytopes.
- **The Guess:** Subhash Khot conjectured that his Unique Games problem is NP-hard. A huge body of lower bounds has been built up on this beautiful conjecture, as discussed in Chap. 6.
 - *The Truth:* Sanjeev Arora, Boaz Barak, and David Steurer have **proved** that the conjecture is unlikely to be true. They have proved that all Unique Games can be solved in time

$$2^{n^\varepsilon},$$

where $\varepsilon > 0$. This shows that Khot’s problem is not nearly as computationally hard as many thought. Of course it does not rule out that Khot is right.

- **The Guess:** Euclid stated his famous Fifth Postulate circa 300 BC. For almost two thousand years mathematicians, professional and amateur, searched for a proof of Euclid’s Fifth Postulate. They all believed it should be provable from his other axioms.
 - *The Truth:* For two millennia they searched for a proof that did not and could not exist. A **book** on this problem and its history, by Jason Bardi, is quite a fun read, definitely recommended. One of the lessons of the search was that for a very long time nobody seriously thought that the postulate could be unprovable, and no one believed looking at geometry without the fifth postulate

made sense. Even the great Carl Gauss was reluctant to announce results he had found on non-Euclidean geometry.

- **The Guess:** The famous George Pólya [conjectured](#) that most numbers have an odd number of prime factors. He made this conjecture in 1919, and it is based on a very reasonable model of numbers.
 - *The Truth:* In 1958 Brian Haselgrove showed it was false below some n of size around 1.845×10^{361} .
- **The Guess:** Leonhard Euler conjectured in 1769 that there were no positive solutions to the equations

$$\sum_{i=1}^n a_i^k = b^k$$

when $k > n > 1$. This is essentially a generalization of the famous Fermat's Last Theorem—set $n = 2$ and $k = 3$, for example.

- *The Truth:* Leon Lander and Thomas Parkin in 1966 found a counterexample for $k = 5$:

$$27^5 + 84^5 + 110^5 + 133^5 = 144^5.$$

Later, in 1986, Noam Elkies found one for $k = 4$:

$$2682440^4 + 15365639^4 + 18796760^4 = 20615673^4.$$

- **The Guess:** Virginia Ragsdale conjectured an upper bound to the number of possible arrangements of real algebraic curves embedded in the projective plane. She made this conjecture in the early 1900s. This was again related to one of Hilbert's problems—the sixteenth.
 - *The Truth:* In 1979 Ilya Itenberg and Oleg Viro [produced](#) counterexamples.
- **The Guess:** Erik Zeeman conjectured that one cannot untie a knot on a four-sphere. I am not a topologist, but it seems like a reasonable conjecture.
 - *The Truth:* After trying to [prove](#) this for almost ten years, one day he worked on the opposite direction, and solved it in hours.
- **The Guess:** John von Neumann conjectured that a topological group G is not amenable if and only if G contains a subgroup that is a free group on two generators. While von Neumann's name is connected with this conjecture, not all agree he believed it to be true. In any event many people did believe this conjecture was true.
 - *The Truth:* The conjecture was [disproved](#) in 1980 by Alexander Ol'shanskii.
- **The Guess:** The conventional wisdom was that bounded width branching programs could not compute the majority function, and many other functions. The rationale was that how could a constant number of bits “encode” the number of 1's in the input?
 - *The Truth:* David Barrington's famous theorem [proved](#) that bounded-width computations can simulate NC^1 , and thus compute the majority function. It is

one of the big surprises in complexity theory. While not the hardest proof, it is one of the most surprising results. The brilliance of David was, I believe, twofold: first looking for an upper bound not a lower bound, and second restricting the bounded-width computations to use only group elements. I definitely guessed wrong and worked hard on the lower bound. Oh well.

- **The Guess:** Most believed that nondeterministic logspace (NLOG) is not closed under complement. It seemed very hard to show that a vertex s is not connected to another vertex t . The obvious way to demonstrate this is to find a “cut” that separates s from t , but a cut could be huge. Thus the conventional wisdom was that logspace did not have the power to show there is *no* path from s to t .
 - *The Truth:* Neil Immerman and Robert Szelepcsényi [proved](#) that NLOG is closed under complement. They did not find a cut. Instead they showed how to count the number of vertices reachable from s by an inductive argument. A very clever result.
- **The Guess:** P is not equal to NP.
 - *The Truth:* Who knows?

21.2 Open Problems

How good do you think we are at guessing the truth of an open problem? Among other experts who have addressed this issue, we note a June 2009 [post](#) by William Gasarch on the Computational Complexity weblog.

I have one specific question concerning the new counterexample to the Hirsch conjecture. Is there any hope to build an *amplifier* for Santos’ bad polytope? I mean can one take his polytope and “tensor” it together and get one of arbitrary dimension that also violates the conjecture? This is not my area so I apologize ahead if the idea is too silly, but as you know I find the idea of [amplifiers](#) very intriguing.

21.3 Notes and Links

Original post:

<http://rjlipton.wordpress.com/2010/06/19/guessing-the-truth/>

Paper on Hirsch conjecture:

<http://arxiv.org/abs/0907.1186>

“A Few Good Men”:

[http://en.wikipedia.org/wiki/A_Few_Good_Men_\(film\)](http://en.wikipedia.org/wiki/A_Few_Good_Men_(film))

Gelfond–Schneider theorem:

http://en.wikipedia.org/wiki/Gelfond-Schneider_theorem

Kalai–Kleitman paper:

<http://arxiv.org/abs/math/9204233>

GeomBlog post:

<http://geomblog.blogspot.com/2010/05/hirsch-conjecture-disproved.html>

ABS paper:

<http://www.cs.princeton.edu/~dsteurer/subexpug.pdf>

Fifth Postulate book:

<http://www.amazon.com/Fifth-Postulate-Unraveling-Thousand-Unraveled/dp/0470149094>

Pólya conjecture:

http://en.wikipedia.org/wiki/Polya_conjecture

Lander–Parkin reference:

<http://rjlipton.wordpress.com/2012/10/26/short-and-tweet/>

Itenberg–Viro counterexamples:

<http://www.springerlink.com/content/l6j787r110101533/>

Erik Zeeman’s idea:

http://en.wikipedia.org/wiki/List_of_disproved_mathematical_ideas

Von Neumann conjecture:

http://en.wikipedia.org/wiki/Von_Neumann_conjecture

Referenced post on Barrington’s theorem:

<http://rjlipton.wordpress.com/2009/03/02/barrington-gets-simple/>

Referenced post on Immerman–Szelepcsényi theorem:

<http://rjlipton.wordpress.com/2009/02/19/we-all-guessed-wrong/>

William Gasarch on guessing:

<http://blog.computationalcomplexity.org/2009/06/are-there-any-longstanding-conjectures.html>

Referenced post on amplifiers:

<http://rjlipton.wordpress.com/2009/08/10/the-role-of-amplifiers-in-science/>

Shimon Even was one of the greats of theory, especially algorithms on graphs—his book *Graph Algorithms* is a classic. Unfortunately, he passed away in 2004 and will always be missed. His *legacy* includes many great results, many strong PhD students, and among many other ideas the notion of a *promise problem*. He invented the latter in 1984 in joint work with Alan Selman and Yacov Yacobi. Oded Goldreich wrote a beautiful paper “On Promise Problems (A Survey in Memory of Shimon Even)” on their importance (referenced in the end notes).

We will raise an interesting “promise problem” about the power of guessing.

I knew Shimon, but never had the pleasure to write a paper with him. He had a wonderful way about him, he was full of energy, and was a joy to talk to on any topic.

I have a story about Even and me that happened when I was at Berkeley and he was visiting there—this was in 1979.

One fall day Shimon came into my office with a big smile on his face. I asked him to sit down, and he proceeded to explain what he found so amusing. The previous spring I had taught the undergraduate automata theory course—at the level of Mike Sipser’s book—except that Mike was taking my class at the time. Not only did Mike take the class but a number of other graduate students did too. One I will call student X . He did okay in my class, and Shimon reminded me that I gave him a B. At Berkeley B was “passing” for a graduate student, so I thought nothing much about giving him this grade.

Shimon smiled and told me:

You were the first teacher ever to give X a B. Ever. He had gotten A’s for every class he ever took since first grade.

I started to say something—but Shimon broke in and said that I had done a great thing. The student was so upset about the B that he was working extremely hard to get his research going. He even had considered leaving Berkeley to go to another department, but decided to stay. In a sense X wanted to prove me wrong.

Shimon thanked me. He said X was probably going to do terrific work and my giving him that single B may have been the spark he needed. I do not recall saying

anything back to Shimon, but today X is a superstar. I would like to think that I helped in some small way. Perhaps I should give out more B's.

Let's turn to discuss a simple promise problem that I have been looking at recently.

22.1 The Promise Problem

I will call the problem TWOPATHS. A function $f(x)$ is given that is computable in polynomial time on bit strings of length n . The function is 1-to-1 but can be "undefined" on some strings, and we also stipulate $f(x) \neq x$ for all x . Let's use $x \rightarrow y$ to denote that $f(x) = y$. In a natural way the function defines a directed graph with \rightarrow as the edge relationship. Every node has out-degree at most 1 because f is a function, and in-degree at most 1 because f is injective. It also has no self-loops because $f(x) \neq x$. Hence the graph is a disjoint union of paths and cycles.

We further stipulate that there are two special start nodes s_1, s_2 and two special end nodes t_1, t_2 , which are given as part of the input. The **promise** is the resulting graph has exactly two paths, each starting from an s_i and ending in an t_j . Thus, either

$$\begin{aligned} s_1 &\rightarrow a_1 \rightarrow a_2 \rightarrow \dots \rightarrow a_k \rightarrow t_1, \\ s_2 &\rightarrow b_1 \rightarrow b_2 \rightarrow \dots \rightarrow b_l \rightarrow t_2 \end{aligned}$$

or

$$\begin{aligned} s_1 &\rightarrow a_1 \rightarrow a_2 \rightarrow \dots \rightarrow a_k \rightarrow t_2, \\ s_2 &\rightarrow b_1 \rightarrow b_2 \rightarrow \dots \rightarrow b_l \rightarrow t_1. \end{aligned}$$

The **task** is to determine the end point of the paths: does the path from s_1 end in t_1 or t_2 ?

Notice that the property of t_1, t_2 being sinks is not part of the "promise." It does not have to be because it is already decidable in polynomial time: being able to tell that $f(t_1) = \text{undefined}$ and $f(t_2) = \text{undefined}$ is considered part of f being computable in polynomial time. The property of s_1, s_2 being sources is not necessarily as easy, because f need not be invertible in polynomial time. However, the property of x not being a source node belongs to the complexity class UP, because then there is a unique y such that $f(y) = x$. The class UP is a subclass of the class into which we will classify this task, so it is unimportant whether s_1, s_2 being sources is part of the givens or part of the promise.

The property that the graph consists of exactly two paths, however, is not so easy. In fact it is polynomial-space complete: Consider functions f that arise as the next-configuration function (for some input size n) of space-bounded Turing machines M that decide the quantified Boolean formulas problem (QBF). For deciding QBF we can label configurations so that in the "yes" case the graph consists of exactly

one path from the start configuration to the accepting one, and in the “no” case it has two. This makes it all the more surprising that under the *promise* of being in the two-path case, the complexity of the task drops down to a class that is commonly believed to be lower than polynomial space.

22.2 The Complexity of TWOPATHS

I do not know the exact complexity of the TWOPATHS problem. The following theorem is the best upper bound that I know:

Theorem 22.1 *The TWOPATHS problem is in $\oplus\text{P}$.*

I find this theorem to be unexpected. The obvious algorithm is to iterate the function f on s_1 until an end is reached. Since the paths can be exponentially long, this does not run in polynomial time. Yet the problem can be reduced to the counting class $\oplus\text{P}$. Recall this counts the number of accepting computations modulo 2.

For a high-level sketch of the proof, suppose we are given the graph G defined by the function f : we know that G consists of two paths. The start vertices are s_1, s_2 and the end vertices are t_1, t_2 . The key idea is to add the edges $t_1 \rightarrow s_1$ and $t_2 \rightarrow s_2$ to form the graph H . Then there are two cases: If the path starting with s_1 ends in t_1 we get two cycles; if it ends in t_2 we get one cycle. Thus, to solve the problem we need to decide whether H consists of one cycle or two.

This we can do by a trick I discussed before. We can view H as defining a permutation on the vertices. The number of inversions of a permutation can be used to tell how many cycles the permutation has modulo 2. But, the number of inversions modulo 2 is easily reduced to a problem that can be computed by $\oplus\text{P}$.

The number of inversions is defined as follows as follows: for each pair of vertices x and y they form an inversion provided

$$x < y \quad \text{and} \quad f(x) > f(y).$$

Finally, why is this only a promise-problem solution? The predicate for counting inversions yields an implicit odd/even number of witnesses for all kinds of graphs. For graphs that did not arise from the promise, the implication for permutations is null and void. Thus the language L arising from this predicate belongs to $\oplus\text{P}$, but L is not restricted to graphs in the *promise set* Q . An ordinary language reduces to the promise problem if it reduces to L , provided the range of the reduction is entirely within Q . Thus we can say that promise problems are hard for, and even complete for, ordinary complexity classes.

22.3 Open Problems

What is the complexity of TWOPATHS? Can we prove it is complete for some known class? Can the bound of the theorem be improved? It seems hard to believe that the problem is in, for example, NP. How would guessing help “follow” a path that can be exponentially long?

22.4 Notes and Links

Original post:

<http://rjlipton.wordpress.com/2010/09/05/promise-problems-and-twopaths/>

Shimon Even, *Graph Algorithms*, Cambridge University Press, 1979, new edition 2011:

<http://www.wisdom.weizmann.ac.il/~oded/even-alg.html>

Oded Goldreich, “On promise problems (A survey in memory of Shimon Even),” Electronic Colloquium on Computational Complexity, ECCC TR05-018, February 2005:

<http://eccc.hpi-web.de/eccc-reports/2005/TR05-018/index.html>

For more on parallel counting of permutations, see this 2009 post:

<http://rjlipton.wordpress.com/2009/09/04/counting-cycles-of-a-permutation-in-parallel/>

Subsequently it was noted by several that the promise problem is indeed complete for $\oplus P$.

Matei David, Periklis Papakonstantinou, and Anastasios Sidiropoulos are three young complexity theorists who are linked by three things, at least. They all were part of the Toronto theory group at the same time, either as students or postdocs; they all are interested in various aspects of complexity theory, especially randomness; and they wrote a wonderful paper together.

We will discuss their extension of the famous Nisan pseudorandom number generator that can fool space-limited computations. This extension is their joint paper (DPS). I think the result is quite pretty and has potentially many consequences.

Nisan's generator comes with an expressed warranty. It starts with the usual kind of legal language:

FOR USERS, WHO ARE COVERED BY PROTECTION LAWS OR REGULATIONS IN THEIR COUNTRY OR, IF DIFFERENT, THEIR COUNTRY OF RESIDENCE, THE BENEFITS CONFERRED BY THIS WARRANTY ARE IN ADDITION TO ALL RIGHTS AND REMEDIES CONVEYED BY SUCH PROTECTION LAWS AND REGULATIONS.

But the critical part for us is this:

The users of this random number generator assume all liability. It is only to be used against space restricted machines, any other use is prohibited. Users must select a really random number for the seed—any attempt to use other than real random bits for the seed are a violation of the operating conditions of the generator. Further, each generated bit is to be read once, not twice, nor three times. Any deviation from this is a violation of the operating conditions and is prohibited.

Note, I converted the above text into lowercase. It is well known that all-caps is hard to read—IS THAT WHY WARRANTIES USE ALL CAPS?

The DPS paper violates the warranty—or you can say, it enables a less-restrictive one to be written. Let's see what happens.

23.1 The Nisan Generator

I will first explain Noam Nisan's generator in the way they use it in their paper. Let U_n and U_m be the uniform distributions on $x \in \{0, 1\}^n$ and $s \in \{0, 1\}^m$, respectively. The 1-norm is the sum over all states $q \in Q$ of the absolute value of the difference between $\Pr_x(Q(x) = q)$ and $\Pr_s(Q(G(s)) = q)$, where $Q(x) = q$ means that the FSM goes from its start state 1 to state q reading x .

Let $G : \{0, 1\}^m \rightarrow \{0, 1\}^n$ be computable in polynomial time in n and let $\epsilon, w > 0$. Say that G is a *pseudorandom generator against Finite State Machines*—for space w with parameter ϵ —if for any FSM Q with 2^w states,

$$\|Q(U_n) - Q(G(U_m))\|_1 \leq \epsilon.$$

Here U_k is the k -bit string drawn from the uniform distribution. See their [paper](#) for all the details.

Nisan's theorem is:

Theorem 23.1 *There exists $c > 0$ such that for any $N > 0$, there exists a function $G : \{0, 1\}^{N^2} \rightarrow \{0, 1\}^{N^{2^{cN}}}$ computable in time polynomial in 2^{cN} which is a pseudorandom generator against FSMs for space cN , with parameter 2^{-cN} .*

The definition allows that the machine Q can only read the “random” bits $G(U_m)$ once. They suggest that you think of the bits as placed on a special *random* tape that is one-way. The machine reads a bit, uses it any way it wishes, and then can move on to the next bit of the tape. The only way it can “recall” the exact bit is by storing it in its limited storage.

23.2 Read It Again?

Consider the case where the FSM is limited to logspace, when the seed is $O(\log^2 n)$ in size, and the generator creates a polynomial number of pseudorandom bits. In this case they ask: can Nisan's generator be “misused” and still work? The somewhat surprising answer is yes. They show that it is possible to allow the FSM to treat the random tape as a read-twice tape: that is the machine can read the random bits, and then read them once again. The cost of this is a relatively small change in the parameters of Nisan's theorem.

This is done without any change to the generator, at all. It really is simply **extending** the generator's warranty. The formal lemma that shows this works is:

Lemma 23.2 *Let $G : \{0, 1\}^m \rightarrow \{0, 1\}^n$ be a PRG against FSMs for space $2s$ with parameter ϵ . Then, the **two-pass** version is a PRG for space s with parameter $\epsilon \cdot 2^{2s}$.*

Note, the space decreased by a factor of 2 and the error parameter went up by a factor of 2^{2s} . What I find beautiful is that this lemma allows them to prove by simply iterating it not too many times— $O(\log \log n)$ —that we get a PRG which fools a

logspace machine, and allows multiple passes over the random tape. Again see their [paper](#) for all the bounds.

This is a very cool result.

23.3 Open Problems

They point out, without any proof, that if there is a PRG that does polynomially many passes and fools logspace machines then $L \neq NP$. They do show that such a generator cannot be the Nisan one—it does break if you try to allow that many passes.

I believe I get how they prove this claim, which is an “aside” in the final part of their paper. I actually conjecture that a much stronger statement should be true. Staying with Nisan’s generator, as an example, I believe that allowing $\log^k n$ passes for any constant k would prove $L \neq NP$. Actually, I think even more is possible. We can await more in the future on the details of such claims.

23.4 Notes and Links

Original post:

<http://rjlipton.wordpress.com/2010/10/31/the-warranty-on-nisans-generator/>

DPS paper:

http://www.cs.toronto.edu/~papakons/pdfs/nisan_RNC.pdf

Ryan Williams is one of the top complexity theorists in the world, young or old. He has just released a draft of a result that moves us closer to understanding the mysteries of circuit complexity classes.

We review a short summary of what Ryan did, and how he did it. I think it is potentially the biggest, best, most brilliant—okay I really like his result.

The beauty of his result is that while the details are non-trivial it follows a schema that I believe could be used to solve other open problems. I think it could even solve...

24.1 The Result

Ryan proves the following result:

Theorem 24.1 *The nondeterministic-time class $\text{NTIME}(2^n)$ does not have nonuniform ACC^0 circuits of polynomial size.*

Actually he proves several other related theorems.

Note, the theorem shows a nonuniform lower bound. That is a breakthrough. Proving circuit lower bounds that allow nonuniformity is extremely difficult. Somehow the power of allowing the circuit to vary with the input size is very difficult for us to understand. For example, we still do not know whether polynomial time is contained in linear-size circuits—see the end notes for more.

24.2 The Proof Schema

If you wish to understand what Ryan has done see his [paper](#). It is well written and explains the details in a clear and careful manner. What I will try to do is give a 50,000-foot view of the proof.

- **Make an Assumption:** Ryan assumes, by way of contradiction, that NEXP has polynomial size ACC^0 circuits. Recall these are constant-depth circuits that can have unbounded fan-in gates AND, OR, and any MOD gate for any $m > 1$.
- **Use the Assumption:** He then proves that if ACC^0 circuit satisfiability can be solved faster than exhaustive search there is a contradiction. The contradiction is with the classic nondeterministic-time hierarchy theorem. This part of the proof is based on his STOC 2010 paper, with an additional twist. He needs the assumption he made in order to prove the simulation step—that step cannot be proved unconditionally.
- **Break the Assumption:** He then needs to show that the SAT problem restricted to ACC^0 circuits can be done slightly better than 2^n . But how do you beat the naive SAT algorithm for ACC^0 circuits? Ryan shows that he can beat it by a clever application of a clever algorithm: a special case of matrix multiplication. He uses the following beautiful result of Don Coppersmith:

Lemma 24.2 *The product of an $N \times N^{0.1}$ matrix with an $N^{0.1} \times N$ matrix can be done in $O(N^2 \log^2 N)$ operations over any ring.*

See his paper for the details. Wonderful.

24.3 Open Problems

Can Ryan's results be improved? What happens if matrix product can be done faster? The last comment I will leave to Ryan:

“Improved exponential algorithms for satisfiability are the only barrier to further progress on circuit lower bounds for NEXP.”

24.4 Notes and Links

Original post:

<http://rjlipton.wordpress.com/2010/11/08/a-breakthrough-result/>

Referenced post on P and NP vis-à-vis linear-size circuits:

<http://rjlipton.wordpress.com/2009/02/12/is-np-too-big-or-p-too-small/>

Williams paper:

<http://www.cs.cmu.edu/~ryanw/acc-lbs.pdf>

Joel Seiferas is a complexity theorist who has proved many important results. One of his results that is especially critical in modern complexity theory is the famous nondeterministic-time hierarchy theorem. I also note that one of his most recent papers is on attempting to make the famous AKS sorting algorithm less [galactic](#).

We outline in more detail how Ryan Williams' great lower bound [result](#) goes. The "we" is Ken Regan and I.

Joel, besides his theorems and results, almost singlehandedly maintained an indispensable resource for theoretical computer science. This was in the days before Citeseer, DBLP, and other online bibliographies, even before Netscape and AltaVista. He typed citations of papers as they appeared in an abbreviated format, producing a compact plaintext file that was easy to browse. There were no page or issue numbers, just enough so you could find the paper—in your physical library. Joel's handiwork is still [online](#).

Recall Williams proved that $\text{NTIME}(2^n)$ does not have ACC^0 circuits of polynomial size. The first class is a **uniform** complexity class—it consists of languages accepted by nondeterministic Turing Machines that can run in exponential time. The class is uniform because any language in the class is defined by a program: the program can be clever, can do different things on different length inputs. But it is a fixed finite program.

The second is a **non-uniform** complexity class—it consists of languages defined by constant-depth circuits with special gates. The gates are allowed to compute certain function types based on the sum of the inputs—and additionally, in ACC^0 circuits, NOT gates are allowed too. Each gate sums its input bits and then decides whether to output a 0 or a 1 based on the sum:

- (1) **OR**: Output 1 if the sum is non-zero.
- (2) **MOD**: Output 1 if the sum is zero modulo a fixed $m > 1$.

Since ACC^0 is non-uniform the circuit can depend on the input length n in *any manner at all*. This is why it is called non-uniform. You can think of it as defined by a "program," but one that is **infinite** in length:

If $n = 1$, then use this circuit. . .

If $n = 2$, then use this circuit. . .

⋮
 If $n = 14159265358979323846$, then use this circuit. . .
 ⋮

The reason there is such a mystery about the behavior of circuits is this ability to vary with the input length n , the ability to change what they do with the value of n . Nonuniform circuits can exploit the *existence* of problem-solving structures that are hard to compute, such as witness sets that arise from applications of the “Probabilistic Method.” Even so, it seems hard to believe that this ability really can be exploited to make problems *much* easier, but even among quite natural problems we cannot rule this out. The “we” here is not Ken and I, but is everyone.

25.1 The Landscape of Ignorance

In this section we summarize what is known about the various complexity classes: ACC^0 , P, NP, PP, EXP, $\text{DTIME}[2^n]$, $\text{NTIME}[2^n]$, and so on. You can *skip* this section and still understand the outline of Ryan’s proof. If you read on do not let the abundance of sans-serif words scare you. The general take-away is that we know some things, but not as much as we would like.

We do not know whether uniform ACC^0 is properly contained in NP, but we do know it is different from PP by a theorem of Eric Allender and his student Vivek Gore. Eric later extended this to separate uniform TC^0 from PP, and Ryan’s paper also builds on Allender–Gore. Note that since $\text{PP} \subseteq \text{EXP}$ and EXP has simple reductions to $\text{DTIME}[2^n]$, it was known that $\text{DTIME}[2^n]$ and hence $\text{NTIME}[2^n]$ do not have uniform TC^0 circuits. For nonuniform circuits, however, the best-known separation from nonuniform ACC^0 was basically the same as for TC^0 and P/poly itself: a class with somewhat more alternation than $\text{NTIME}[2^n]$ that was defined in terms of “Merlin–Arthur games.”

Understand, most people believe that even nonuniform ACC^0 is really weak, in some sense exponentially weaker than P. And P is exponentially weaker than $\text{DTIME}[2^n]$, let alone $\text{NTIME}[2^n]$. Thus two exponentials and a little more were still not giving us a separation with nonuniform circuits—until Ryan’s paper, that is.

This is why circuits are so mysterious. Thanks to Ryan they are a bit less today.

25.2 Outline of the Proof

Suppose that you wish to prove that $\text{NTIME}(2^n)$ is not contained in the circuit family ACC^0 of polynomial size. Here is how you might proceed, especially if you are Ryan.

There is no cost in assuming that this is false: that $\text{NTIME}[2^n]$ is contained in the circuit class ACC^0 . You can assume this—of course you then must show that it leads to a contradiction. Godfrey Hardy once said:

Reductio ad absurdum, which Euclid loved so much, is one of a mathematician's finest weapons. It is a far finer gambit than any chess play: a chess player may offer the sacrifice of a pawn or even a piece, but a mathematician offers the game.

The plan is to use the assumption to show that we can simulate any nondeterministic Turing machine that runs in time 2^n in time $o(2^n)$. Provided the little $o(2^n)$ is not too close to 2^n , then by the wonderful time hierarchy for such machines you will have your contradiction. This theorem, due to Joel Seiferas, Michael Fischer, and Albert Meyer is one of the great [results](#) of complexity theory. Note, it is critical that the machines are nondeterministic, since the corresponding hierarchy theorem for deterministic machines is much weaker.

We will now outline the steps of the proof. We will leave out the detailed constants and calculation parts—proofs like this have to check various bounds carefully. They are not that hard, but will detract from giving you the overall flow of the beautiful ideas. We hope this outline is helpful.

- **Pump up the Volume.** We have assumed that $\text{NTIME}[2^n]$ is in ACC^0 . This is a powerful assumption, but it implies even more. So the first thing to do is to use it, via standard arguments, to show that another complexity class has ACC^0 circuits. Let's just call this class N^* . For the expert it is E^{NP} .
- **All the Small Things.** Steve Cook's famous theorem showed that any nondeterministic computation could be encoded into a 3-SAT problem. Since he first proved this almost 40 years ago, there have been improvements made to the exact encoding. At first any encoding was fine, later for more refined results more efficient encodings became necessary.

The next step is to assume that L is a language in $\text{NTIME}[2^n]$. Then we can conclude that there is a very simple circuit $\text{Des}(k)$ that describes the set of 3-clauses. This is a critical step and uses the better encodings. The point is important: *we cannot afford to write down the clauses that describe the computation of L* . They would be at least 2^n long, actually longer, and we would already be in trouble. How can we run faster than 2^n if we write down an object that size? We certainly cannot. Hence, we need a small circuit that **indirectly** describes the clauses. The circuit Des can be used to get the k^{th} clause in polynomial time. Again the key is that the describing circuit is tiny compared with the set of clauses it describes.

This type of Cook's Theorem is called **SUCCINCT SAT**. While the details of proving it are quite neat, the idea that such a describing circuit exists should not be too shocking. The point is that Cook's Theorem is quite "regular" and this allows all to work.

- **Can I Get a Witness?** We have to decide if a given input x is in L . We know that this is true if and only if—I hate iff—the clauses described by the circuit Des are satisfiable. So the next step is to try and guess an assignment that makes all the clauses happy. Again we have a problem: the assignment is huge, and so we cannot even think about writing it down. But we have a powerful assumption: we have that N^* is in ACC^0 of polynomial size. Thus the idea is to use an N^* machine to guess the assignment, but again use our assumption to get another polynomial-sized circuit Wit . This circuit has the property that $\text{Wit}(i)$ is the assignment to the i th variable used in the clauses.

There is a technical point here—actually the reason we “pumped up the volume” to the class N^* . In order to show that Wit exists we use a standard trick: we need to select the “first” satisfying assignment. If there were many assignments, then there would be the possibility that our witness circuit could be wrong. This step needs N^* and uses a binary search trick to get the first assignment. Are you still with us? This last part is a technical but necessary part; you can just avoid thinking about it if you like.

- **Trust Yourself.** Let’s summarize: we have a small circuit that describes the clauses and a small circuit that describes an alleged witness. The last step is to check that they are in agreement, that is to verify that every clause is satisfied. Again there are way too many clauses to do this by brute force. So what do we do?

This is Ryan’s main insight: this is another SAT problem itself, but it is about a polynomial-sized ACC^0 circuit. The SAT problem checks whether or not each clause is satisfied by the assignment encoded by the witness circuit. The good news is this is a small object—it is only polynomial size. The bad news is it is a SAT problem for a quite powerful class of circuits. But the last insight is that Ryan can prove this can be done in $o(2^n)$.

- **Don’t Worry About the Government.**

Finally, Ryan exploits the fact that ACC^0 circuits C_n of size s can be represented as a simple function of many simple subcircuits. Namely, there are circuits D_n that have just two levels: a single output gate that computes a symmetric function of $s' = 2^{\text{polylog}(s)}$ -many level-1 gates, where each level-1 gate is an AND of input bits, some of which may be negated. It should be noted that his top-level argument does not *use* the standard feature of such “SYM+” circuits that the AND gates have only $\text{polylog}(s)$ fan-in, though this property is needed inductively in their construction, and is achieved by his paper’s appendix. This is like having a system of very small local governments with overlapping jurisdictions, united by a symmetric, hence unbiased, but very simple central authority. Would this be a better system of government, without state and county layers? Since it would need many thousands of congressmen, probably not.

It is also important that ACC^0 circuits are closed under a second kind of “localization”: For any circuits C_n and $\ell < n$, we can form new circuits C'_m with $m = n - \ell$ inputs, by choosing a set J of ℓ -many input variables, and taking a big OR of 2^ℓ -many circuits C_m^a . Each of the C_m^a is obtained by fixing the variables in J to an assignment $a \in \{0, 1\}^\ell$. Then, and only then, do we apply the above to get D'_m from C'_m . Applications of dynamic programming, replacing a fast multiplication method for “skinny” matrices in the original draft, then complete Ryan’s proof.

25.3 Open Problems

Did this outline help you? Did it make any sense?

We have been thinking about how to axiomatize the properties of ACC^0 circuits (in particular, the $n^{O(1)}$ -size case of the class ACC^0), so as to state general conditions for a circuit class \mathcal{C} to enjoy Ryan’s separation theorems. We would like to get this

for $\mathcal{C} = \text{TC}^0$ or even $\mathcal{C} = \text{P/poly}$. Well, everyone would like to get this—there has been much discussion of how possibly to extend Ryan’s methods, and we hope we can help in the conversation.

25.4 Notes and Links

Original post:

<http://rjlipton.wordpress.com/2010/11/16/how-ryans-circuit-lower-bound-works/>

Referenced post on galactic algorithms:

<http://rjlipton.wordpress.com/2010/10/23/galactic-algorithms/>

Ryan Williams’ paper:

<http://www.cs.cmu.edu/~ryanw/acc-lbs.pdf>

Seiferas bibliography link:

<http://liinwww.ira.uka.de/bibliography/Theory/Seiferas/index.html>

Seiferas–Fischer–Meyer paper:

<http://portal.acm.org/citation.cfm?id=322061>

Post by Scott Aaronson:

<http://scottaaronson.com/blog/?p=472>

Post by Lance Fortnow:

<http://blog.computationalcomplexity.org/2010/11/is-ryans-theorem-only-interesting-to.html>

Post by Luca Trevisan:

<http://lucatrevisan.wordpress.com/2010/11/08/a-circuit-lower-bound-breakthrough>

The last three links were given in the post itself. In response to these items and other comments we revised the outline accordingly:

- (1) Assumption A1: $\text{NTIME}[2^n] \subseteq \text{ACC}^0$.
- (2) Goal: Simulate a general $\text{NTIME}[2^n]$ machine N in nondeterministic time $2^n/g(n)$, for big enough $g(n)$ to contradict A1.
- (3) A1 \implies A2: $\text{NEXP} \subseteq \text{ACC}^0$.
- (4) $\text{NEXP} \subset \text{P/poly}$ (which is weaker than A2) still implies A3: every language in NEXP has poly-size witness-describing circuits. In turn this implies A3’: every yes-instance of Succinct 3SAT has a succinct satisfying assignment. (This uses the Impagliazzo–Kabanets–Wigderson technique, which uses derandomization.)
- (5) $\text{A3}' \wedge \text{P} \subseteq \text{ACC}^0 \implies$ A4: the Succinct 3SAT instances obtained from the cases where the machine N accepts its input x have ACC^0 circuits describing a witness. (This routes through the proof for the E^{NP} case; the binary-search step for E^{NP} is not part of where these roads join.)
- (6) Then A4 feeds into the “Trust Yourself” section.

We also took a second pass at explaining the paper in 2012:

<http://rjlipton.wordpress.com/2012/11/13/the-ryan-williams-combine/>

Victor Klee was one of the greatest geometry experts ever, and is known for many wonderful results and problems. For example, he was the first to raise the question now called the “art gallery problem,” and also the question now called Klee’s measure [problem](#). The latter is especially relevant, since it is a pure computational geometry problem that is still not completely settled.

When I wrote this post, in late 2010, I expressed that “I wish to share a feeling that I have, a feeling that is very positive, yet is just a feeling. I think that complexity theory is on the verge of major breakthroughs of all kinds. I feel this is already starting to happen, and I wish to share the reasons I feel this way.” Not all of my thoughts came true quickly, but we have still seen progress.

Our field, like most areas of science, has times of huge advances and times when the advances are small. I think that we are in a time of potentially major advances across the entire spectrum of theory. It is an exciting time, and much of that excitement is in seeing problems solved or at least partially solved that were open for years and sometimes decades.

I think Klee would have liked one of the new results very much. I met him only once or twice, and we never had a chance to work together. But I believe that he would be excited to see his famous lower bound on the simplex method finally improved.

26.1 Three Big Results

I feel that there are many results already that are great advances, but I also feel that they point to even greater results in the near future. The next few weeks, months, and years could be a most exciting time for computational complexity.

- **Unique Games:** We have already discussed the [result](#) of Sanjeev Arora, Boaz Barak, and David Steurer on the Unique Games Conjecture, proving that Unique Games can be approximately solved in time 2^{n^ϵ} for any $\epsilon > 0$. Their result does leave an interesting open problem: *Is the approximation problem for unique games in polynomial time?*

- **Circuit Lower Bounds:** The breakthrough paper of Ryan Williams is another clear example. His beautiful [result](#) is that $\text{NTIME}[2^n]$ does not have non-uniform ACC^0 circuits of polynomial size. His result leaves open many problems including: *Can the circuit class be improved to general polynomial-size circuits?*
- **Simplex:** The result of Oliver Friedmann, Thomas Hansen, and Uri Zwick is another wonderful [example](#). They have proved strong, almost exponential, lower bounds on randomized pivoting rules for the *simplex method*. See Gil Kalai, who is an expert in this area, for [comments](#) on their result. Gil also is the discoverer of one of the best methods known for running Simplex: his method requires at most

$$2^{\tilde{O}(\sqrt{m})},$$

for m constraints.

Let me give a short history of the search for “good” pivoting rules for Simplex. The Simplex method is really a family of algorithms: at each step the algorithm essentially moves around the vertices of the polytope defined by the linear constraints. The key is the decision of where to move next: the *pivot rule*. In practice there are many simple rules that work extremely well; however, the complexity question is to find a rule that always works efficiently.

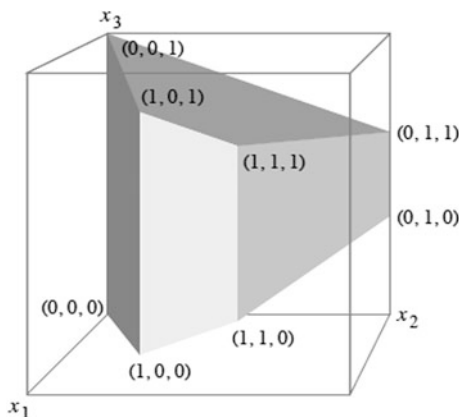
In 1977 Klee and George Minty [discovered](#) a “difficult” linear program. Their program was based on polytopes that are now called *Klee–Minty cubes*:

$\min x_n$:

$$0 \leq x_1 \leq 1,$$

$$\varepsilon x_{i-1} \leq x_i \leq 1 - \varepsilon x_{i-1}$$

for $2 \leq i \leq n$ and $0 < \varepsilon < 1/2$. Here is a picture for $\varepsilon = 1/3$, it is from the [paper](#) by Bernd Gärtner, Martin Henk, and Günter M. Ziegler.



Klee and Minty proved that the naive pivot rule had a worst-case exponential running time on these programs. Quickly other rules were suggested, and for a long time various random rules were considered potential winners. That is, many researchers believed these rules took expected polynomial time.

I am definitely not an expert in this area, but I am amazed that Friedmann, Hansen, and Zwick were able to prove that various random pivoting rules do not run in expected polynomial time. Finding a worst-case example for any algorithm can be hard. Finding one for a randomized algorithm is harder: the example must cause the algorithm to fail on paths weighted by their probability, something that is often difficult to show.

Finally, they exploit a connection between linear programming and certain games that seems quite important. We have long known that games and linear programming are related, but that the relationship could be used to prove a lower bound seems quite surprising—at least to me. Their result does leave an interesting open problem: *Is there a pivot rule that is even a reasonable conjectured efficient one?*

26.2 One More

- **Learning Theory:** The result of Amir Shpilka and Avishay Tal is my last example of a great new result. They **prove** new bounds on the Fourier degree of symmetric Boolean functions.

The truth is that I am not quite unbiased. I was part of a team that had the previous best result. Mihail Kolountzakis, Evangelos Markakis, Aranyak Mehta, Nisheeth Vishnoi, and I **studied**: What is the smallest t such that every symmetric Boolean function on k variables—that is not a constant or a parity function—has a non-zero Fourier coefficient of order at least 1 and at most t ? Let $\tau(k)$ be the smallest such t . Our main result was that $\tau(k) \leq 4k/\log k$ for symmetric Boolean functions on k variables. The question arises in learning theory and in particular in the special case of the **Junta problem** for symmetric Boolean functions.

Our proof involves careful analysis of the structure of a symmetric Boolean function. Even though we could only prove $o(k)$, we did conjecture that the correct upper bound was much better—perhaps even $O(1)$. Shpilka and Tal prove a theorem that is much stronger than our result. Much. Further their proof is quite pretty and uses some quite interesting connections between this pure complexity question and the structure of prime numbers. Very neat.

They use a deep result from the distribution of primes. Define $\Gamma(m)$ to be

$$\max\{b - a \mid 1 \leq a < b \leq m \text{ and there is no prime in the interval } (a, b]\}.$$

It is known that $\Gamma(m) \leq m^{0.525}$; that is, it is known to those who study the deep structure of the primes. It is believed that there are better bounds on $\Gamma(m)$, and it has been conjectured that $\Gamma(m)$ is $O(\sqrt{m})$ and perhaps even $O(\log^2 m)$.

Shpilka and Tal prove the following theorem:

Theorem 26.1 *The function $\tau(k) \leq O(k^{.525})$ for symmetric Boolean functions with k inputs.*

This immediately gives new bounds on learning symmetric Boolean functions—see their paper for details. Their theorem does leave an interesting open problem: *What is the best bound on $\tau(k)$? Can it really be $O(1)$?*

26.3 Open Problems

Do you agree that there are many new exciting results being discovered in complexity theory? I think so, and I am optimistic that the trend will continue—that more and more problems will be solved or at least chipped away some.

26.4 Notes and Links

Original post:

<http://rjlipton.wordpress.com/2010/11/19/is-complexity-theory-on-the-brink/>

Klee’s measure problem:

http://en.wikipedia.org/wiki/Klee's_measure_problem

ABS paper:

<http://www.cs.princeton.edu/~dsteuerer/subexpug.pdf>

Ryan Williams’ new lower bound:

<http://www.cs.cmu.edu/~ryanw/acc-lbs.pdf>

Referenced post on Williams’ circuit lower bound:

<http://rjlipton.wordpress.com/2010/11/16/how-ryans-circuit-lower-bound-works/>

Paper on simplex algorithm under randomized pivoting:

http://www.daimi.au.dk/~tdh/papers/random_edge.pdf

Post by Gil Kalai on randomized pivoting:

<http://gilkalai.wordpress.com/2010/11/09/subexponential-lower-bound-for-randomized-pivot-rules/>

Notes on Klee–Minty theorem:

<http://glossary.computing.society.informs.org/notes/Klee-Minty.pdf>

Shpilka–Tal paper:

<http://www.eccc.uni-trier.de/report/2010/178/>

Joint paper on Junta Problem:

<http://research.microsoft.com/en-us/um/people/nvishno/webpapers/klmmvjunta.pdf>

Referenced post on Junta Problem:

<http://rjlipton.wordpress.com/2009/06/04/the-junta-problem>

Picture credits:

Klee and Minty polyhedron—screen capture from cited paper

<http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.24.1405>

George Dantzig is famous for the invention of the simplex method for solving linear programming problems.

We plan on presenting a definition and explanation of $P = NP$, but avoiding the usual technical details that seem to be needed to explain the problem. The approach will depend greatly on the famous linear programming problem and its generalizations.

Dantzig is a source of the “legend” that a student once solved an open problem that he thought was just a hard homework problem. I heard various versions over the years, but his story is apparently genuine. I quote his own [words](#):

During my first year at Berkeley I arrived late one day to one of Neyman’s (Jerzy Neyman) classes. On the blackboard were two problems which I assumed had been assigned for homework. I copied them down. A few days later I apologized to Neyman for taking so long to do the homework—the problems seemed to be a little harder to do than usual. I asked him if he still wanted the work. He told me to throw it on his desk. I did so reluctantly because his desk was covered with such a heap of papers that I feared my homework would be lost there forever.

About six weeks later, one Sunday morning about eight o’clock, Anne and I were awakened by someone banging on our front door. It was Neyman. He rushed in with papers in hand, all excited: “I’ve just written an introduction to one of your papers. Read it so I can send it out right away for publication.” For a minute I had no idea what he was talking about. To make a long story short, the problems on the blackboard which I had solved thinking they were homework were in fact two famous unsolved problems in statistics. That was the first inkling I had that there was anything special about them.

I do wonder sometimes if $P = NP$ will be solved this way. Not knowing that a problem is “impossible” must have some psychological advantage. Can we forget that $P = NP$ is hard and take a fresh look at the problem? I wish.

Let’s turn to the explanation. If you know what a linear equation is, then I believe you will be able to follow my explanation of what “P” is, what “NP” is, and what “ $P = NP$?” means. There will be no Turing machines mentioned anywhere—except here. This mention does not count. Even if you are an expert on $P = NP$, I hope you will like the approach I take here.

27.1 Linear Equations

In high school we all learned how to solve linear equations—at least we were “taught” how to solve such equations. No doubt given the equations

$$a + 2b = 5,$$

$$a - b = 2$$

we could all solve the linear equations and get that $a = 3$ and $b = 1$. One method is to try some values; guess values or go through values in some order. Another is to be more systematic and argue that $a - b = 2$ implies that

$$a = b + 2,$$

which implies

$$b + 2 + 2b = 5,$$

by substitution. This shows that $3b = 5 - 2$, and thus $b = 1$ and $a = 3$.

There is a systematic method of eliminating one variable at a time that was known to Carl Gauss over 200 years ago, and today we call his method **Gaussian Elimination** (GE). This algorithm (GE) is used everywhere, it is one of those fundamental algorithms that are central to computations of many kinds. If Gauss were alive today, between royalties on GE and the Fast Fourier Transform, he would be very wealthy.

One of the central questions we need to understand the $P = NP$ question is how to measure the efficiency of algorithms. One way to calculate the time GE requires is to count only the major operations of addition and multiplication that are performed. Let's say we have n equations on n variables. It is not hard to prove the cost of eliminating the first variable is about $(n - 1)n$. The next variable is a bit cheaper, but still costs at most $(n - 2)n$. The total cost is roughly

$$(n - 1)n + (n - 2)n + \dots$$

which adds up to order n^3 .

27.2 Linear Equations over Non-negative Numbers

In the above example of equations we did not say it explicitly, but we allowed rationals for the solution. The solutions for a and b turned out to be integers, but this is usually not the case—often rational numbers are required. For example,

$$a + 2b = 1,$$

$$a - b = 2,$$

now has the solution $a = 5/3$ and $b = -1/3$. Note, the solution uses not only rational numbers, but also uses negative numbers.

Many linear systems that arise in practice require that the solutions be non-negative. Suppose a company is trying to calculate how many trucks will be needed to haul their products, it is highly likely that the values of the variables must be non-negative. It does not make any sense to use -2 trucks. In general the problem of solving a linear system of equations with the added constraint that all variables are non-negative is called **Linear Programming** (LP).

There is a fundamental difference and a fundamental similarity between plain linear equations (LE) and LP. You probably did not learn in school how to solve such systems. Even with a few variables the task of solving such a problem can be a challenge.

The surprise is we know something the great Gauss did not know. We have a method for solving LP instances that is fast—almost as fast as GE—for solving linear equations. The result is quite beautiful, was discovered after many doubted it existed, and is used today to solve huge systems. Just as GE has changed the world and allows the solution of linear systems, the ability to solve LP has revolutionized much of science and technology. The method is called the *simplex* method, and was discovered by George Dantzig in 1947.

One measure of the importance of LP is that it is one of the few mathematical results whose inventors were awarded a Nobel Prize. In 1975 Tjalling Koopmans and Leonid Kantorovich received this great honor. You might note that Dantzig did not win, and the reason for that is another [story](#). Another measure is the dozens of implementations of the algorithm that are available on the Web—see [Wikipedia's list](#) in its “Linear programming” article.

The computational cost of the simplex algorithm is surprisingly not much more than GE's cost. In practice the running time is close to n^3 . The exact running time of the simplex algorithm is complex—it is a technical detail I would like to skip. But there are two points that I will mention about its running time.

- (1) The simplex algorithm is really a family of algorithms. The algorithms all move from vertex to vertex of a certain geometric object: usually there is a choice for the next step, and the running time is sensitive to this choice. For some choices it is known to be slow, and for other choices it works well in practice. Even more complex there are choices that are open whether they always run fast.
- (2) The simplex algorithm is not the only way to solve linear programs in practice or in theory. I will not get into a discussion of the algorithms that have provably fast running times.

27.3 Linear Equations over Natural Numbers

Linear systems can model even more interesting problems if we allow the solutions to only be natural numbers. Now the variables must take on values from

$$0, 1, 2, 3, \dots$$

The reason this is so useful is one of the great insights of complexity theory. It was discovered by Gödel, Cook, Karp, and Levin. These problems are called **Integer Programming** Problems (IP). They can express almost any practical or theoretical

problem. In many linear problems you can only allocate a whole number of something: try assigning $1/3$ of a person to a task or setting a value in a Boolean circuit to $1/2$. These are not possible. This is the reason IP problems rule, they can be used to solve essentially all problems.

We know more than Gauss, since we can solve LP's. But we are not that smart. We are stuck at IP's. We do not know how to solve these problems. The central point of the $P = NP$ question is, does there exist an algorithm for these problems like there does for rational linear systems?

Put another way, does there exist an algorithm for IP's that is *like* those for Gaussian Elimination and LP? This is the central question. A solution would have to take polynomial time—recall LE uses order n^3 time and LP uses order n^4 time. No one knows. The conventional wisdom is that there is no such algorithm, that IP's have crossed the threshold and cannot be efficiently solved. They have extended simple linear systems so far that there can be no efficient way to solve them.

This fundamental question goes by the name of the “ $P = NP$ ” question. The name is easy to say, “the P equal NP question,” and you can write it down quickly. Often we typeset it as

$$P \stackrel{?}{=} NP.$$

There are T-shirts for sale, in men's and women's sizes, that have this on their front. Alan Selman, a complexity theorist, has the license plate

$$P.NE.NP$$

Where I used to be a faculty member at Princeton, our Computer Science building has [encoded](#) in the bricks of the back of the building “ $P = NP?$ ” in ASCII code. A brick sticks out for a 1 and goes in for a 0. Clearly, “ $P = NP$ ” is an important problem, or are other conjectures encoded into buildings?

You might want to know why the problem is called “ $P = NP?$ ” and not the “Is there a fast way to solve integer programs?” problem. Okay, the latter would be an awful name, but why not something better?

In mathematics there are several ways that problems get named. Sometimes the first person who states the problem gets it named after them; sometimes it's the second person. Sometimes the problems are named in a very strange way.

27.4 Programming IP

We call LP, linear **programming**, and we call IP, integer **programming**. Both are real programming languages, even if they are different from standard programming languages.

A programming language allows one to express computational problems. The difference with LP and IP is that they are not universal languages, which means they cannot express all computations. They are not universal in the sense of Church—there are computations they cannot express.

Languages like C, C++, Perl, Python, and Java are universal languages. Anything can be expressed in them. This is both their advantage and their disadvantage. A universal language is great since it can do anything, but it also is subject to what Alan Perlis used to call the “Turing tar-pit.”

Beware of the Turing tar-pit in which everything is possible but nothing of interest is easy.

The dilemma is that if a language is more powerful, then many more problems are expressible in the language, but fewer properties can be efficiently checked. If a problem can be written as an LP problem, then we know we can determine efficiently if it has a solution. If, on the other hand, a problem can only be written as an IP problem, then we cannot in general determine if the problem has a solution. This is the dilemma: one trades power of expression for ease of solution. But both LP and IP are much more powerful than you might think.

LP can solve many important problems exactly, and even more problems can be solved approximately. IP can do just about anything that arises in practice. For example IP can be programmed to do all of the following:

- Break the AES encryption standard.
- Factor an integer, and thus break the RSA public-key system.
- Find a three-coloring of the edges of a graph, if one exists.
-

27.5 Programming Tricks

Just like any other programming language there are “tricks” in programming IP. To be a master IP programmer is not trivial, and I would like to present just a few simple tricks, and then show how to use them to solve 3-edge coloring on a cubic graph.

As an IP programmer one trick is to be able to use inequalities. This is really simple: if we want the inequality

$$x \leq y, \tag{27.1}$$

then we add a new variable z and replace the above by the equality

$$x + z = y. \tag{27.2}$$

Clearly, (27.1) is true if and only if (27.2) is true. A common use of this is to force variable x to lie in the set $\{0, 1\}$. This is done by simply writing $x \leq 1$. Since x is a non-negative integer it must be 0 or 1.

Here is another cool trick. Suppose we want to constrain x to be in a finite set, we can program this in IP. For example, to make x lie in the set $\{1, 2, 4\}$ we use the following IP program.

$$\begin{aligned} x &= a + 2b + 4c, \\ a + b &\leq 1, \end{aligned}$$

$$\begin{aligned}
 a + c &\leq 1, \\
 b + c &\leq 1, \\
 a + b + c &\geq 1,
 \end{aligned}$$

where a, b, c are all in the set $\{0, 1\}$.

Now suppose G is a cubic graph on the n vertices and m edges. Then, the IP program for 3-edge coloring is the following: Let x_1, \dots, x_m be the variables for the program. Add constraints so that each x_i is 1, 2, 4 valued. Then add the constraints for each vertex

$$x_i + x_j + x_k = 7,$$

where i, j, k are adjacent vertices.

Suppose there is a solution x_1, \dots, x_m to the program. I claim that this yields a coloring of the edges. Just color them according to their values. The only issue is can some vertex get two colors that are the same. Suppose the colors used are a, b, b . Now $a + 2b = 7$. Clearly, b cannot be 1 else the sum is too small, also it cannot be 4 else the sum is too large. So $b = 2$. Then $a = 3$, which is a contradiction.

27.6 Limits on the Power of IP

I have said that IP can handle just about any problem you can imagine. There are some limits on its power. For example, there is no way to add constraints so that variables x, y , and z satisfy

$$x \times y = z.$$

This is a curious result that seems “obvious,” but I do not know a simple proof. I would be interested in a simple proof, but for now let’s just say that there is no way to define multiplication.

27.7 Complexity Theory

For those who know some complexity theory, the approach I have taken to explain $P = NP$ may seem a bit strange. I hope even if you are well versed in complexity theory you will still enjoy the approach I have taken here.

In complexity theory the class of problems solved by LP is essentially all of P , and the class of problems solved by IP is all of NP . The advantage of the approach here is that $P = NP$ can be viewed as asking the question:

Can all IP problems be programmed as LP problems?

Note, to make this a precise question requires the notion of reduction. I will not define that precisely here, but imagine that changing an IP problem to an LP problem must be a reasonable type of transformation.

27.8 Conventional Wisdom

Most theorists seem to be sure that IP cannot be solved efficiently. One of the many reasons for this is the power of IP as a programming language. There is just too much power, which means that if it were easy to solve, then too many problems would be easy.

I wonder before GE was discovered would many have thought that LE's are hard to solve, or before simplex was discovered would many have thought that LP's were impossible to solve?

One approach that many have tried to resolve $P = NP$ is to encode IP problems into LP problems. The obvious difficulty is that a solution to a set of equations may have some of its variables set to proper rationals. The idea many have tried is to make the linear equations so "clever" that there is no way that solutions can be fractional. No one has yet shown a general method for doing this, which does not mean it is impossible, but it does mean it is hard.

27.9 Open Problems

Does this discussion help? Or is it better to present $P = NP$ in the normal way? What do you think?

Here is what we know,

$$LE \subseteq LP \subseteq IP.$$

Most believe that all these subset relationships are strict, but it is possible that they all are equal. This is one of the great mysteries of complexity theory.

27.10 Notes and Links

Original post:

<http://rjlipton.wordpress.com/2010/06/26/stating-pnp-without-turing-machines/>

Dantzig quotation:

http://www-groups.dcs.st-andrews.ac.uk/history/Biographies/Dantzig_George.html

Dantzig story:

<http://www.lionhrtpub.com/orms/orms-8-05/dantzig.html>

Linear programming list:

http://en.wikipedia.org/wiki/Linear_programming

Princeton Computer Science engraving:

<http://stuckinthebubble.blogspot.com/2009/07/three-interesting-points-on-princeton.html>

Source for Perlis quotation:

<http://www.cs.yale.edu/quotes.html>

Srinivasa Ramanujan was one of the most remarkable mathematicians of the last century. He discovered countless beautiful theorems in both analysis and number theory. With almost no formal training, he was able to discover results that other mathematicians found amazing, even magical. Sadly he died at the age of 32. Much has been made of his [notebooks](#), including the “[lost notebook](#),” but one wonders what further great things he could have discovered had he lived longer.

We will discuss the role that formal training plays in solving open problems in mathematics. When people talk about open problems, sometimes the $P = NP$ question is raised as one that might be solved by an amateur. Is this realistic or not?

I have read Keith Devlin’s [book](#) *The Millennium Problems: The Seven Greatest Unsolved Mathematical Puzzles of Our Time*—of course the count is now six thanks to the brilliant [work](#) of Grigori Perelman. I recommend it to you, with the proviso that I could only follow the chapters on the Riemann Hypothesis and the $P = NP$ question. The others on the Hodge Conjecture, the Yang–Mills existence and mass gap, the Navier–Stokes existence and smoothness, and the Birch and Swinnerton-Dyer conjecture, were very hard for me to follow. I do not think this is Devlin’s fault—he is an outstanding science writer—I think it is an inherent feature. Problems such as the Hodge Conjecture require, even to just understand their statement, quite a bit of technical background.

In his third chapter, on the $P = NP$ question, Devlin states that perhaps this problem is one that could be solved by an amateur. I have heard this claim before, and I have mixed feelings about it. Of course we cannot predict who will solve any open problem, and that includes the Millennium problems as well as all other open problems. My mixed feeling comes from saying $P \neq NP$ could be solved by an amateur seems to mean it is “easier” than the other problems. I do not think this is the case, but ranking the difficulty of open problems is probably impossible.

Let’s take a look at what amateur mathematicians have done in the past, and what they might be able to do in the future on these and other open problems.

28.1 Who Is an Amateur?

I think that one of the key issues is: who is an amateur? Is an amateur someone who is completely untrained in mathematics, or is an amateur someone who is not an expert in a particular area of mathematics? According to the dictionary, *amateur* is from the old French and means “lover of.” Thus amateurs often work on problems without any formal reward or pay. Curiously Perelman was essentially working for “free” when he solved the Poincaré Conjecture, but I think no one would consider him an amateur.

Was Ramanujan an amateur? He was basically self-taught, yet capable of discovering formulas that amazed even Godfrey Hardy. For example,

$$1 - 5\left(\frac{1}{2}\right)^3 + 9\left(\frac{1 \times 3}{2 \times 4}\right)^3 - 13\left(\frac{1 \times 3 \times 5}{2 \times 4 \times 6}\right)^3 + \dots = \frac{2}{\pi}$$

was one identity that greatly impressed Hardy.

Ramanujan shared one trait with many amateurs: his “proofs” were often non-standard and difficult to follow. This is perhaps one of the hardest issues for an amateur mathematician to overcome—the formally trained world of mathematics has certain rules and styles of how to present mathematics. An amateur may use nonstandard notation, may create new definitions for existing concepts, and may reprove basic facts. For example, Ramanujan’s first published paper was on some new properties of Bernoulli numbers. The journal editor Narayana Iyengar [noted](#):

Mr. Ramanujan’s methods were so terse and novel and his presentation so lacking in clearness and precision, that the ordinary [mathematical reader], unaccustomed to such intellectual gymnastics, could hardly follow him.

28.2 Some Results of Amateurs

Here is a partial [list](#) of some of the results obtained by amateur mathematicians. Some had no formal training, while others had training but worked on mathematics only as a “hobby.”

- **Thomas Bayes:** He was a minister by training, yet he introduced in 1764 the now famous formula that is named after him:

$$\Pr[A|B] = \Pr[B|A] \cdot \frac{\Pr[A]}{\Pr[B]}.$$

This formula is the basis of Bayesian analysis. Most of the credit—although not the name—of the consequences of this formula go to others. Quoting [Wikipedia](#):

Bayes was a minor figure in the history of science, who had little or no impact on the early development of statistics; it was the French mathematician Pierre-Simon Laplace who pioneered and popularized what is now called Bayesian probability.

This comment seems a bit harsh to me. I agree more with Bill Bryson's opinions about its foundational nature expressed in the opening pages of the book *Seeing Further*, which he produced and edited.

- **Alfred Kempe:** In 1879, while he was a barrister, Kempe discovered a “proof” of the Four-Color theorem for planar graphs. This stood until 1890 when Percy Heawood discovered Kempe's proof was flawed. Heawood did use Kempe's ideas to give a correct proof of the Five-Color theorem. In particular, he used Kempe's notion of chains of alternating colors. Such chains—now called Kempe chains—play a key role in both current proofs of the [Four-Color theorem](#).
- **Oliver Heaviside:** He invented in the 1880s the notion of operator calculus—a theory that can be used to solve linear differential equations. His methods work and give the correct answers. Correct answers or not, operator calculus was not well received by mathematicians. Yes it worked, but there were issues with some of the liberties he took that upset professional mathematicians. Heaviside could solve linear differential equations, yet the “theory” was not sound. Much later, Thomas Bromwich found a correct way to justify the operator calculus of Heaviside.
- **William Shanks:** In 1873 there were no computers, yet he was able to compute π to many more places than anyone had previously. He claimed it was correct to 707 places—it was correct to 527 places. It was still an incredible feat for his time—he did this [work](#) as a hobby when he was not running his boarding house.
- **Marjorie Rice:** She found new ways to tile the plane in pentagons—she did this as a hobby. Doris Schattschneider, a professional mathematician, helped make [Rice's results](#) known to the mathematical community.
- **Kurt Heegner:** He was a radio engineer and published in 1952 a claimed solution of one of the great then-open problems in algebraic number theory. As with many amateurs his proof was not accepted, due to mistakes in his paper. In 1969 the eminent number theorist Harold Stark solved the problem. Apparently, Stark went back to look once more at Heegner's paper, and he saw that it was essentially correct. The “errors” were minor and could easily be fixed.

28.3 Problem Statements

In order for anyone, amateur or not, to be able to solve an open problem they must understand the problem statement. This is almost silly to state, but it is important. It is hard to hit a fuzzy target. One of the reasons some think that $P \neq NP$ is more approachable than many other open problems is that an amateur is more likely to be able to state $P \neq NP$ correctly, than many other conjectures.

This is the reason so many amateurs have worked on the Four-Color Theorem, the $P = NP$ question, Fermat's Last Theorem, and Graph Isomorphism: they have relatively simple statements. A simple statement does not mean that a problem is easy, but it does mean that anyone can start thinking about it.

An interesting question is, which problems have simple statements?

- **$P = NP?$:** This does have a relatively simple statement. See the [discussion](#) in Chap. 27 featuring George Dantzig for my take on the problem statement.

- **Riemann Hypothesis:** Jeff Lagarias has a surprisingly “simple” [problem](#) that is equivalent to the Riemann Hypothesis. Let $H_n = \sum_{j=1}^n 1/j$. Then the problem is whether, for all $n \geq 1$,

$$\sum_{d|n} d \leq H_n + e^{H_n} \ln(H_n).$$

- **Hodge Conjecture:** I do not know how to even state this and the other Millennium problems in a simple way. No doubt this makes them unattractive to non-specialists.

28.4 Can Amateurs Help?

Can amateurs help make progress on modern problems? I think that it is possible. Their lack of fear, the ability of thinking out-of-the-box, the ability to think against the conventional wisdom, all make it possible for them to contribute to science. One of the reasons the pros usually get the credit is that amateurs often do not write their work up properly. They often write papers that have errors, and once we see an error we stop reading and skip the rest—Heegner is a perfect example.

It may be useful to look at the type of contributions amateurs have made in the past.

- **Definitions:** Bayes, Kempe, and partially Heaviside’s contribution was in defining new concepts.
- **Computations:** Shanks’ contribution was certainly a computation.
- **Examples:** Rice’s contribution was in the discovery of new examples.
- **Proofs:** Kempe, Heaviside, and Heegner’s contributions were in proofs.

This suggests there are several ways amateurs can help advance our understanding.

28.5 Open Problems

What do you think? Is $P \neq NP$ the problem most or least likely to be solved by an amateur? Can amateurs still make contributions to mathematics and complexity theory?

A related question: are there relatively simple statements of all the Millennium problems? Such a statement would not necessarily advance their solution, but I would find it interesting if they could be made more accessible. Even experts might be helped by short equivalent statements.

28.6 Notes and Links

Original post:

<http://rjlipton.wordpress.com/2010/07/01/can-amateurs-solve-pnp/>

Ramanujan's notebooks:

<http://www.imsc.res.in/~rao/ramanujan/notebookindex.htm>,

Ramanujan's lost notebook

http://en.wikipedia.org/wiki/Ramanujan's_lost_notebook

Devlin book:

<http://www.amazon.com/Millennium-Problems-Greatest-Unsolved-Mathematical/dp/0465017290>

Poincaré conjecture:

http://en.wikipedia.org/wiki/Poincare_conjecture

Ramanujan at Wikipedia:

http://en.wikipedia.org/wiki/Srinivasa_Ramanujan

Wikipedia's list of amateur mathematicians:

http://en.wikipedia.org/wiki/List_of_amateur_mathematicians

Bayesian probability:

http://en.wikipedia.org/wiki/Bayesian_probability

Bryson on Bayes:

<http://www.amazon.com/Seeing-Further-Science-Discovery-Society/dp/0061999776>

Four-Color Theorem:

http://en.wikipedia.org/wiki/Four_color_theorem

William Shanks:

http://en.wikipedia.org/wiki/William_Shanks

Results on tessellations:

http://en.wikipedia.org/wiki/Marjorie_Rice

Referenced post on P vs. NP statement:

<http://rjlipton.wordpress.com/2010/06/26/stating-pnp-without-turing-machine>

Lagarias' problem:

<http://www.math.lsa.umich.edu/~lagarias/doc/elementaryrh.pdf>

This post had a very active comment thread, with participation by amateur mathematicians.

John Rhodes is a world expert on semigroups and related mathematical objects. He is the “Rhodes” in the famous **Krohn–Rhodes Theorem**. Rhodes recently has worked on the $P = NP$ question, and even held a [conference](#) at Berkeley on his approach. It is based on ideas related to semigroup theory.

We will talk about methods mathematicians use to attack hard problems. Some of these methods are used often by theorists too, while others do not seem to be used as much as they could be.

One of the powerful tools that mathematicians use is related to the early work of Rhodes. But first I want to relate two stories about him. In the early 1960s Rhodes was a graduate student at MIT, and was working with another graduate student at Harvard, Kenneth Krohn. Together they discovered the theorem that became the Krohn–Rhodes Theorem. There is a long story here, but for now let me just say that they got PhD’s for the identical work in 1962. Identical. The theses were the same—a pretty rare event.

When I was at Berkeley in the late 1970s I was walking to lunch one day with Dick Karp. As we crossed the campus, Dick pointed out someone in the distance, and said, “Do you know who he is?” I answered that I did not recognize him. Dick added in a low voice,

That is Rhodes; he is in real estate.

I have never heard before or since Karp say something with such awe. I believe that Dick respected Rhodes as a mathematician, but he was filled with wonder at Rhodes the heavily leveraged real estate investor.

29.1 Approaches Used by Mathematics and Theory

There are a number of approaches to problems that are used by both mathematicians and computer theorists.

- **Restrict the Solutions:** In mathematics often if a problem is too difficult to solve, then progress can be made by restricting the solution space. Thus, rather than looking for the general solution to an equation, sometimes only linear solutions are considered. Another example is to consider only highly symmetric solutions.

In circuit complexity theory this is a common method too. Proving general lower bounds is so far a dream for any “natural” problem. Progress can be made, however, by restricting the class of circuits—Boolean or arithmetic. For instance we have lower bounds when the circuits are restricted to be:

- Monotone,
- Constant depth, or
- Non-commutative.

Even with these restrictions the results can be quite deep and difficult, but at least we can prove something.

For monotone, the “we” refers to Alexander Razborov, whose famous [paper](#) on the clique problem proved that it cannot be solved efficiently by any polynomial-size monotone circuit. This does not rule out that $P = NP$. It does give an important insight into the structure of this famous NP-complete problem.

Maurice Jansen and Ken Regan, in a [paper](#) on constant-depth arithmetic circuits, used a discrete uncertainty principle to prove their lower bound. Non-commutative circuits were studied in a [paper](#) of Pavel Hrubeš, Avi Wigderson, and Amir Yehudayoff.

- **Generalize the Problem:** In mathematics, generalizing a problem can often make it more tractable. This sounds paradoxical, but it is not. By generalizing the problem we are asking a different question, and that question may be quite a bit easier to solve. The hope, of course, is that the generalized problem will yield some insight into the original problem.

A simple example from mathematics is the famous $3x + 1$ problem, also called the [Collatz Conjecture](#) after Lothar Collatz. It has been open for over 70 years. Various generalizations have been studied and some of these solved, but the original problem stands open. Paul Erdős is quoted as saying:

Mathematics is not yet ready for such problems.

In computational complexity we have used this method too. One example is the use of [oracles](#). The famous results of Theodore Baker, John Gill, Robert Solovay showing that $P = NP$ can be true or false relative to different oracles, certainly give an important insight. As I have said before, I am not a [fan](#) of oracle results, but these results are important. Of course the real question is when we generalize a problem, how much does the generalization help us understand the original question.

29.2 Approaches Unique to Mathematics?

There are a number of approaches to problems that are used often by mathematicians. But they are not used by complexity theorists—at least I do not think as often as they should be.

- **Classification:** In mathematics a powerful method is to try to build a theory that classifies all objects of some type. The most famous such program is probably the [classification](#) of all finite simple groups. This tremendous result allows many problems to be solved that would otherwise be impossible. An example is the famous conjecture of the number of generators of a finite simple group.

Another classification type theorem is the Krohn–Rhodes [Theorem](#). The theorem proves that any finite automaton can be decomposed into “simple” automata. The hope at the time was that this decomposition theorem would help resolve many questions. I believe that this expectation at best was way too optimistic.

It is unclear if the classification method makes sense for complexity theory. There are many other classification theorems in mathematics that are very powerful, but there are many areas where such theorems seem to be hopeless. For example, I think trying to classify the models of set theory is completely impossible. Not just impossible—completely impossible.

However, it would be neat if there were useful classification theorems in complexity theory. What if we could prove a theorem like:

Theorem 29.1 *If the Boolean function $f(x)$ is computable in linear time by a Turing Machine, then $f(x)$ can be decomposed into...*

I have no idea what this would look like, but it is interesting to think about it. One issue, however, is that every computable function f has a “padded” version f' that is linear-time computable, and it is not clear what decomposable structure would apply to f' but not f .

There are some examples of classification theorems in complexity that have been quite useful. In the study of SAT there are the pretty results of Thomas Schaefer which are called his [Dichotomy Theorem](#). Roughly his theorem classifies the type of SAT problems that are easy and the types that are NP-complete.

Les Valiant and Jin-Yi Cai have [proved](#) some quite beautiful results on the boundary between counting problems that are either in P or are #P complete. Jin-Yi has several papers based on ideas of Valiant, but they extend that work greatly. The work of Cai is also joint with numerous others: including Xi Chen, Sangxia Huang, Michael Kowalczyk, Pinyan Lu, and Mingji Xia.

- **Make the Objects Really Big:** One tool to better understand a problem that mathematicians use and we rarely seem to is: replace “finite” by various notions of “infinite.” The study of finite groups is extremely difficult: one reason is that the interplay between small primes can be very complex and subtle. Studying infinite groups instead is not easy, but there are some cases where the theory does become much easier. For example, studying [Lie groups](#) is not easy, but it does avoid much of the “accidental” number issues that arise in finite groups. One way to see this is their classification occurred first and is simpler than the classification of finite simple groups.
- **Continuous:** Often replacing finite objects by continuous ones can yield a very powerful method. The entire field of analytic number theory is based on the ability of continuous functions to be used to solve discrete problems. Johann Dirich-

let's famous theorem on primes in arithmetic progressions was one of the first of these great results:

Theorem 29.2 *If $a > 0$ and $b > 0$ are relatively prime, then the sequence*

$$a, a + b, a + 2b, \dots, a + nb, \dots$$

contains an infinite number of primes.

The original proof of this used the structure of the following functions over the complex numbers s where χ is a Dirichlet character:

$$L(s, \chi) = \sum_{n=1}^{\infty} \frac{\chi(n)}{n^s}.$$

Dirichlet had to prove that they were not zero at $s = 1$.

In complexity theory we sometimes use continuous methods, but perhaps not as much as we could.

- **Model-Theoretic:** One of my favorite methods that has been used very successfully in mathematics is model-theoretic methods. My personal favorite is the [theorem](#) commonly named for James Ax and Alexander Grothendieck:

Theorem 29.3 *Let P be a polynomial map from \mathbb{C}^n to \mathbb{C}^n . If P is injective, then it is also surjective.*

The miracle of the proof is that the theorem is obvious for any map from a finite set to itself: if it is one-to-one, then it is onto. The great thing about model theory is that essentially the same argument works for the very non-finite set \mathbb{C}^n .

I wonder if we could use model-theoretic methods to solve some of our major problems.

29.3 Open Problems

Are there other methods that mathematics uses that we could try to exploit? Also any suggestions to add to the ones that I already have listed here?

29.4 Notes and Links

Original post:

<http://rjlipon.wordpress.com/2010/10/08/math-and-theory-proof-methods-a-comparison/>

Rhodes conference:

<http://math.berkeley.edu/~rhodes/JohnRhodes.pdf>

Razborov's paper:

<http://people.cs.uchicago.edu/~razborov/files/cliq.pdf>

Regan–Jansen paper:

<http://www.cse.buffalo.edu/~regan/papers/pdf/JaRe06-2rev.pdf>

Non-commutative paper:

<http://www.math.ias.edu/~pahrubes/data/Papers/Amir/NonComm6.pdf>

Collatz $3n + 1$ conjecture:

http://en.wikipedia.org/wiki/Collatz_conjecture

Oracles:

http://en.wikipedia.org/wiki/Oracle_machine

Referenced post on oracle machines:

<http://rjlipton.wordpress.com/2009/05/21/i-hate-oracle-results/>

Classification of finite simple groups:

http://en.wikipedia.org/wiki/Classification_of_finite_simple_groups

Krohn–Rhodes theory:

http://en.wikipedia.org/wiki/Krohn-Rhodes_theory

Schaefer's theorem:

http://en.wikipedia.org/wiki/Schaefer's_dichotomy_theorem

Cai–Valiant work:

<http://www.cis.upenn.edu/~mkearns/valiant/cai.pdf>

Lie groups:

http://en.wikipedia.org/wiki/Lie_group

Ax–Grothendieck theorem:

http://en.wikipedia.org/wiki/Ax-Grothendieck_theorem

John Nash is famous for his creation of what has become one of the central notions of modern game theory—his concept of what is a “solution” for a non-zero sum game. For this work he was awarded the Nobel Prize in Economic Sciences in 1994, along with John Harsanyi and Reinhard Selten.

We will talk about connections between problems in mathematics that are surprising.

Nash is famous for three things: his wonderful work on game theory, his wonderful embedding theorem, and his unfortunate illness that has made him unable to work for most of his adult life. His embedding theorem proves that every Riemannian manifold can be isometrically embedded into some Euclidean space. This is a deep result—way out of my area of expertise. But the core is that a Riemannian manifold is defined abstractly without assuming that it lies in any ambient Euclidean space. Nash’s theorem proved that such a manifold could always be embedded into a Euclidean space. His original proof was quite difficult. Later the proof was unraveled and it was shown that it depended on a new implicit function [theorem](#).

One wonders what other great results he could have proved had he not been ill for so long.

During his illness he was a well-known “figure” who spent much of his time at the main Princeton Library, Firestone. My wife, Judith Norback, has told me she often would see him, since she spent many days at the library. Her research required a great deal of “lit” review. I almost never saw him, since I spent my library time at the main mathematical library, which was located across campus. Ken does not recall meeting him, but knew his son via his undergraduate eating club for some months.

Roger Myerson has written a [paper](#) on Nash’s life and work. Myerson points out that Nash’s work on game theory spread slowly. It started with a short note in the Proceedings of the National Academy of Sciences, which was published in 1950. His concept of what is a solution to a non-zero sum game is elegant, but elegant or not it did not cause immediate excitement. It is useful to remember this in our own research: often it takes a long time for others to appreciate the importance of a

new result. Even the great John von Neumann did not acknowledge Nash's concept quickly: Myerson states this:

The short perfunctory reference to Nash's work in the preface to the 1953 edition of von Neumann and Morgenstern's book is particularly disappointing.

Either von Neumann and Oskar Morgenstern did not understand the importance of Nash's achievement, or . . . who knows.

30.1 Connections

In science, especially mathematics, it is reasonable to expect that different areas of research, even if they are “far” apart, may have interesting connections. Part of this is due to the beauty and power of mathematical ideas—there often is no precise demarcation between areas. Often a problem can be viewed, for example, as a question about a group, or a graph, or a set. Each view may yield different insights into a problem, and may eventually yield enough different insights to allow the problem to be solved; this fusion of ideas is often very powerful. We expect and welcome these connections between disparate areas, since they usually enrich both of them: ideas and methods from one can flow into the other.

These expectations aside, we are still sometimes taken aback by unexpected connections.

One of the most exciting events in mathematics is when an area that seems very far from another one helps solve a problem there. For a concrete example: if the $P \neq NP$ question is solved by a clever argument using complexity theory and some key new combinatorial lemma, we would all rejoice, but few would be shocked by the connection. If it is solved by the use of model theory and large cardinal axioms, or by the application of the theory of nonlinear partial differential equations, however, I believe we would be truly surprised.

It is these connections that are the most exciting and the most fun. They do happen. Here are two that come to mind, I will add one that is the main point of discussion in a moment.

- **Analytic Number Theory:** The [book](#) of this title by Donald Newman is a gem, in my opinion. I love how he starts the book:

The most intriguing thing about Analytic Number Theory (which means the use of Analysis, or function theory, in number theory) is its very existence! How could one use properties of continuous valued functions to determine properties of those most discrete items, the integers. Analytic functions? What has differentiability got to do with counting? The astonishment mounts further when we learn that the complex zeros of a certain analytic function are the basic tools in the investigation of the primes.

- **Model Theory and Analysis:** James Ax's solution of a problem in analysis using model theory is one of my favorite examples of connections that are unexpected. We [discussed](#) this in Chap. 29.

A recent [book review](#) in the Bulletin of the AMS is a great source of very interesting comments on the nature of connections. The review, by Joan Birman, is on braids, knots, and links. Birman is one of the world experts in this area, and has

written a beautiful summary of some of the unexpected connections that this area has made with other parts of mathematics. It comes strongly recommended.

30.2 The Connection

The connection I wish to discuss is between game theory and a crypto-type problem. The latter is the so-called *planted clique problem*. Lorenz Minder and Dan Vilenchik have proved a neat theorem that connects these two areas. I find the connection surprising, and one that could have important consequences.

In order to state their theorem I have to first explain what the planted clique problem is. It sounds a bit like a botanical problem, but is a natural question about the difficulty of solving the well-known clique problem. Of course we have known for a long time now that determining whether or not a graph has a clique of size k is NP-complete.

Dick Karp first asked, in 1976, whether or not one could find a clique of size $(1 + \varepsilon) \log n$ in a random graph $G_{n,1/2}$ in polynomial time. This is the usual random graph model where each edge is there independently with probability $1/2$. He noted that a simple greedy algorithm would yield size about $\log n$, so his question was essentially could one slightly improve the greedy algorithm. His question appears to still be open today—see the [thesis](#) of Benjamin Rossman for details and some other interesting results.

Lack of progress can often suggest that we modify our goals; hence the introduction of the planted clique problem. This problem is this: take a $G(n, 1/2)$ and add a clique of size k to the graph. The question is: can we find the added clique in polynomial time? If the size of the clique is very large, it is easy. The best known is that it is possible to solve if k is order \sqrt{n} . Any value of k that is smaller seems to be hard.

The connection of Minder and Vilenchik is to relate this problem to Nash solutions. In particular they [prove](#):

Theorem 30.1 *There exists a positive constant ε_0 so that if there is a polynomial-time algorithm that finds in a two-player game the ε -best ε -equilibrium, $0 < \varepsilon \leq \varepsilon_0$, then there is a probabilistic polynomial-time algorithm that distinguishes with high probability between two graphs: G in $G_{n,1/2}$ and H , an arbitrary graph on n nodes with a clique of size $(2 + 28\varepsilon^{1/8}) \log n$.*

Roughly, any good approximation algorithm for Nash equilibria, even approximate Nash equilibria, yields insights into the planted clique problem. As always take a look at their paper to see the details, and the proof. I find this connection pretty interesting—I hope you do too.

30.3 Open Problems

What are your favorite unexpected connections in mathematics?

30.4 Notes and Links

Original post:

<http://rjlipton.wordpress.com/2010/12/26/unexpected-connections-in-mathematics/>

Implicit function theorem touched on by Nash:

<http://maths-proceedings.anu.edu.au/040/CMAproc40-andrews.pdf>

Myerson paper:

<http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.130.7955>

Newman book:

<http://www.amazon.com/Analytic-Number-Theory-Graduate-Mathematics/dp/1475771657>

Referenced post comparing proof methods:

<http://rjlipton.wordpress.com/2010/10/08/math-and-theory-proof-methods-a-comparison/>

Book review:

<http://www.ams.org/journals/bull/2011-48-01/S0273-0979-2010-01305-7/>

Rossmann thesis:

<http://www.mit.edu/~brossman/mit-thesis.pdf>

Minder–Vilenchik paper:

<http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.150.3575>

Chee Yap is one of the top experts in various aspects of computational geometry. He has worked on problems as diverse as upper and lower bounds for comparison problems, to algorithms in geometric complexity, to methods to control rounding errors in geometric computations. One of his specialties is in the area of numerical computation from a complexity viewpoint. He was a PhD student of mine many years ago, so I may be biased in the following discussion.

We will talk about a recent paper of Yap on a question I raised in a 2009 [post](#). He has solved the problem, and I am quite happy to see that these discussions have helped raise some open problems that are getting solved.

The [LEDA](#) project was started by Kurt Mehlhorn years ago to implement geometric algorithms and data structures. It has grown into a major software system that is used by many in research, in education, and in companies.

Mehlhorn told me he almost immediately ran into a serious problem with their first implementation project, which was to implement an algorithm for finding the convex hull in two dimensions. This algorithm takes any set of n points in the plane and finds their bounding convex hull in time $O(n \log n)$. The algorithm is due to Ron Graham and is really a type of sorting algorithm. Mehlhorn's group tested the algorithm on some examples. It worked, and they moved on to more complex algorithms.

Shortly afterward, someone used their algorithm as a subroutine in a more complex computation—let's call it C . They quickly noticed that C did not always work; sometimes it gave the right answer, but sometimes it did not.

What was going on? His group looked into the problem and discovered an issue with their implementation of the convex hull: sometimes the “convex hull” returned was not convex. This is of course impossible—the hull by definition is convex. This meant that their implementation was incorrect. They quickly discovered the problem: certain examples caused the implementation of the algorithm to round incorrectly. Graham's algorithm was fine, but it assumed infinite precision. They redesigned the implementation and fixed the bug. Now C worked fine.

Mehlhorn said that this accuracy problem became a major issue for the whole LEDA project. They have to be sure that their implementations work given they

only have finite precision. They use a variety of tricks: sometimes they use high precision, sometimes they check the answers, sometime they change the algorithms.

Let's now turn to the problem Yap worked on, which is related to the difficulty that Mehlhorn faced in LEDA.

31.1 The BBP Algorithm

In 1995, David Bailey, Peter Borwein, and Simon Plouffe discovered a beautiful algorithm—now called BBP—that computes the n th bit of the number π without first computing any of the previous bits. From the viewpoint of complexity theory they showed that the problem of computing the n th bit of π is in SC. The latter is “Steve’s Class” named after Steve Cook and consists of problems computable in polynomial time and poly-logspace.

The BBP result was quite unexpected—see my 2009 [discussion](#) on their great algorithm (referenced in the end notes).

31.2 The BBP Algorithm Is Not an Algorithm

Their algorithm is wonderful. Or is it? The following remarkable quote is from [Wikipedia](#):

“There is a vanishingly small possibility that a particular computation will be akin to failing to add a small number (e.g. 1) to the number 9999999999999999 and that the error will propagate to the most significant digit, but being near this situation is obvious in the final value produced.”

Excuse me, this is *nonsense*. The statement “vanishingly small possibility” is a statement of probability theory. There is nothing random here at all. I agree that it seems that the problem may never arise in practice, but that is different. The last point of the quote is correct. But that means that the BBP is not really an algorithm.

For a theorist an algorithm must have two basic properties: it must always compute the correct answer and it must have a provable running time. I am leaving aside randomized algorithms, which have the above properties but with only probabilistic assurances. If your method works on “most” inputs or runs “fast” in practice, then it may be extremely useful, but it is not an algorithm in the strict sense.

This means that the BBP algorithm is not an algorithm in our strict sense.

31.3 Yap’s Theorem

Chee Yap has improved the BBP result in two fundamental ways. He shows that π can be computed in not just SC but in logspace. Further he shows that his algorithm really works—not that it just works in practice. Note, his algorithm is different from BBP’s algorithm, and it remains open if BBP’s exact algorithm works as claimed.

What is neat is how he used the notion of *irrationality measure* to prove his algorithm works, which is an idea I suggested in the 2009 discussion.

This is an inequality that bounds how closely the number can be approximated by rational numbers with small denominators. The rational numbers are dense, so given any irrational number α there is a rational number p/q so that

$$|\alpha - p/q|$$

is as small as desired. However, not all irrational numbers are created equal when it comes to how small this approximation can be. If

$$|\alpha - p/q| > 1/q^3$$

for all p and q , this means to get a close approximation to α one must have a large q .

The standard definition is that α has irrationality measure $\kappa > 0$ provided

$$|\alpha - p/q| \leq 1/q^\kappa$$

has only a finite number of solutions p and q . We would prefer to have the above hold for all p, q . But we usually cannot prove this for all, so we settle in the definition for almost all—that is for all q large enough.

Theorem 31.1 *Any number with a positive irrationality measure and a BPP series has an algorithm that computes its n th bit in polynomial time and space $O(\log n)$.*

See Yap's [paper](#) for the details.

31.4 Open Problems

What numbers can be computed in logspace? Can Yap's theorem be generalized?

This is a paper that solved an open problem from this blog, and I hope there will be many more in the future.

31.5 Notes and Links

Original post:

<http://rjlipton.wordpress.com/2010/07/14/making-an-algorithm-an-algorithm-bbp/>

Referenced post on algorithms for pi:

<http://rjlipton.wordpress.com/2009/03/15/cooks-class-contains-pi/>

LEDA:

http://www.mpi-inf.mpg.de/LEDA/information/what_is.html

Quote on BBP formula:

http://en.wikipedia.org/wiki/Bailey-Borwein-Plouffe_formula

Yap's paper:

<http://www.cs.nyu.edu/exact/doc/pi-log.pdf>

Henri Lebesgue is famous for the creation of modern measure theory. His definition of what we now call the [Lebesgue Integral](#) replaced previous notions of integrals. These were due to such great mathematicians as Isaac Newton, Gottfried Leibniz, Augustin-Louis Cauchy, and Bernhard Riemann. His definition was one of the landmark achievements of mathematics. His notion has many beautiful properties that all previous definitions lacked.

We will examine the role that projections play in mathematics. And further how tricky the seemingly simple concept of projection can be.

The main reason we are interested in discussing projections is to raise points of comparison between these two developments:

- (1) A classic result of Volker Strassen on lower bounds for arithmetic circuits. His result, even after 37 years, is the best known general lower bound on arithmetic computation.
- (2) A claimed result of Vinay Deolalikar on Turing Machine lower bounds. His claim, if true, would be the best result ever on Turing lower bounds.

Both attacks need to face and handle issues that arise from projections. Strassen does this in a clever way that we will describe shortly in some detail. It is a beautiful argument. Deolalikar may or may not know how to handle projections. We will see. But, this issue of projections in his “proof” has emerged as the main conceptual gap in his proof strategy for $P \neq NP$.

How hard can the notion of projection be? After all a projection is just the notion of a shadow—right? Well it can be very tricky. The great Lebesgue once made a “throw-away” remark about how sets behave under projections in a monograph. Here is a wonderful [summary](#) of the error from an AMS notice in 1931:

Lebesgue in his preface (to Lusin’s book) humorously points out that the origin of the problems considered by Lusin lies in an error made by Lebesgue himself in his 1905 memoir on functions representable analytically. Lebesgue stated there that the projection of a Borel set is always a Borel set. Lusin and his colleague Souslin constructed an example showing that this statement was false, thus discovering a new domain of point sets, a domain which includes as a proper part the domain of Borel sets. Lebesgue expresses his joy that he was inspired to commit such a fruitful error.

Let's now turn to discuss projections, and how Strassen handles them while perhaps Deolalikar does not. But that a great mathematician like Lebesgue can be mistaken about projections should be a reminder to us all.

Ken Regan wrote the rest of this article. Except for an earlier post on computer chess endgames, given later as Chap. 60, this was his first full post for the blog.

32.1 Let's Be Nice

Let us abbreviate Deolalikar's key concept of "polylog parameterizability" of a distribution by the English word "nice." Then the facts established by his paper, as summarized on a [wiki page](#) opened by Terence Tao, become:

- (1) If $P = NP$, then the solution space of any k -SAT problem with at least one solution supports a projection of a nice distribution.
- (2) The hard phase of k -SAT does not have a nice distribution.

Subsequent to a [comment](#) that Tao made while the review of Deolalikar's paper was progressing, the paper's public examiners agreed that these two facts do not, and conceptually cannot, combine to make a proof of $P \neq NP$. The extra clause "projection of" severs the link. In fact, the need for a projection cycles back upon the $P = NP$ problem itself, as we shortly explain. The question this raises for anyone working similarly on lower bounds is,

Do your concepts stay nice under projections?

We will see how Strassen managed with a concept that stays nice, but then explore whether any strategy for better lower bounds can do so.

32.2 Projections

Consider your shadow cast by a broad light against a back wall, which we will call the y, z plane. If the light is level with your head along the x -axis, then your silhouette on the wall is a *mathematical projection* that takes an especially simple form: it removes the x -coordinate. This is a natural physical concept, so you wouldn't think it typifies the abstract idea of nondeterminism by which NP is defined. But it does. The link is shown by noting:

A point (y, z) is in the projection if and only if **there exists** an x such that the point (x, y, z) is in the 3D image of your head.

Now the point (x, y, z) might be unique—it might be the tip of your nose. Or there might be many such "solution points," as with light (not) going through your skull. Either way, the "exists" makes it an NP matter.

To see why, first suppose y, z somehow code a Boolean formula ϕ in some number n of variables, and that x represents numbers from 0 to $2^n - 1$ that encode assignments to ϕ . Then the "3D image" is the set S of cases where $\phi(x) = \text{true}$. This is an easy set in P. The *projections*, however, form the language of formulas ϕ for which **there exists** an x making $\phi(x) = \text{true}$. This is the NP complete set SAT.

The analogy to Vinay Deolalikar's paper can be made even better by considering just z to encode a formula, and y to be a partial assignment to some of its variables. Then the projection becomes a language called EXT that was [described](#) as being computationally difficult during the review. Either way, the upshot is that in computational complexity we believe that a projection can make a set *more* complicated.

32.3 Shadows and Slices

The idea that your shadow can be more complex than you sounds laughable. If anything, features of your head get simplified. If you have a double chin, it's gone. Your hair may now mask your ears. If people enter the room, their shadows may combine with yours. But you will never, ever, see more people in a shadow than you have in the room.

The abstract mathematical fact is that the number of connected components of an image can never increase under projections. This is so even under general kinds of projections that may be onto an arbitrary subspace, even an *affine* subspace meaning one not through the origin, or may be *stereographic* meaning from a pinpoint not broad source. It seems that anything "physical" or "statistical" about a structure should become *simpler* when you take a projection. Or at least it should not get more complex.

Projections should not be confused with *slices*. A slice is what you get by fixing a variable, or substituting an affine linear expression for a variable. If you fix a formula ϕ , its solution space $S_\phi = \{x : \phi(x) = \text{true}\}$ is a slice of the global solution space S we defined above.

If you take a thin slice of a donut across the hole, you get two circular pieces. Thus it's not a surprise that the complexity can go up with slices. In fact, you might start with a formula ϕ that has many easy-to-find solutions, but begin with choices y_1, y_2, \dots for the "y" variables that make it very difficult to find a good answer with the "x" variables. You have thus created a difficult 1D slice for the EXT problem from the easy 2D slice of the SAT problem given by your fixed instance ϕ . It is interesting that Strassen's complexity measure described next does *not* go up under slices either.

Slices can be used as supports of distributions. You can even take a *section* which is an adjacent set of slices. Further you can induce distributions on subsets of the global solution space S that in a sense represent "phases." Thus slices are part of the conceptual environment in Deolalikar's paper. They are not, however, where the main problem is believed to lie.

32.4 Strassen's Complexity Measure

Recall we talked about arithmetical formulas last week in Chap 13. If you allow the $+$ and $*$ gates to connect to each other in a directed acyclic graph rather than just a tree, and allow multiple output gates, then you get an arithmetical *circuit* C .

Suppose C has input variables x_1, \dots, x_n and output gates y_1, \dots, y_s . As with formulas, each y_j is a polynomial $g_j(x)$ in (some or all of) the inputs x_i . Thus the circuit computes a polynomial *mapping* G , which is given by the tuple of polynomials $(g_1(x), g_2(x), \dots, g_s(x))$. If we think of y_1, \dots, y_s as variables standing for the values, you can represent the mapping by the set of equations (each set to zero)

$$G = \{y_1 - g_1(x), y_2 - g_2(x), \dots, y_s - g_s(x)\}.$$

We are now ready to define Strassen's measure. We have s equations in $n + s$ unknowns. We can get a well-specified system by adding n more equations that are affine-linear, allowing a constant term. The resulting system is well-specified if its solution space is a finite set F of single points in $(n + s)$ -dimensional space. What is the maximum size μ of such a finite set? That is Strassen's measure, $\mu = \mu(G)$. What he proved is:

Theorem 32.1 *Every arithmetic circuit C computing G must have size at least $\log_2(\mu(G))$.*

Walter Baur later helped him extend this to lower-bound the circuit size of a single polynomial $p(x)$, by taking G to be the set of first partial derivatives of p (so $s = n$). The key lemma needed was also covered in Chap. 13. A simple example is the polynomial $p(x) = x_1^n + \dots + x_n^n$. This gives

$$G_p = \{y_1 - nx_1^{n-1}, \dots, y_n - nx_n^{n-1}\}.$$

Make n more equations $y_i = n$ for each i . The resulting system becomes $x_i^{n-1} = 1$ for each i . Over the complex numbers each has $n - 1$ roots, and the equations are independent, so we get $(n - 1)^n$ points. Strassen's theorem then concludes that every arithmetical circuit that computes G_p must have at least $n \log_2(n - 1)$ gates. That yields an ironclad non-linear, $\Omega(n \log n)$ lower bound on computing p , allowing a constant of about 4 that comes from the Baur–Strassen lemma. Unfortunately no better lower bound is known for *any* reasonably explicit family of polynomials—but at least that's better than the situation in Boolean circuit theory, where no nonlinear lower bound for any function in E^{NP} is known at all.

32.5 How the Measure Plays Nice

It is immediate that Strassen's measure $\mu(G)$ is non-increasing under slices. A slice is the same as attaching a new affine linear equation. For example if you want to substitute $x_1 + 3x_2 - 7$ for x_3 , you can add the equation $x_3 - x_1 - 3x_2 + 7 = 0$. Since the definition of $\mu(G)$ takes a maximum over all choices of such equations to add, adding one can't give it a higher value.

Non-increasing under projections is a little trickier, but boils down to what we said above about the number of connected components. There is no way to get *more* single points by taking a projection of the ultimate set F than it already has. Now let's see how projections are *used* in Strassen's proof.

Consider a hypothetical circuit C computing the mapping G . Let us assign not only variables y_j to the output gates of C , but also variables z_k to the intermediate gates g_k . For every binary addition gate g_k getting inputs from gates g_i and g_j , possibly multiplied by constants c and d respectively, adjoin the equation

$$z_k - cz_i - dz_j = 0.$$

And if g_k is a multiplication gate, adjoin $z_k - z_i z_j$. Note that z_i and/or z_j might be an input variable x_i and/or x_j (importantly, $i = j$ is possible too), or a constant, and that g_k might be an output gate, making it have a "y-variable" y_k rather than a z_k . These added equations make a system G_C and define a solution space S' over a larger universe of variables that includes the z -variables. The key observation is:

The graph of the mapping G is obtained by projecting the z -variables out of the solution space S' of G_C .

Since every equation introduces a new variable, the full system G_C of equations has the essential properties of a mapping needed to well-define $\mu(G_C)$. By the non-increasing property under projections, $\mu(G) \leq \mu(G_C)$. Finally to upper-bound $\mu(G_C)$, consider what happens when we add gate equations one-at-a-time. Each addition gate gives a linear equation (affine if one input to the gate is a constant), and we already know from slices that this does not increase $\mu(G_C)$. Thus the entire lower bound follows from observing that adjoining the quadratic equation $z_k - z_i z_j = 0$ can *at most double* $\mu(G_C)$ as C is built up. This follows from a special case of *Bézout's Theorem*, for which an elementary argument can be found in Madhu Sudan's course notes, which are listed below in the end notes. The argument comes down to a polynomial of degree d in a single variable having at most d roots.

We have thus proved a lower bound on C by proving an upper bound on $\mu(G_C)$. Note that Strassen's lower bound actually applies to the number m of multiplication gates in the circuit alone. We proved $\mu(G_C) \leq 2^m$, thus $m \geq \log_2(\mu(G))$. Hence we learned more than we set out to prove, which is a hallmark of a good proof. Can we learn even more than this here? Alas perhaps not.

32.6 The Problem with Projections

The problem with the above observation about G being a projection of G_C is that it does not care that the circuit C is deterministic. Because it is deterministic, every input x and correct output y has a unique "solution" z that satisfies the expanded set of equations G_C . However, the concepts used do not care that these solution points are unique, and do not avail themselves of determinism in any other way. Thus again we learn more—we could frame a way that Strassen's technique gives

the same lower bound on some kind of nondeterministic arithmetic circuits. But this is more power than we want, applied to something other than what we want, which is to understand deterministic circuits.

The problem also shows up as a limitation on the measure μ itself. For a single polynomial p of degree d in n variables, the maximum possible $\mu(G_p)$ on the first partials of p is $(d-1)^n$. When d itself is $n^{O(1)}$, meaning that p comes from a family of *low-degree polynomials*, the log of this is $O(n \log n)$. This means that the best possible lower bound which Strassen's technique alone can prove for low-degree polynomials such as the permanent is $\Omega(n \log n)$, no more. This already is maxed out by the simple polynomial p given above, where it is asymptotically tight because x^n can be computed with $\sim \log_2 n$ multiplication gates via iterated squaring.

We speculate that these lacks may owe ultimately to the single fact that *Strassen's μ plays nice*. It is non-increasing under projections, as well as slices. Here we have actually concealed a little shock—the central concept of the theory that arguably underlies this work is not itself well defined under projections. Namely, put in your head the solution space of the equations $xy = 1$ and $xz = 1$. Now projecting out the x co-ordinate leaves the line $y = z$, but *missing* the origin $(0, 0)$. Missing this point prevents the projection from itself being the solution space of a set of polynomial equations. Superficially this resembles one of the concrete counterexamples on [Tao's wiki page](#) to preservation of Deolalikar's "pp" measure (while the other reminds us of the lower bounds on parity). Strassen's proof is able to escape, however, because the $xy = 1, xz = 1$ example is not the graph of a *mapping* (most to the point, its solution space is not an irreducible variety), while his tools work well for mappings.

Ironically this puts Deolalikar's "pp" complexity measure in the opposite situation from Strassen's μ . It does not stay nice under projections—per the counterexamples. If it did stay nice, then steps (1) and (2) at the beginning of this chapter would mesh, and presumably he would be able to complete a proof of $P \neq NP$ by fixing issues with each step separately.

32.7 Endgame?

It remains to ask whether it can be modified in a way that sidesteps the problem like Strassen's μ did. Here we can revisit our intuition about projections from the beginning:

Can a "statistical" property of distributions ever become more complex under projections?

If by "statistical" we mean purely numerical or physical properties encountered in everyday situations, the answer may seem to be "no," as surmised by one of my colleagues in computer vision. However, two others postulated an example where a high-entropy distribution in a 2D plane masks a low-entropy one in the 3D interior. More crudely, one can picture a complex 2D pattern simply being replicated along the axis of projection.

A “no” answer would be great if Deolalikar’s “pp” measure were solely statistical. However, the measure’s definition involves the existence of an acyclic graph on which the numerical properties are mounted. This graph is very like the circuits C we have been talking about. Hence this graph is very much like a computation. This parallel was demonstrated most aptly by the blog commenter named “vloodin” (whom we mentioned in Chap. 1), with a proof now linked from the wiki page that the needed graphs can arise directly from consideration of a polynomial-time (oblivious) Turing machine, without needing to go through the FO (LFP) characterization of P. The final rub may be that Deolalikar’s measure is more “computational” than “statistical,” so that the example of projections giving SAT or EXT from an easy set S becomes the ruling obstacle.

32.8 Open Problems

Can we hope to prove lower bounds with a complexity measure that is preserved under projections? Can we hope to prove lower bounds with a complexity measure that is not preserved under projections? What happens with slices?

Can the arithmetical $\Omega(n \log n)$ lower bounds be carried over to Boolean circuit complexity? The Baur–Strassen technique fails over finite fields, for example because over $\text{GF}(2)$, $x^2 = x$ while “ dx^2 ” = $2x = 0$.

Can results over infinite fields *lead* work over discrete domains at all? Is this like, or unlike, the history of the Prime Number Theorem?

32.9 Notes and Links

Original post:

<http://rjlipton.wordpress.com/2010/08/19/projections-can-be-tricky/>

Lebesgue integral:

http://en.wikipedia.org/wiki/Lebesgue_integral

AMS quotation:

<http://www.ams.org/journals/bull/1931-37-07/S0002-9904-1931-05171-5/S0002-9904-1931-05171-5.pdf>

Wiki page for Deolalikar’s proof claim:

http://michaelnielsen.org/polymath1/index.php?title=Polylog_parameterizability

Referenced post on EXT:

<http://rjlipton.wordpress.com/2010/08/15/the-p%E2%89%A0p-proof-is-one-week-old/>

Referenced post about arithmetical formulas and the claimed proof:

<http://rjlipton.wordpress.com/2010/08/10/update-on-deolalikars-proof-that-p%E2%89%A0p>

Referenced earlier post on Strassen’s work:

<http://rjlipton.wordpress.com/2010/03/27/fast-matrix-products-and-other-amazing-results/>

This was the capstone on the series of posts devoted to the real-time review of Vinay Deolalikar's claim to have proved $P \neq NP$. I (Ken) was especially happy to be able both to describe Strassen's degree measure and place it in a context relevant to Deolalikar's strategy, because I myself have tried to adapt Strassen's measure to my own attacks on problems related to $P \neq NP$. There was also some capstone discussion of the proof claim itself in the comments. Two further years have not shed much further light on the claim, although even speaking with fellow researchers at the 2012 FOCS conference in New Brunswick, some others said there are valuable ideas in this attempt that should be developed correctly and carefully.

Nina Balcan is one of the top young researchers in computational learning theory. She is not only a strong problem-solver, but has a vision of her field that is wonderful.

We will describe a breakthrough she made with Avrim Blum and Anupam Gupta. Their work has created an entirely new way of looking at problems, and I think it will be considered soon, if not already, as one of the great results of recent times.

Nina was hired in 2008, then she spent a year at the Microsoft Labs in Boston. We—at Georgia Tech—held our collective breath hoping she would leave there and come “back” to Tech. She did, much to our delight.

33.1 Classic Complexity Models

One of the main contributions of complexity theory is the formalization of what it means for an algorithm to be efficient. Prior to modern complexity theory, there were algorithms, but there was no way to understand their performance abstractly. Of course, one could use a stop-watch approach, and people did. Algorithms were timed on benchmark data—this is still done today. But, the ability to compare the performance of algorithms as a mathematical notion is, in my opinion, one of the great contributions of the field.

In the following let \mathcal{I} be a collection of problem instances. As usual n will be used to denote the size of the problem.

- **Worst-Case Model:** This is the classic gold standard model we all know and love. Given an algorithm we judge its performance by how well it does on the worst possible instance $I \in \mathcal{I}$ of size n .

The strength of the model is also its weakness: sometimes problem instances are not worst case. Sometimes, it “feels” that instances have some additional structure or some additional properties—but it is hard to make this “feeling” precise.

- **Average-Case Model:** This is now a classic model whose formal version is due to Leonid Levin—see this [survey](#) by Andrej Bogdanov and Luca Trevisan. The concept here is that additional structure is captured by assuming the problem instances come from a random distribution.

There are many pretty results in this area, but there seems to be something unrealistic about the model. How often are instances of a problem *really* selected according to a random distribution? This is the primary difficulty with this model of complexity—it is not easy to see why instances are random.

There are other models, but these are two of the main ones. In the next sections I will discuss the new model due to Nina and her colleagues.

33.2 A New Model

Nina’s new model is currently called the *BBG* model after the three authors of the seminal [paper](#): Balcan, Blum, and Gupta. I suggest a more descriptive name.

- **Shadow-Case Model:** This is the new model—the name is mine. The central idea is that in practice, problem instances are not selected by an all-powerful adversary, nor are they selected by the flip of a coin. BBG argue instances are selected to have a certain formal property that allows approximation algorithms to work. I think we need a better name, but for now I will stay with “the Shadow-Case model.”

I have always thought real-problem instances are neither worst-case nor random, but have never been able to find a good replacement. BBG seem to have succeeded in finding the right way to model many situations. At first it seems like their idea might be circular: a problem instance is “good,” provided our algorithms work well on it. This might be true, but it seems useless for building a sound theory. The **magic** is BBG have a notion of what a “good” instance is, they can build a sound mathematical theory, further the theory is beautiful, and even better it works in practice. Hard to beat.

33.3 Application to Clustering

Here is how they make the Shadow-Case model concrete for the classic problem of clustering. As always see their paper for the details and for the generalization to domains other than clustering.

They assume there is a metric space X with a distance function d . You are given n points from the space, and the task is to find a division of the points into k sets, *clusters*. They assume there is a *true* clustering into sets,

$$\mathbf{C} = C_1, \dots, C_k.$$

Finding this cluster is usually very hard, so we almost always settle for a clustering

$$\mathbf{C}' = C'_1, \dots, C'_k$$

that is “near” the true clustering. The notion of near is quite simple—just count the fraction of points that are in the wrong cluster. This induces a natural distance function on set systems. The goal is to find a clustering so the distance to the true one is at most some small ε .

So far nothing new, nothing different: this is all standard stuff. Of course the clustering problem as set up is impossible to solve. Not just hard. Impossible, since we have no prior knowledge of the true clustering.

The key to making all this work is the following. Let Φ be any measure on the quality of a particular clustering. Say the **Shadow Hypothesis** holds for Φ with parameters (α, ε) provided: for *any* clustering $\mathbf{C}' = C'_1, \dots, C'_k$ with

$$\Phi(\mathbf{C}') \leq (\alpha) \cdot \text{OPT}_\Phi,$$

then \mathbf{C}' is within ε of the true clustering \mathbf{C} .

There are several remarks I need to make. Note, the definition is not meant to be checkable, since it refers to the OPT_Φ which is almost always impossible to compute efficiently. Second, they use the assumption in a very clever way. They do not go and find a clustering that is close to the Φ optimum and then move it around somehow. No. They use the assumption purely in the background. The mere **existence** of the property is used to get an algorithm that works. This is the brilliance of their notion.

The reason I call it the Shadow-Case Model should now be clear. It is not that I do not like BBG, but I wanted to highlight that there is something lurking in the “shadows.” The assumption talks about an optimum they never find, and an approximation to that optimum they never find either.

Here is a kind of lemma they can prove using the Shadow Hypothesis.

Lemma 33.1 *Suppose a k -median clustering problem instance satisfies the Shadow Hypothesis with parameters (α, ε) . Then, at most $10\varepsilon n/\alpha$ points have a distance to their true cluster of more than*

$$\frac{\alpha w_{\text{avg}}}{10\varepsilon},$$

where w_{avg} is the average distance of each point to its true cluster.

Lemmas like this allow them to create simple and fast algorithms that get terrific clustering results. Again the measure Φ , the optimal clustering, and even the approximation to it, all remain in the shadows.

This is very neat work—see their paper for how all the magic works.

33.4 Open Problems

There are two obvious open problems. What is the right name for this notion? Is BBG the right one? If they use Shadow they can use, perhaps, “Shadow the Hedgehog” as their logo.

The most important open problem is to start using the model on other areas of theory. This is already well underway, but I expect a flood of papers using their brilliant idea.

Finally, the complexity theorist in me wants to know if there are ways to use their model to define formal complexity classes? Is this possible?

33.5 Notes and Links

Original post:

<http://rjlipton.wordpress.com/2010/05/17/the-shadow-case-model-of-complexity/>

Bogdanov–Trevisan survey:

<http://arxiv.org/pdf/cs/0606037>

BBG paper:

<http://portal.acm.org/citation.cfm?id=1496886>

Sam Buss is a logician who works in proof theory, and is one of the world's experts on bounded arithmetic. He has proved some quite neat theorems—I will discuss his work in more detail in the future, since proof theory is closely related to the $P = NP$ question.

We will talk about a much simpler problem from logic. I hit upon it while trying to prove a theorem. I failed in proving my theorem—so far at least—but I discovered an interesting limitation of first-order logic. This limit is not new, but I do not think it is as widely known as it should be.

I have always loved mathematical logic. I took undergraduate courses with two famous logicians, and was lucky to work with two other famous ones. Richard Vesley taught my first logic course; I took an advanced course, also as an undergraduate, from Helena Rasiowa. Later, as a graduate student I worked with Don Loveland. Finally, as a young assistant professor at Yale, I talked to Angus Macintyre a great deal about logic. He was kind enough to explain how the famous Paris–Harrington Theorem worked—among many other things.

Those I took courses from—Vesley and Rasiowa—each had their own unique styles of teaching:

- Vesley read from our textbook, which was Elliott Mendelson's famous [book](#) on logic. He would read a bit, and then we would discuss what he had just read. He often seemed to be confused, and we had to help him out. I think this was a great way to make us listen very carefully to his every word. I do not remember what I thought then, but my guess now is he knew the material very well. My guess is that his “confused” method was just his way to teach the material. A great course.
- Rasiowa lectured, without any notes, without a textbook, and without any motivation. Yet it was one of the best courses I have ever had. She was a visiting professor at Case, where I got my undergraduate degree, who taught a course on advanced topics in logic. She had a way of presenting the material that made it hard to ask questions: there was almost no motivation, just definitions, theorems, and proofs. One time we started a new topic on proof tableaux. We had no idea why we were starting to discuss this topic. None. One brave soul raised his hand,

she called on him, and he asked “Why are we studying this?” She answered with a stern voice,

You will understand.

That ended the discussion of motivation, and we moved back to definitions, theorems, and proofs. We did eventually understand, but not for many weeks. Let’s turn now to a problem in first-order logic.

34.1 A Problem with First-Order Logic

First-order logic is well known to be extremely powerful, yet also to have strong limitations. For example, first-order logic cannot in general define the transitive closure of a relation.

I know about these limits, but recently ran into another limitation that I was unaware of. The limit has to do with defining functions implicitly with first-order statements.

The problem I ran into was simple: Consider a statement of the form,

$$\forall x \forall y \exists a \exists b : S(x, y, a, b).$$

What this says is:

For each x and y , there are a and b , so that $S(x, y, a, b)$ is true.

For example, S could be the formula:

$$(a^2 = x) \wedge (b^2 = y).$$

The situation I ran into was that I wanted to express the following:

- (1) That a depends only on x .
- (2) That b depends only on y .

Note,

$$\forall x \exists a \forall y \exists b : S(x, y, a, b)$$

makes a only a function of x , but b is still a function of both x and y . Of course re-ordering variables to form

$$\forall y \exists b \forall x \exists a : S(x, y, a, b)$$

makes b only depend on y , but again a depends on both.

I tried for a while to express what I wanted in various ways by some more complex tricks. None worked. No matter what I did one of a and b wound up depending on both x and y .

I finally got to the point where I could actually prove that there is no way to do this. By then, I guessed that this was known. Perhaps, well known, but not well known to me.

Sam Buss to the rescue. I asked him about the issue, and he answered immediately: This is a well-known construction, known as a Henkin quantifier or **branching quantifier** (after Leon Henkin). It is usually denoted as a rectangular block:

$$\begin{array}{l} \forall x \exists a \\ \forall y \exists b \end{array} : S(x, y, a, b). \quad (34.1)$$

The meaning of (34.1) is just what I needed: the notation means that for each x and y there are a and b so that $S(x, y, a, b)$ is true. Further, the value of a only depends on x , and the value of b only depends on y .

Sam added: “Statement (34.1) is known **not** to be expressible in first-order logic, so you are correct that it cannot be encoded in standard first-order logic.”

If there had been a way to encode this I would have had a neat result. Oh well.

34.2 Open Problems

I still am interested in situations where you can get around this problem. I believe that for certain statements S it may be possible to define what I wanted in first-order logic. I conjecture that there should be some property that will allow the branching quantifier to be defined for such statements.

34.3 Notes and Links

Original post:

<http://rjlipton.wordpress.com/2010/01/17/a-limit-of-first-order-logic/>

Paris–Harrington theorem:

http://en.wikipedia.org/wiki/Paris-Harrington_theorem

Mendelson text:

<http://www.amazon.com/Introduction-Mathematical-Fourth-Elliott-Mendelson/dp/0412808307>

Branching quantifier:

<http://planetmath.org/node/33363>

Anton Klyachko is a mathematician who is an expert on group theory. He has worked extensively on equations over groups and related problems, and is on the faculty of the famous Mathematics and Mechanics Department at Moscow State University.

We will talk about why some theorems are considered more important than others, here jointly with Ken Regan.

What I will use to make this concrete is a result proved by Klyachko in 1992. The result is called the *Car Crash Theorem*. Now that is a neat name, but does not sound like a theorem that most would work on, or even think about. But I will argue it is really a quite important theorem.

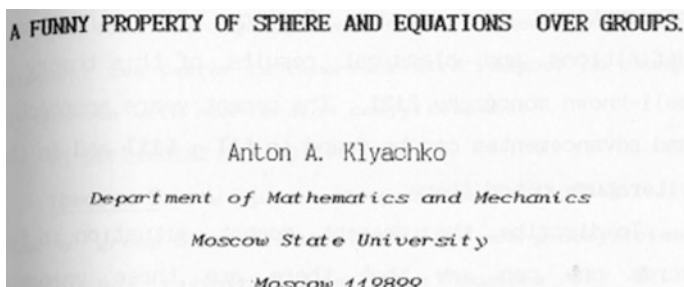
His paper starts in a way that might suggest that the result is just a fun result. Indeed the title of his paper is: “Funny Property of Sphere and Equations over Groups.” I love funny properties, but not all funny properties are important. His is.

The paper has the following unique beginning:

In this paper we present and prove a simple but non-obvious topological fact. This is so simple and funny that [it] can be included in a collection of puzzles or suggested as a problem for a school tournament.

This paper seems quite neat, but it has some issues as pointed out by Roger Fenn and Colin Rourke in their 1993 [paper](#). They graciously attribute the problems to language—Russian to English—and to the journal’s editors. Certainly Klyachko’s main idea is quite powerful—if you are interested in the full details see their paper.

I was unable to get an online pointer to Klyachko’s paper. Thanks to our library, yes the library, Farbod Shokrieh was able to get the actual paper. Here is the title part:



Note the curve on the left, that is from the book's shadow as it was being xeroxed. Do you remember when that was the only way to get journal articles?

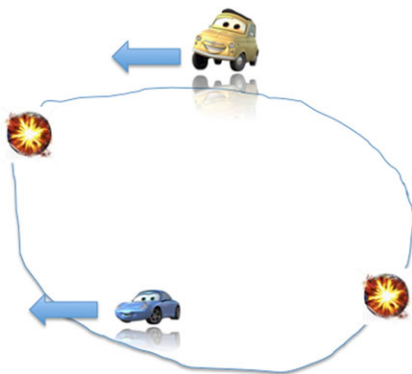
35.1 Car Crash Theorem

Let Γ be a finite connected graph on the 2-sphere—that is fancy terminology for a ball, as in a basketball or a soccer ball. Around the boundary of each region a car will drive according to these rules:

- (1) Each car stays on the same boundary forever, it never moves to another region.
- (2) The motion is continuous, cars cannot jump.
- (3) Each point of the graph is visited infinitely often, and there is a lower bound on the speed of the cars—they never stop nor slow to a “crawl.”
- (4) Each car moves with the same orientation.

The rules are really natural: cars move around their region over and over, they never get too slow, nor can they jump, and finally they all have the orientation.

Let's look, actually look, at the simplest case possible: when Γ has two regions. It would look like this:



Note the inner car and the outer car are really moving with the same orientation. Think of each car driver's side against the region, and then the car moves forward.

Klyachko's theorem is:

Theorem 35.1 *Let Γ be such a connected graph. If cars travel as described above there must be at least **two** distinct points where there are collisions.*

As a check it is easy to see that the cars will “crash” twice as they go around the cycle. Note, we assume that the crash is really not a “crash,” since they keep on moving. Perhaps it should be called a meeting, but I think the Car Meeting Theorem would not be as fun to state.

After I started to write this I discovered that the problem was previously [posted](#) to MathOverflow in October 2009 by Anton Geraschenko. He used ants instead of cars. I decided to stay with “cars,” since that is what is used in Klyachko’s paper. I also decided to continue with this discussion too. Please look at MathOverflow for additional comments.

35.2 A Proof Sketch

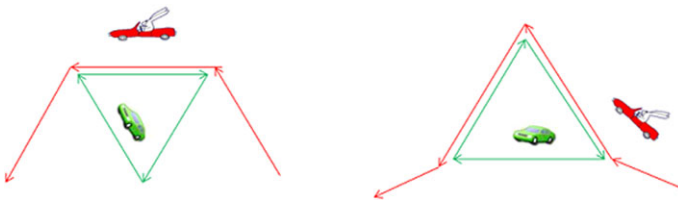
Ken Regan worked out a proof that reads differently from Fenn and Rourke, but uses the same basic ideas. Here it goes:

The idea of the proof is to reduce the general case to trivial cases like the one shown in the diagram above.

First, we can choose one region and flatten the sphere so that the region forms a boundary in the plane. Then the car on that boundary is seen from the opposite direction, so it travels counterclockwise. One source calls this the “drunken car” (or ant). The strategy is to contract the drive path of this guy until he causes a crash.

Next, we can help the proof by adding more traffic. Add a center point to every region and triangulate its interior. Now consider a car A running clockwise around the boundary. Just before it comes to an intersection, imagine a car A' that just beats it to the corner and turns in front of it. We can now make the original car A turn right and head to the center in the opposite direction that A' just traveled. Thus, with a gap that can be made arbitrarily small, car A' takes over the function of car A , while A itself disappears to the outside world. This pattern can be continued all the way around the boundary avoiding all crashes, even at the center. Thus the new triangulated graph has a crash-free itinerary if and only if the new graph has.

Now consider any edge being traversed by the drunken car. Either it is the outer edge of a triangle with two edges toward the middle, or part of two boundary edges of a triangle with one interior edge.



In the latter case, picture the interior edge being the base of a triangle with the two boundary edges pointing up. As the drunken car approaches the first boundary edge from the right, the normal car must be on the interior edge. It cannot be on either boundary edge, else the drunken car will crash into it before it finishes the triangle. Moreover, in a crash-free itinerary, the drunken car must complete the two

edges before the normal car finishes that edge. Once the drunken car has moved off that triangle, we don't care what the normal car does. Hence if the original itinerary is crash-free, then we can get a crash-free itinerary in the graph obtained by deleting the two boundary edges, and making the drunken car take over the no-longer-needed normal car on the interior edge. We have thus made progress in contracting the graph.

In the former case we have similar reasoning, but with an extra twist. As the drunken car approaches the boundary edge of the triangle, the normal car can be on either of the two interior edges. If it is on the interior edge away from the drunken car's entry, then it is still the case that it must stay on that edge until the drunken car is off the triangle. Hence in the graph obtained by removing the boundary edge, we may suppose the drunken car speeds up quickly to overtake and then take over the normal car. If it is on the interior edge incident to the drunken car's entry, it might not even reach the second edge before the drunken car has moved on. But this is the case where we don't care what the normal car does thereafter—so we again can just speed up the drunken car *after* it has overtaken the normal car. Again the transformation of removing exterior edge(s) from the triangle does not increase the number of crashes.

The process can continue until we get a single triangle (or the degenerate two-edge case above). Then the drunken car is driving counterclockwise along the outside of the triangle, while the last normal car is driving clockwise on the inside. Clearly they will crash, but we can even say more. If both cars do one full circuit, then they must crash *twice*. Although we spoke in terms of having a crash-free itinerary in proving our single-triangle reductions, the logic applies to say that the reduction does not increase the number of crashes. Hence the original graph must have at least 2 crashes, which completes the proof.

35.3 Another View

We can translate spherical or planar map instances into a constraint-satisfaction problem of building certain *interval graphs*. Let there be r “tracks” for the r regions in the map, including the outer region for the drunken car. Give each edge two labels for its two directions. Then we divide each track r into intervals corresponding to the labels of the edges in region r . The *constraints* are:

- As the intervals are formed going left-to-right, the label of the next interval must be the successor to the previous one.
- For each edge, its two labels form a forbidden pair.

A crash-free itinerary then yields an interval graph that meets these constraints. This graph has nodes for each interval on each track. Two nodes are adjacent if their intervals overlap vertically. The graph meets the second set of constraints if no adjacent nodes form a forbidden pair. If we extend the adjacency relation to successive nodes within each track, then we forbid all pairs other than the successor in the region's cycle.

Finally we require that track 1, for the drunken car, must be a full cycle. Then the theorem is equivalent to saying there is no way to complete intervals on the other tracks without violating one (or two) of the constraints.

35.4 Why Is It Important?

Why is this theorem important? Why are any **theorems** important? There are at least several reasons that come to mind:

- Some theorems are important because of their intrinsic nature. They may not have applications, but they just are beautiful. Or they have an interesting proof.
- Some theorems solve open problems. Of course any such theorem is automatically important, since the field has already decided that the question is interesting.
- Some theorems create whole new directions for mathematics and theory. These are sometimes—not always—relatively easy theorems to prove. But it may be very hard for us to realize that theorems like these should be formulated and proved. Their importance is that they show us that something new is possible.
- Some theorems are important because they introduce new proof techniques. Or contain a new lemma that is more useful than the theorem proved.
- Some theorems are important because of their “promise.” This is a subjective reason—a theorem may be important because people feel it could be even more important. Here, both the relation to group equations and the constraints-on-interval-graphs view make us feel that Klyachko’s Car Crash Theorem has some hidden possibilities.

35.5 The Application

Klyachko’s Car Crash Theorem is important because it partially solved a long-standing open problem. The problem concerns when equations over certain groups can be solved. An equation over a group G is:

$$a_1x^{e_1}a_2x^{e_2}\cdots a_mx^{e_m} = 1,$$

where x is the unknown and a_1, a_2, \dots are elements of the group G and the exponents e_i are integers. Say it has a solution provided there is a group \hat{G} that contains G as a subgroup and an element x in \hat{G} so that the equation is true. The nature of the equation is sensitive to the sum of the exponents. Consider

$$ax = xb,$$

which is the equation

$$b^{-1}x^{-1}ax = 1.$$

It has no solution if a and b have different finite orders. This follows since $b = x^{-1}ax$, which implies that a and b must have the same order. Note the sum of the exponents in this example equation is 0.

In particular Klyachko used his theorem to prove:

Theorem 35.2 *An equation over a group G without torsion is solvable if the exponent sum is 1.*

Recall that *torsion* means that there are elements in the group of finite order.

It might seem strange that his basic topological fact about graphs has anything to do with group equations, but it does. Very roughly: words over a group can be used to build a graph that naturally lies on the 2-sphere. See the [paper](#) for the details. Then the Car Crash Theorem is invoked to finish the argument.

35.6 Open Problems

Did you know Klyachko's Car Crash Theorem? I ran across it in my continuing attempt to understand solvable groups: I am still trying to prove or disprove whether or not a solvable group can be [universal](#). A yes answer would imply that NC^1 is in TC^0 .

35.7 Notes and Links

Original post:

<http://rjlipton.wordpress.com/2010/12/04/what-makes-a-theorem-important/>

Fenn–Rourke paper:

<http://citeseer.ist.psu.edu/viewdoc/summary?doi=10.1.1.45.3032>

MathOverflow post:

<http://mathoverflow.net/questions/2372/the-ants-on-a-ball-problem>

Referenced post on equations over groups—see chapter:

<http://rjlipton.wordpress.com/2010/11/03/equations-over-groups-a-mess/>

Picture credits:

1. Scan of Klyachko paper

Other figures are original, one made from clip-art bunny and car.

Bernard Chazelle is one of the world experts in computational geometry, lower bounds on data structures, upper bounds on data structures, and discrepancy theory—among many other aspects of theory. His [book](#) titled *The Discrepancy Method: Randomness and Complexity* is a classic. If you need to know about placing points so they are “well distributed,” you should read his book.

We discuss a presentation he gave at Georgia Tech, actually at the ARC Center, on “Natural Algorithms.”

I have had the pleasure of knowing Bernard since he was a graduate student at Yale, where his advisor was David Dobkin. After I got to Princeton in 1980, I quickly recruited David, who then quickly recruited Bernard. I cannot imagine Princeton without either of them.

I have one story I would like to relate about Bernard. It concerns the PhD examination process we used at Princeton years ago. One year I was in charge of the graduate examination process. We gave the students written exams, and then got the faculty together to discuss each student’s case. Our pass/fail decision was based on the exam and other information—for example, input from their advisor.

The written part of the exam was given over three days: one day was on systems, one on theory, and one on applications. There was a question on the systems exam that most of the theory students failed. This led to a pretty heated discussion about whether or not the question was fair. It was my question, so I thought it was okay, but many did not. Curiously, the systems faculty liked the question, but argued it was a theory question.

Bernard liked the question, thought it was fair, but still agreed it was a systems question—one theory students might not need to understand. Finally, cooler heads prevailed and we made decisions about exam grades all the faculty could support.

After the meeting Bernard was still convinced my question was not something a theory student should know. So I asked him to come to my office, where I pulled out an old paper on Turing Machines, and pointed to a lemma—there was my question. He finally agreed the question was definitely a theory one. I always thought the question was just a basic fact about data structures. Here is the question, you judge: theory or systems?

Using two stacks, show you can simulate a queue so the cost of n operations of the queue is bounded by $O(n)$.

What do you think? Theory or systems?

36.1 Natural

The word “natural” is a wonderful word, full of positive feelings—I think good thoughts when I hear it. This is why the word is used so often: there are natural resources, there are natural remedies, there is natural selection, there is natural history, there is the movie “The Natural,” and on and on. Not surprisingly, “natural” is also popular in theory and in mathematics. Here are a few uses of “natural” in our area of theory.

- **Natural Proof Systems.** Gerhard Gentzen created *natural deduction* in 1935. He created a set of inference rules for predicate calculus: rules with a simple clarity, rules with a certain naturalness. For example,

$$\frac{\frac{\Gamma \vdash A \quad A, B \vdash C}{\Gamma, B \vdash C}}{\Gamma \vdash B \rightarrow C}$$

is a simple natural deduction. You should read $\Gamma \vdash \phi$ as meaning: if the formulas in Γ are true, the formula ϕ follows. The brilliance of Gentzen was creating a logic consisting only of natural rules that was still complete: this uses his famous *cut-elimination method*.

- **Natural Mappings.** I have never used category theory in my research, and only know the very basics. With apologies to the experts, I view category theory as a way to study functions rather than objects as the primary concern. This shift in focus seems simple, but has remarkable consequences. For example, category theory is indispensable in explaining why some maps are “natural” and others are not. Category theory can do much more, but this is certainly one of the elementary but very powerful uses of it. Saunders Mac Lane once said:

“I didn’t invent categories to study functors; I invented them to study *natural transformations*.”

An important example comes from the theory of vector spaces. Consider, to be concrete, finite-dimensional vector spaces over the reals. Every such space has a dual space V^* of the same dimension: the dual is nothing more than all the linear functionals defined on the vector space. A linear functional f is just a linear map from V to \mathbb{R} .

Since any two such vector spaces are isomorphic if they have the same dimension, the following two statements are true for any V :

- (1) The space V is isomorphic to V^* .
- (2) The space V is isomorphic to V^{**} .

Yet, intuitively these statements are different: in (1) the map seems to depend on the basis used to define the spaces; in (2) the map is canonical or natural. The latter follows since there is a “canonical map” from V to V^{**} : Define M by

$$M(v)(\phi) = \phi(v),$$

where v is in the vector space V and ϕ is in the dual space V^* . This map M clearly does not depend on any basis of the vector spaces—it is natural. Category theory gives a precise way to define natural maps, and unravels why these two isomorphisms are different.

- **Natural Selection.** John Holland in the early 1970s created a notion of “[natural selection](#)” for the creation of algorithms. I will have to discuss this in more detail another time.
- **Natural Proofs.** In 1994 Alexander Razborov and Steven Rudich created the notion of [natural proofs](#). This is an important notion at the difficult frontier of attempts to resolve the P versus NP question.
- **Natural Algorithms.** This is an [area](#) started by Chazelle in the early 2000s and was the topic of his talk. NA is an attempt to apply complexity theory to the study of natural phenomena like bird flocking.

36.2 Form of Bernard's Talk

Bernard gave a beautiful talk on his recent work on NA. It was terrific on two levels: a great presentation and great results. The presentation was so special I thought I would discuss how he did it. It was unique.

His slides were very sparse. Some slides had a single phrase or a single picture or a single equation. Usually we see talks with slides full of bullets and multiple pieces of information. Bernard's was the opposite—a minimalist approach to the use of slide creation. It flowed easily from one idea to the next, and had a beautiful rhythm. I was very impressed with the form of his talk. If you can have him visit and give a talk, you will be very happy you did.

36.3 Content of the Talk

In Bernard's talk were some quite striking results and some intriguing open problems. One of the most surprising was an “old” result he has proved before on the theory of how birds flock. This is an intriguing question, with no compelling answer. They do flock, but we do not understand why they do, or how.

There are formal mathematical models, however, that let us try to study flocking. One of the models Bernard studies is a variant of the algebraic model of Felipe Cucker and Stephen Smale. All of the models he considers try to capture the following ideas:

- (1) Birds will try to avoid local crowding.

- (2) Birds tend to fly towards the average direction of their neighbors.
 (3) Birds tend to fly towards the average position of their neighbors.

What is surprising is Bernard proves an upper bound and a lower bound on how long it will take for the birds to achieve a steady state. The times are immense—both the upper and the lower bound.

Theorem 36.1

- A group of n birds reach steady state in at most $2 \uparrow \uparrow (4 \log n)$ time steps.
- There are initial configurations of n birds requiring more than $2 \uparrow \uparrow \log n / 2$ time steps before steady state.

The notation with the arrows is defined by: $2 \uparrow \uparrow 1 = 2$ and for $n > 1$,

$$2 \uparrow \uparrow n = 2^{2 \uparrow (n-1)}.$$

The proofs and the results are beautiful, but they suggest something is missing in the model, since the bounds are immense. Birds do actually flock in reasonable amounts of time, so there must a more realistic model. His results still are amazing.

36.4 Open Problems

One open problem is to find a model of flocking, for instance, where the bounds do not need exponential notation. This should be possible. The current work is beautiful, but birds do flock together faster than Bernard's lower bound. He and I spoke about this after his talk and he agrees it is one of the key open problems in the area.

36.5 Notes and Links

Original post:

<http://rjlipton.wordpress.com/2010/04/25/natural-natural-and-natural/>

Chazelle book:

<http://www.cs.princeton.edu/~chazelle/pubs/book.pdf>

Chazelle presentation:

<http://www.cs.princeton.edu/~chazelle/pubs/algorithm.html>

Natural deduction:

http://en.wikipedia.org/wiki/Natural_deduction

Cut elimination:

http://en.wikipedia.org/wiki/Cut-elimination_theorem

Natural transformations:

http://en.wikipedia.org/wiki/Natural_transformation

Natural selection of algorithms:

http://en.wikipedia.org/wiki/Genetic_algorithm

Natural proofs:

http://en.wikipedia.org/wiki/Natural_proof

Referenced post on natural proofs:

<http://rjlipton.wordpress.com/2009/03/25/whos-afraid-of-natural-proofs/>

Natural algorithms workshop:

<http://intractability.princeton.edu/blog/2009/09/natural-algorithms-workshop/>

Cucker–Smale paper:

<http://ttic.uchicago.edu/~smale/papers/math-of-emergence.pdf>

Thomas Jech is a set theorist and logician, who among many other things wrote a classic [book](#) on the [Axiom of Choice](#) (AC). I strongly recommend this book for its wonderfully lucid explanation of many aspects of AC—including the results by Kurt Gödel and Paul Cohen on the independence of AC from ZF set theory.

We will discuss an old result about the famous axiom. It concerns a finitary version of AC, and may be related to some of our problems in complexity theory. In any event it is a quite neat result, in my opinion.

At the 2010 Association of Symbolic Logic meeting, hosted by George Washington University in Washington, DC, a group of us went to dinner after our session on complexity theory. As usual we discussed many things during dinner: who was hiring, who was moving, $P = NP$, but somehow the conversation hit upon the AC. I told the group that I recalled vaguely a result on the AC for finite sets. No one seemed to be aware of such a result. I was a bit embarrassed, since I could not recall the exact statement of the theorem, but did recall it concerned choice on finite sets. We then moved onto other topics.

This is my attempt to make up for forgetting these pretty theorems about the AC for finite sets.

37.1 Finite Choice

Recall the AC is all about choice functions, since we are computer scientists we will only consider choices among finite objects. Consider a family \mathcal{F} of sets each of cardinality exactly n . The statement C_n says every such family has a choice function: More precisely, there is a function s from sets $A \in \mathcal{F}$ to elements so that

$$s(A) \in A$$

for every set A . Jech gives several theorems, but the most interesting ones solve the following problem: when does C_m imply C_n ? Here are two key examples:

Theorem 37.1 *The statement C_2 implies C_4 .*

Theorem 37.2 *The statement C_2 does not imply C_3 .*

What I find interesting is the non-monotone nature: the relationship from C_m to C_n is not simple.

Let me prove the first, C_2 implies C_4 . I will follow Jech's proof almost exactly. It is important to understand what we are allowed to do with sets:

- (1) Given a finite set A we can tell the cardinality of the set.
- (2) Given a set A and B we can form the set difference $A - B$.
- (3) Given any set $A = \{a, b\}$ of two elements, we have a **universal** function g that is a choice function:

$$g(A) = x,$$

where $x = a$ or $x = b$.

The last property follows from the assumption C_2 .

We now assume we have a family \mathcal{F} of four-element sets, and are looking for a way to select among any four elements. We will construct the choice function using the above properties—the existence of g is critical.

Let A be in the family \mathcal{F} . There are six two-element subsets of A . For every $a \in A$, let

$$q(a) = \text{Number of pairs } \{a, b\} \subset A \text{ such that } g(\{a, b\}) = a.$$

Obviously $q(a)$ cannot be the same for all $a \in A$: if all had $q(a) = 1$ this would be too few choices, if all had $q(a) = 2$ this would be too many choices. Let u be the least positive value of $q(a)$ over all $a \in A$. Then, define B as

$$B = \{a \mid q(a) = u\}.$$

Thus, B must have 1, 2 or 3 elements.

- (1) Suppose B has one element. Then use this element as the choice for the set A .
- (2) Suppose B has two elements. Then use g to choose among the two elements—and this is the choice for the set A .
- (3) Suppose B has three elements. Then use the unique $a \in A - B$ as the choice for the set A .

As arguments go this one is pretty neat.

37.2 The General Theorem

In order to define the general relationship, we need what Jech calls condition S : Say (m, n) satisfy condition S provided there is **no** decomposition of n into a sum of primes,

$$n = p_1 + \cdots + p_s,$$

such that $p_i > m$ for all $i = 1, \dots, s$.

Theorem 37.3 *If (m, n) satisfy condition S , and if C_k holds for every $k \leq m$, then C_n holds.*

See Jech's book for the details of the proof. The proof is an induction argument and uses some simple properties of binomials. Note first that $(2, 4)$ does satisfy property S .

He also shows that the condition S can be used to completely characterize when C_m implies C_n . The technically harder direction is to show that a certain C_m does not imply a certain C_n , since this requires set theory independence machinery. If you are interested please check his book for the details. It allows him to prove:

Theorem 37.4 *If (m, n) does not satisfy condition S , then there is a model of set theory in which C_k holds for every $k \leq m$ but C_n fails.*

37.3 Open Problems

Can we use this finite AC in any part of complexity theory? Does the "trick" of proving C_2 implies C_4 have any application in any part of theory?

37.4 Notes and Links

Original post:

<http://rjlipton.wordpress.com/2010/04/12/socks-shoes-and-the-axiom-of-choice/>

Book by Jech:

http://www.amazon.com/Axiom-Choice-Thomas-J-Jech/dp/0486466248/ref=pd_sim_b_3

Axiom of Choice:

http://en.wikipedia.org/wiki/Axiom_of_choice

Alfonso Bedoya was not a theoretician, but an actor. He is perhaps most famous for uttering the lines:

“Badges? We ain’t got no badges. We don’t need no badges. I don’t have to show you any stinking badges!”

He played Gold Hat, in the movie “The Treasure of the Sierra Madre.”

We must talk about definitions. Mathematics consists of definitions, theorems, and proofs.

Instead of badges we are going to talk about definitions. To paraphrase Bedoya:

Definitions, definitions, we don’t need no definitions.

38.1 Definitions

A major part of mathematics, of all kinds, is played by definitions. Every area of math has many definitions that must be learned if one is to be able to understand that area. Here are some key points, in my opinion, about definitions in general.

Definitions can be eliminated A definition can always be eliminated from a proof. Essentially, a definition is a shorthand; a definition allows us to make arguments clearer and shorter. For example, the statement that there are an infinite number of primes

$$\forall x \exists p > x, \quad p \text{ is a prime}$$

could be written without using the definition “prime.” But, the statement would be longer and less readable:

$$\forall x \exists p > x, \quad p > 1 \wedge (\forall y \forall z \ p = y \cdot z \rightarrow y = 1 \vee z = 1).$$

This is an important role that definitions play: they help reduce the size of proofs. There is some nice work in proof theory on the role of definitions in reducing proof size.

Definitions can be “silly” Any definition is allowed; however, some are much more useful and productive than others. Consider the definition: p is a *special* prime if and only if p , $p + 2$, and $p + 4$ are all prime. Is this a good definition? No. Because p is special if and only if $p = 3$. We will see other, less trivial, examples of “silly” definitions in a moment.

Definitions can be “critical” Definitions shape a field; they affect the way we think, and what theorems we even try to prove. A good definition can suggest new research, can open up new directions, and can have a profound effect on progress.

Definitions need justification While any definition is “valid,” some are more useful than others. Often we read something into a definition that is not justified. So a good question to ask about any definition is: Does it actually capture the property that we are trying to “define”?

38.2 Mathematical Definitions

I will continue the discussion of the last point about the utility and acuity of definitions by giving a series of examples.

- **What is a function?** This is one of the hardest definitions to get “right.” Early mathematicians argued over what was the right definition: some insisted that any “rule” was fine, others that a function had to have an analytic form, others that it had to be given by a series of some kind.
- **What is a set?** Georg Cantor gave a beautiful definition:

A set is a Many that allows itself to be thought of as a One.

The trouble with his definition is that it is not precise enough. Unfortunately, there are many paradoxes possible with a naive definition of set. The most famous perhaps is due to Bertrand Russell: consider the set

$$S = \{x \mid x \notin x\}.$$

Is S a member of S ? Either answer yields a contradiction.

- **What is a “nice” function?** Define a continuous function f on the interval $[0, 1]$ to be K -Lipschitz provided

$$|f(x) - f(y)| \leq K \cdot |x - y|.$$

Similarly, continuous functions that satisfy the below property are said to be Lipschitz of order α , or Hölder continuous functions,

$$|f(x) - f(y)| \leq K \cdot |x - y|^\alpha.$$

These conditions are named after Rudolf Lipschitz.

There is a story: a student is giving his thesis defense at Princeton University. He has studied the class of Lipschitz functions where $\alpha > 1$ —why not study these too—he reasons.

His thesis has many neat results, and he has proved that these functions are closed under many interesting operations. The story continues that a visiting professor raises his hand, during the talk, and asks:

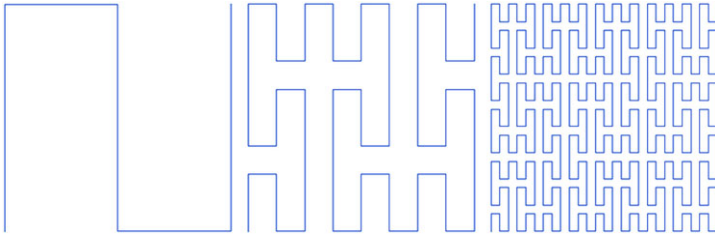
Can you give me an example of any function in this class that is not constant?

The sad answer is no. End of talk, end of thesis. Hopefully, only an urban legend.

- **What is a curve?** An obvious definition of a curve is a mapping from $[0, 1]$ to $[0, 1]^2$ defined by

$$(x(t), y(t)),$$

where $x(t)$ and $y(t)$ are continuous functions. This seems to be well-defined and reasonable. The problem is that such “curves” have a non-intuitive property: a curve can fill the entire square. Giuseppe Peano discovered that a [space-filling curve](#) was possible.



This discovery showed that the definition was not “correct.” In order to avoid space-filling curves one needs to add more to the definition. For example, if the maps $x(t)$ and $y(t)$ are smooth, then there is no space-filling curve possible.

38.3 Computational Definitions

Let’s consider two definitions from computer science: sorting and security.

- **What is a sorting algorithm?** Suppose that someone sells you a sorting algorithm, one that takes groups of records and sorts them according to any key that you like. What does it mean to be a *correct sorting algorithm*? I once asked someone this question—their answer was:
 - (1) The algorithm’s output must be a permutation of the input records. The algorithm cannot change the records in any manner at all.
 - (2) The algorithm’s output must have the records in the correct order based on the given key. If the keys of the output records are

$$k_1, k_2, \dots, k_n,$$

then

$$k_1 \leq k_2 \leq \dots \leq k_n.$$

Is this the right definition of a sorting algorithm? Would you buy this algorithm from me? Or is this definition “wrong”?

The definition is not precise enough; if you used this algorithm to do many tasks that arise, there would be a problem. The issue is that the definition does not insist that the sorting algorithm be **stable**. A sorting algorithm is stable if it retains the order of the records that have the same key.

For example, quicksort is not stable, but mergesort is stable.

Consider three records

$\langle \textit{Alice}, \textit{Smith}, 67890 \rangle$

$\langle \textit{Bob}, \textit{Jones}, 12345 \rangle$

$\langle \textit{Alice}, \textit{Jones}, 12900 \rangle$

Suppose we sort the records on the second key: the last names.

$\langle \textit{Bob}, \textit{Jones}, 12345 \rangle$

$\langle \textit{Alice}, \textit{Jones}, 12900 \rangle$

$\langle \textit{Alice}, \textit{Smith}, 67890 \rangle$

Then, we sort on first names. This could be the outcome:

$\langle \textit{Alice}, \textit{Smith}, 67890 \rangle$

$\langle \textit{Alice}, \textit{Jones}, 12900 \rangle$

$\langle \textit{Bob}, \textit{Jones}, 12345 \rangle$

This is not what we wanted: the records are *not* sorted according to last names and then first names. A stable sort would have yielded the correct order:

$\langle \textit{Alice}, \textit{Jones}, 12900 \rangle$

$\langle \textit{Alice}, \textit{Smith}, 67890 \rangle$

$\langle \textit{Bob}, \textit{Jones}, 12345 \rangle$

- **What is a secure protocol?** Modern cryptography has stated many definitions that attempted to capture some notion of security. Then, later on, the notion was discovered to be deficient: usually the definition sounded right, but lacked the exact property that was wanted. So a new definition was created and the process continues . . .

38.4 Justification of Definitions

If definitions can be “wrong,” how do we know that they are “right?” The best methods that I know are based on two kinds of evidence.

- **Equivalence Approach:** One method of getting some confidence that a definition is right is to show that it has many different equivalent definitions. This is done throughout mathematics. For example, there are many very different—but equivalent—definitions of a continuous function. Some use ε and δ 's, others use topology, and others use. . . In computing there are many equivalent definitions of the complexity class NP, for example.

The fact that very different definitions yield the same concept is good evidence that the definition is the right one.

- **Consequence Approach:** Another method for increasing confidence that a definition is correct is to examine its *consequences*. For example, the fact that the original definition of a curve could fill a square was not expected. One could argue that the “right” definition of a curve would not have this property.

Another example is the definition of continuous functions. The fact that they are closed under certain basic operations gives one a good feeling that the definition is right. If our definition of some property is *not* closed under reasonable operations—operations we feel it should be closed under—then the definition probably is not capturing our intuition.

38.5 Open Problems

Any definition is valid. Some definitions are interesting, some are silly, some definitions are useful. But, getting the right definition is not easy. Often it is quite hard to get the definition that captures the property that you are after.

The open question you might think about is: are the definitions you work with the right ones? How do you know that they make sense? I would argue that more thought should go into definition *justification*. Often—especially in computing—we are given a definition without any supporting arguments.

38.6 Notes and Links

Original post:

<http://rjlipon.wordpress.com/2010/01/23/definitions-definitions-do-we-need-them/>

Film quote on badges:

<http://www.rudebadmood.com/badges/index.shtml>

Peano curve:

http://en.wikipedia.org/wiki/Space-filling_curve

Stable sorting:

http://en.wikipedia.org/wiki/Sorting_algorithm#Stability

Hartley Rogers is a mathematician at MIT who wrote arguably the definitive book on basic and advanced aspects of recursion theory. His [book](#) entitled *Theory of Recursive Functions and Effective Computability* is a classic. He also had many strong PhD students over the years, including several who are well-known computer scientists: Patrick Fischer, David Luckham, and Paul Young.

We address the foundations of complexity theory. Several commentators have asked more than once, “What is a complexity class?” I will try to answer that and a bit more.

I learned recursion theory from Rogers’ book. I learned early on to beware of exercises marked with the scary ▲ symbol. These were very hard problems. There may be more recent treatments of recursion theory, but to me Rogers’ book will always be one of the great textbooks on any subject.

A recent New York Times front page article on live vs. distance learning states two main points. More and more schools are using online technology. It is driven by lack of funds, and the difference between live and online learning is being found to be lessening. While I never was in a live class with Rogers, I wish I had been. There are many stories of how he taught, but my favorite is that he rewarded top students with [Leibniz](#) cookies.



I guess that will be always hard to do in an online environment—perhaps one day it will be possible to have a cookie roll out of your laptop. I do support existing ways that are trying to make better use of technology to teach, but getting a chocolate cookie seems hard to beat.

Let’s move on to the foundations of complexity theory, and I will connect back to Rogers later.

39.1 The Big Three Ideas

Complexity theory is based on three ideas, they are ideas that are so built into our “DNA” as theorists that I think we sometimes forget how important they are. There are other key ideas, but these three come first, in my opinion.

- **Boolean strings:** I got myself in trouble in 2009 by proclaiming too sweepingly that “the world is digital,” but I will try my point again while keeping it to the technical side. The starting insight is that we can use Boolean strings to represent all our basic objects. From integers to matrices, from permutations to groups, we can and do use strings of 0’s and 1’s to represent them. Strings are the basic building blocks that we use every day in complexity theory. We say all the time,

Let x be a natural number . . .

It is implicitly assumed that everyone will convert that to

Let x be a string $x_1x_2 \dots x_m$ of 0’s and 1’s that encodes the number in the usual way.

I believe the first to talk at length about binary numbers was the great Gottfried Leibniz. He was not the first to discover the binary system, but he did write about it in his [Explication de l’Arithmétique Binaire](#) in 1703.

- **Problems:** The definition of a **problem** is simple: a problem is nothing more than a set of strings. That is it. Problems often consist of Boolean strings, but we can allow arbitrary finite alphabets, since any alphabet can be encoded into a Boolean one.

This is one of the simplest definitions in mathematics, yet it is really quite important. The notion of a problem as a formal mathematical object allows us to study problems, without this definition a problem would be a vague notion. I think that one of the great achievements of theory is the formalization of the “problem” notion.

Before this definition people might have studied a problem like: “How do we assign tasks to machines?” or “How do we find the shortest path in a graph?” But they were not precise mathematical objects. So even though the definition is almost trivial, it is very important.

In the early days of theory problems were usually called **languages**. A language L was just a set of strings:

$$L \subseteq \Sigma^*.$$

The reason they were called languages initially is based on the early motivations of the field. The initial questions were driven by two types of languages: natural languages and artificial ones. Certainly Noam Chomsky and many others were interested in trying to model [real languages](#), such as English. Others were driven by their interest in computer programming languages. The questions raised in early theory reflected these interests, as did the results. For example, one beautiful idea is called [BNF](#)—and stands for Backus–Naur Form. John Backus was interested in ways to describe programming languages like Algol, which lead him to create BNF. Later he used related tools in his important work on FORTRAN; for this and other seminal work he received the Turing Award in 1977.

Properly a language denotes a *decision problem*. There are also function problems, such as, given a whole number N , find its prime factors. But these can be converted to decision problems over strings, such as the set $F = \{N\#w : w \text{ is less than some prime factor of } N\}$. A fast way to solve F yields a fast way to factor numbers.

- **Classes:** A family of problems is called a **complexity class**. Thus, C is a complexity class provided it consists of problems. It is that general, that simple, and that elegant. Any such family is a complexity class. However, as in the famous book *Animal Farm* by George Orwell:

All complexity classes are equal, but some complexity classes are more equal than others.

Our friends at Wikipedia give a much more restrictive [definition](#) of a complexity class:

In computational complexity theory, a complexity class is a set of problems of related resource-based complexity.

I think the above definition is fine for helping pin down the notion of *complexity*, but for *class* it is too restrictive in two ways. Not all interesting classes are related by a known computational resource. The notion of what are important resources itself keeps changing and growing. In the beginning there was just time and space, then we added nondeterminism, then probabilities of various sorts, then interaction, streamability, and so on. Perhaps energy is next.

Second, the notion of “related” better applies to the idea of a complexity *level*. The NP-complete problems are all closely related and form a level as well as a complexity class, but EXPTIME is a complexity class with levels that are quite unrelated. Of course, relatedness depends on the level of fine detail—compared to primitive-recursive all of EXPTIME is relatively easy—but saying “related” misses the larger point, I think. The notion of class identifies problems that share important characteristics, which may not be resource-bound and may not be “equal” even in Orwell’s sense.

I like having a general view of what a complexity class is, to focus on what makes problems *interesting*. In any event, not all possible complexity classes are interesting, and we will soon see some things that make a class, well, “classy.”

39.2 How Many Classes Are There?

The answer to this question depends on what you mean. The cardinality of strings is countable \aleph_0 , the cardinality of problems is 2^{\aleph_0} , and the cardinality of complexity classes is $2^{2^{\aleph_0}}$.

Of course most of these classes are uninteresting. But I thought it would be fun to point out the sizes involved. If nothing, it is clear there is a hierarchy here: there are many more complexity classes than problems, and many more problems than

39.4 Properties of Complexity Classes

One of the properties that makes a complexity class special is having a complete problem. Suppose that C is a class, if there is a single problem S in the class that captures the entire class this is very neat. That a complexity class can be represented by *one* problem is quite special. This happens, of course, for the class NP: the famous SAT problem, among many others, is NP-complete. All the intricate structure of NP is captured by SAT.

The precise definition of being a complete problem S for a class C depends on the important notion of “reduction.” For S to be complete it must have two properties:

- (1) The problem S must be in the class C .
- (2) Given any problem T in the class there must be a reduction from T to S .

Note, the type of reduction can and does vary: there is polynomial-time reduction, Turing reduction, reductions based on lower complexity classes, and so on. The intuitive notion is, however, simple: T reduces to S provided the ability to solve the S problem allows one to solve T .

I wanted a name for a complexity class that does have this special property: having a complete problem. I have tried to create a good name, I do not know one that describes exactly this property. I thought it might make sense to call such a class a K -class after the usual symbol used for one of the first complexity classes ever discovered to have a complete problem. In recursion theory, see Rogers’ book, the letter K is used to denote the Gödel numbers of those Turing Machines that halt on input 0. If there is a known name, or a better one, let me know. For now a class is a K -class if it has a complete problem.

In Rogers’ book various notions of complete are defined. There are, by the way, important classes that are not known to be K -classes. Perhaps the most famous, notorious even, is BPP. This is the class of problems accepted by polynomial time probabilistic machines that have two-sided bounded error. One of the great open problems is: Does this class have a complete problem? Another is the same question for $NP \cap co - NP$.

All of this is laid out clearly in Rogers’ book, although he does not state some of what I have as explicitly. He assumes natural numbers and does not start with Boolean strings. But the notion of problems is clear, as is the notion of complexity classes.

39.5 Open Problems

There are countless open problems. The main questions fall into two types: what are the relationships between various classes and what are the properties of a class. The former most famous example is the $P = NP$ question, and the latter most famous is the question does BPP have a complete problem, is it a K -problem?

39.6 Notes and Links

Original post:

<http://rjlipton.wordpress.com/2010/11/07/what-is-a-complexity-class/>

Rogers' famous text:

<http://www.amazon.com/Theory-Recursive-Functions-Effective-Computability/dp/0262680521>

Leibniz cookies:

<http://en.wikipedia.org/wiki/Leibniz-Keks>

Referenced post for Boolean strings:

<http://rjlipton.wordpress.com/2009/10/04/the-world-is-digital/>

Leibniz on base 2:

<http://www.leibniz-translations.com/binary.htm>

Leibniz original:

http://ads.ccsd.cnrs.fr/docs/00/10/47/81/PDF/p85_89_vol3483m.pdf

Chomsky hierarchy:

http://en.wikipedia.org/wiki/Chomsky_hierarchy

BNF:

http://en.wikipedia.org/wiki/Backus-Naur_Form

Animal Farm:

http://en.wikipedia.org/wiki/Animal_Farm

Complexity class:

http://en.wikipedia.org/wiki/Complexity_class

Complexity Zoo:

https://complexityzoo.uwaterloo.ca/Complexity_Zoo

Tag cloud:

http://en.wikipedia.org/wiki/Tag_cloud

Picture credits:

1. Leibniz cookies

<http://en.wikipedia.org/wiki/Leibniz-Keks>

http://upload.wikimedia.org/wikipedia/commons/thumb/5/52/Choco_leibniz.jpg/200px-Choco_leibniz.jpg

2. Complexity Zoo wordle is original.

At press time, the Zoo is maintained at the University of Waterloo by the above-mentioned plus “Co-Zookeeper” Charles Fu and “Zoo Conservationist” Vincent Russo.

Ron Fagin is a great theorist who has made many important contributions to diverse aspects of computing. He is perhaps most famous for his brilliant categorization of polynomial time by second-order logic, which is known as Fagin's Theorem.

Today I plan on talking about a related theorem, due to Bruno Courcelle, which he proved in 1990. Yet the theorem is not, in my opinion, as well known as it should be. The truth is I did not know this pretty theorem until I ran across it recently. So perhaps everyone else knows Courcelle's result except for me. Ah well.

So why are we naming this chapter for Fagin? Actually, the file name we gave to the publisher is "courcelle.tex." The reason is that Courcelle proved a Fagin-type result.

40.1 Courcelle's Theorem

Monadic second-order logic, called MSO, allows one to quantify over predicates if, and only if, they have a single argument parameter. A seminal example of such a predicate is whether a string has a 1 in a position j given as argument. The string itself is not an argument, but rather a structure over which the predicate is defined. The edge relation of a graph, however, is binary, so MSO cannot quantify it. Nevertheless we can define monadic predicates on graphs, such as whether a vertex has odd degree.

Courcelle's theorem is quite striking, in my opinion:

Theorem 40.1 *Let ϕ be any graph property that is definable in MSO logic. For any fixed k , there is a linear time algorithm for testing the property ϕ on any graph of treewidth at most k .*

I will explain the theorem and try to give a high-level view of its proof. But, even if concepts associated to MSO such as treewidth are new to you, the theorem should still be striking. There are precious few theorems of the form that any graph property expressible in a powerful logic is computable on a class of graphs in *linear* time.

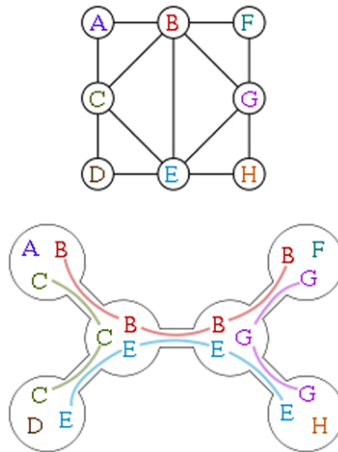
40.2 Treewidth

The concept of treewidth is fairly widely known, and was used extensively in the famous [work](#) of Neil Robertson and Paul Seymour on graph minors. I will include a definition here for completeness. My intuition is that for most graph problems the case of trees is easy: for example, there is a linear-time algorithm for tree isomorphism. The notion of small treewidth is an attempt—a very successful attempt—to generalize trees to include a much larger class of graphs.

A **tree decomposition** of a graph $G = (V, E)$ is composed of a family of sets of vertices B_1, \dots, B_m , called **bags**, and a tree T on the nodes $1, \dots, m$ with these properties:

- Every vertex of G is in at least one bag.
 - Every edge of G has both endpoints in some bag.
 - For all vertices of G , the set of bags that contain the vertex form a subtree of T .
- The last can be rendered more precise by stating: for all vertices v of G the set $\{i \mid v \in B_i\}$ is a subtree of T . The **width** of the decomposition is the maximum of $|B_i| - 1$ over all i . The **treewidth** of a graph G , $\mathbf{tw}(G)$ is the minimum such width over all valid tree decompositions. The -1 is there to make the treewidth of a tree 1.

Here is an example, from our friends at [Wikipedia](#), of a graph and a decomposition:



40.3 MSO

Suppose that $E(x, y)$ stands for the edge relationship of some simple undirected graph. Then first-order sentences allow variables only over vertices. For example, the sentence

$$\forall x, y, z \quad E(x, y) \wedge E(y, z) \rightarrow \neg E(y, x)$$

means that the graph has no triangle. This is a first-order sentence, since all the variables range over vertices of the graph.

First-order sentences can capture some interesting properties. They can express the property of having a k -clique for any fixed k . However, first-order logic is too weak to express many important graph properties, such as k -colorability.

This leads to the idea to extend the first-order logic to allow more powerful variables. A natural notion is to allow variables to range over arbitrary sets of tuples of vertices; this is called **Second Order (SO)** logic. This theory is very powerful, as shown by Fagin's Theorem itself:

Theorem 40.2 *Properties expressible by second-order logic existential sentences are precisely the complexity class NP.*

An existential sentence is a sentence of the form:

$$\exists R \exists S \dots \phi(R, S)$$

where R and S and \dots range over relations and ϕ is a first-order formula.

Clearly, there are properties of this form requiring more than linear time, even on trees. This is the reason that Courcelle's Theorem must restrict the properties to MSO. A sentence is a monadic one provided all the second-order variables range over sets. Notice that being expressible in MSO logic is a stronger requirement than being expressible in Second-Order logic: MSO is a subset of Second-Order logic.

It is now easy to define k -colorability. Here is how to express being 3-colorable:

$$\exists A \exists B \exists C: \phi(A, B, C),$$

where $\phi(A, B, C)$ uses first order quantifiers to check A, B, C form a partition of the vertices, and that if $E(x, y)$ then x and y are colored differently. Think of $A(x)$ as meaning x is colored "A," and the same for the other sets.

The logic MSO is quite powerful, but is still not powerful enough to define some properties: for example, the property of having a Hamiltonian Circuit is not definable in MSO.

40.4 Proof Idea

The proof of Courcelle's Theorem is based on two key ideas:

- (1) The ability to find a k -treewidth decomposition in linear time.
- (2) The ability to check the MSO property on a tree decomposition in linear time.

The first is a result due to Hans Bodlaender, who improved the original algorithm of Robertson and Seymour from quadratic time. Bodlaender surveyed this and other algorithms for finding treewidth in an article titled "Discovering Treewidth" in 2005 (referenced in the end notes).

Once a tree decomposition is found, the key is that a tree automaton can be constructed to check the MSO sentence. Tree automata are a generalization of finite automata from linear words to finite trees. Luckily many of the same constructions and properties of finite automata carry over to tree automata, and this allows the checking to be done in linear time.

40.5 Open Problems

Courcelle's Theorem is, in my opinion, quite pretty. However, the hidden constants are huge, and an obvious question is when can "real" linear-time algorithms be found for properties expressible in MSO? Another interesting open problem is, what is the largest class of sentences of SO that lead to linear-time algorithms on graphs of bounded treewidth?

40.6 Notes and Links

Original post:

<http://rjlipton.wordpress.com/2010/03/08/a-class-of-graph-properties-computable-in-linear-time/>

Robertson–Seymour Theorem:

http://en.wikipedia.org/wiki/Robertson-Seymour_theorem

Tree decomposition:

http://en.wikipedia.org/wiki/Tree_decomposition

Hans Bodlaender, "Discovering treewidth," in the proceedings of the 2005 "Software Seminar" (SOFSEM) conference, formally titled "Theory and Practice of Computer Science," pages 1–16. Also available at

<http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.61.3586>

Picture credits:

1. Tree decomposition:

http://en.wikipedia.org/wiki/Tree_decomposition

http://upload.wikimedia.org/wikipedia/commons/thumb/a/a7/Tree_decomposition.svg/240px-Tree_decomposition.svg.png

Daniel Lokshantov is a research fellow in algorithms whose PhD was on parameterized algorithms and complexity. When I looked at his home page I saw his email address is written in the form:

```
name [at] somewhere [dot] rest of the address
```

Of course the point is to “fool” robotic agents, and make it harder for them to collect email addresses for spamming. I have always wondered if this method really works, but at first glance it looked like he had gone one step better. His homepage was even trickier: it says his email is his office location and his phone number is his email. I thought: what a clever trick. But my student Subruk Kalyanasundaram pointed out, if I just maximized my browser window everything would line up. Subruk was right—oh well.

We will make some observations about a paper in the STOC 2010 conference on, among other things, one of my favorite problems: the Knapsack problem. The paper, authored by Daniel with Jesper Nederlof, uses a “new” method, and the method yields, in their hands, some quite striking results.

As with many new methods, the method is not completely new. I sometimes wonder why it is so hard to figure out who invented something. I think figuring out the inventor of a *method* rather than the inventor of a theorem is even harder. A theorem is a precise statement and acts like a point; a method is a general idea and acts more like a cloud. I can think of many examples of methods whose correct history would be hard, if not impossible, to unravel.

Some of the issues blocking the definitive statement of who invented a method are:

- Sometimes methods are independently discovered in essentially the same form. Until researchers become aware of each others’ work, it is often hard to decide who did what when. Even a result published in a well known conference or journal can be innocently missed. The amount of material published these days makes searching harder than ever.
- Sometimes methods are discovered in slightly different forms. Even if researchers are aware of each others’ work, they still may not realize their respective methods are really the same.

- Sometimes methods are discovered in different parts of science. In this case it is quite easy for researchers to have no idea the method is already known. A perfect example of this is probably the various “discoveries” of the quadratic algorithm for the [edit distance problem](#).

The method used by Daniel and Jesper was used previously by Yiannis Koutis—at least both used a very similar idea. So did Yiannis invent the method? I do not know, nor will I try to do a definitive historical study, but will try to point out the beautiful work that Yiannis also did.

Before we worry about who invented the method, let me explain what the method is.

41.1 The Method

Suppose you wish to calculate some value c . The CEM method has three steps:

- (1) Define a polynomial $f(x_1, \dots, x_n)$ so the coefficient of a given monomial of f is equal to c .
- (2) Construct an arithmetic circuit that computes the value of $f(x)$ at any point efficiently.
- (3) Use a fast algorithm to determine the coefficient of the given monomial.

The name, CEM, stands for **Coefficient Extraction Method**—please suggest a better name, but now I will use this name.

There are several remarks I will make about CEM. The first two steps, defining the polynomial and the arithmetic circuit, are quite creative. Each application requires that you figure out how to make your value c appear in a known monomial of an efficiently computable polynomial. The last step is a general lemma—so this step is essentially automatic.

The CEM method has more flexibility than the version I have given. The ring the polynomial is defined over can be selected in a clever manner too. For example, in some applications the ring is the integers, in some the complex numbers, in others it is a more general commutative ring. The latter happens in the work of Yiannis. This is an additional degree of freedom, and can be exploited to make CEM work to solve your problem.

Another source of greater flexibility is the monomial. In many applications the monomial is exactly known. In other applications the monomial is only partially known: for example, the key value may be the coefficient on any monomial of the form

$$x_1^{e_1} \cdots x_n^{e_n}$$

where all the exponents e_1, \dots, e_n must be odd. Clearly, the CEM is a powerful family of tools—this is the reason such pretty results have been obtained with the method.

41.2 Constant Term Method

As I have already stated, it is often the case with a “new” method, there is an older method quite similar in spirit to it. The **constant term method** (CTM) is used in combinatorics to enumerate many interesting objects and is very close to CEM. CTM uses a Laurent polynomial $f(x)$ so that the constant term of $f(x)$ has the same value as whatever you wish to count. Since $f(x)$ is a Laurent polynomial it can have negative as well as non-negative powers of x . Thus,

$$f(x) = 1/x^2 - 1/x + 1 + x - x^2$$

is a perfectly fine Laurent polynomial. Note, the constant term of $f(x)$ is not simply $f(0)$: the value of the function may not be defined at 0.

A famous example of this is the [Dyson Conjecture](#): the constant term of

$$\prod_{1 \leq i \neq j \leq n} (1 - t_i/t_j)^{a_i}$$

is equal to

$$\frac{(a_1 + a_2 + \dots + a_n)!}{a_1! a_2! \dots a_n!}.$$

The conjecture, now a theorem, was made by Freeman Dyson in 1962; it arose in a problem in particle physics.

There is one fundamental difference between CEM and CTM, besides the simple difference: CEM allows any coefficient and CTM always means the constant term. In combinatorics almost always one wants an expression for the term; in complexity theory one is usually happy with an algorithm for computing the term. We do want efficient algorithms, but do not need or expect the term will have a simple closed form expression, we might even allow small errors. In combinatorics the closed form solution often gives additional insights: for instance, using a closed form may allow a simple argument about the parity of the coefficient.

41.3 Applications of CEM

I would like to give two basic applications of CEM. As usual please see Daniel and Jesper’s [paper](#) from STOC 2010 and Yiannis’ [paper](#) from ICALP 2010 for details and exact statements.

- **The STOC Paper** Consider the subset sum problem: Given a set S of natural numbers s_1, \dots, s_n is there a subset I of $[n]$ so

$$\sum_{i \in I} s_i = w,$$

where w is the target. Let $f(x)$ be the polynomial

$$\prod_{i=1}^n (1 + x^{s_i}).$$

The key is that the coefficient on x^w is non-zero if and only if the subset sum problem has a solution. This follows since

$$(1 + x^{s_1})(1 + x^{s_2}) \cdots (1 + x^{s_n}) = \cdots + c_w x^w + \cdots,$$

where c_w is the number of ways to multiply the terms

$$x^{s_1}, \dots, x^{s_n}$$

together and yield x^w .

It is easy to see that this polynomial has an efficient arithmetic circuit, and thus CEM can be applied. Pretty neat.

- **The ICALP Paper** The second application uses a generalization (GCEM) of CEM: the detection of any *multilinear* term in a polynomial whose coefficient is odd. Consider the k -simple path detection problem: Given a graph, is there a k -simple path in the graph? There is a very clever reduction of this problem to the generalization of CEM that yields a random algorithm whose dominant cost is order $2^{3k/2}$. The exact running time is bit more complex, and one of the great things about the algorithm is it uses only polynomial space.

41.4 A Lemma

I will present one sample lemma—see the papers for details on how to solve CEM and GCEM.

Given a polynomial $P(x_1, \dots, x_n)$ the key parameters are D , S , and d . The first is defined by regarding each term of P in the form

$$x_1^{e_1} \cdots x_n^{e_n},$$

and letting D be the maximum of $e_1 \times e_2 \cdots \times e_n$. The other two are the size S and depth d of the circuit that computes the polynomial. Then CEM can be solved in time roughly DSd —here we are leaving out lots of logarithmic and other low-order terms. Also the space of the algorithm is quite small, with the dominant term being S .

While I will not state the lemma Yiannis uses in any detail, there is one aspect of his lemma I really like. As I stated before the methods can use different rings—they need not be restricted to integers or reals or even complex numbers. In Yiannis' work a trick he uses is to work over a ring that has nilpotent elements: recall x is nilpotent if some power of x is 0. See his well-written paper for the full details.

41.5 Open Problems

The main open problem is, can we use CEM to solve some other open problems? I am trying to use the method to solve some old problems, but have not had any success, yet. The method seems quite powerful, and I hope you can use it to get new theorems. Good hunting.

41.6 Notes and Links

Original post:

<http://rjlipton.wordpress.com/2010/03/03/beating-bellman-for-the-knapsack-problem/>

Referenced post on edit distance:

<http://rjlipton.wordpress.com/2009/03/22/bellman-dynamic-programming-and-edit-distance/>

Dyson conjecture:

http://en.wikipedia.org/wiki/Dyson_conjecture

STOC paper:

<http://www.ii.uib.no/~daniello/papers/SaveSpace.pdf>

ICALP paper:

<http://www.cs.cmu.edu/~jkoutis/papers/MultilinearDetection.pdf>

Albert Einstein is, well, Albert Einstein. He is perhaps the best known, most recognizable, iconic scientist of all time. He once said,

Do not worry about your problems with mathematics, I assure you mine are far greater.

We discuss some problems about solving equations—equations of a type that are not as well known as polynomial equations.

There is something fundamental about solving equations. Einstein faced the problem of solving equations, especially in his theory of General Relativity. Many early results in mathematics were driven by the desire to solve an equation of one kind or another. Even simple linear equations were hard for early mathematicians: consider the equation

$$2x + 7 = 3.$$

For us the solution is clear, $x = -2$, but negative numbers were not accepted as “real” numbers initially. See the fun book by Alberto Martínez titled *Negative Math* (referenced in the end notes) for a history of negative numbers.

The Greeks had even more difficult times with quadratic equations: what is the solution to

$$x^2 = 2?$$

The root is irrational, again a “real number” for us, but not apparently for the early Greek mathematicians.

For equations of one variable, in modern terms, there is the famous Fundamental Theorem of Algebra (FTA):

Theorem 42.1 Any polynomial equation with $n \geq 1$,

$$x^n + a_{n-1}x^{n-1} + \cdots + a_1x + a_0 = 0,$$

where the coefficients a_{n-1}, \dots, a_0 are complex numbers, has a solution over the complex numbers.

This ends the story of equations in one variable—or does it? Not quite. There are countless kinds of more-general equations to consider, indeed many generalizations of the notion of “polynomial” itself. I will discuss several such generalizations.

42.1 Polynomials Plus

The FTA has been generalized to include more general functions of all kinds. For what classes of functions f is the equation

$$f(x) = 0$$

guaranteed to have a solution over the real or complex numbers? Note that we need to have *some* restriction on f in order to get a reasonable theory.

One interesting family of functions are the harmonic functions. We can form some of them as polynomials in variables $z = x + iy$ and their complex conjugates $\bar{z} = x - iy$. For the case of two variables x, y with complex coefficients, this in fact gives all the harmonic polynomials. Thus for an ordinary (analytic) complex polynomial $p(z)$ we can form the harmonic polynomial $p^*(z) = p(z) + \bar{z}$. Thus, questions arise like: how many roots can the equation

$$p^*(z) = 0$$

have? The theory here is still incomplete, because the behavior of such equations is much more involved than the behavior of pure polynomials. Yet there are some quite pretty results that are known, and there are many interesting open questions.

Dmitry Khavinson and Genevra Neumann wrote a general introduction to this wonderful theory in a 2008 article titled “On the Fundamental Theorem of Algebra: A ‘Harmonious’ Path.” Right away the behavior of such polynomials is much more complex. For example, each of the harmonic polynomials

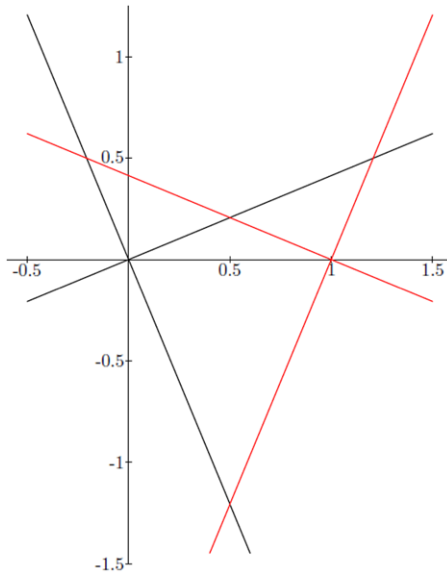
$$p_n(z) = z^n - \bar{z}^n = 0$$

has an *infinite* number of solutions. The equation holds if and only if z^n is a real number, so the solution set is the star formed by $2n$ equally spaced rays from the origin.

Given such infinite cases, it is remarkable that one can find broad conditions under which there are finitely many distinct zeroes—though still more than the degree as with ordinary polynomials. There is the following pretty theorem:

Theorem 42.2 *If $h(z) = p(z) - \overline{q(z)}$ is a harmonic polynomial of degree n such that $\lim_{z \rightarrow \infty} |h(z)| = \infty$, then h has at most n^2 zeroes.*

See the references in their paper for the details. The intuition behind this is to look at the behavior of the real and the imaginary parts of $h(z)$ separately. They have zero sets that are *lines*. Of course both the real and the imaginary parts must both be zero to have a zero of h , so the issue is how many intersections can there be? This is where the $n^2 = n \cdot n$ comes from. The black lines are where the real part is zero, and the red lines where the imaginary part is zero.



42.2 Polynomials Plus Plus

Khavinson and Neumann proved a related result concerning rational harmonic functions—polynomials plus plus. In a research paper titled “On the Number of Zeros of Certain Rational Harmonic Functions,” they [proved](#):

Theorem 42.3 *Let $r(z) = p(z)/q(z)$, where p and q are relatively prime polynomials in z , and let n be the degree of r . If $n > 1$, then the number of solutions of*

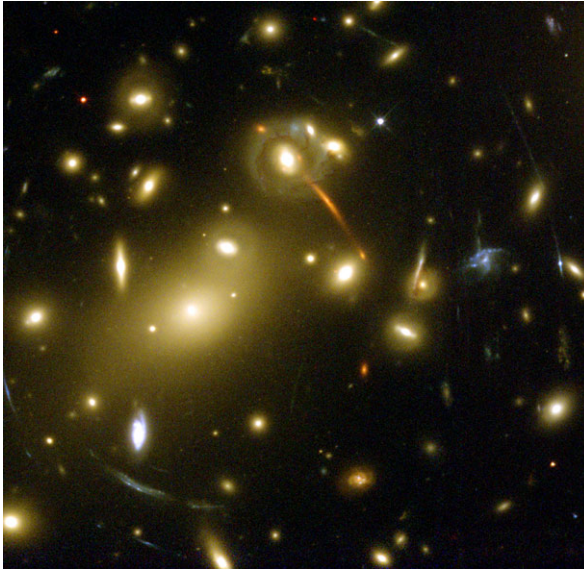
$$r(z) - \bar{z} = 0$$

is at most $5n - 5$.

It turns out that the mathematical upper bound is closely related to the physics concept of gravity lenses. They have been called “[accidental astrophysicists](#).” I will explain why in the next section.

42.3 Gravity Lenses

According to Einstein's theory of General Relativity light is bent in the presence of gravity. This leads to the phenomena of *gravity lenses*. A single object in the presence of a powerful source of gravity can appear as multiple objects. Here is a picture of the formation of such images:



One of the critical questions is how many images can be formed? This is of importance to astrophysicists.

The surprise is that it is possible to model the number of images not with a polynomial equation, but with a harmonic equation. Actually, the astrophysicist Sun Rhie proved a lower bound, and she conjectured that her construction was optimal. Thus, when Khavinson and Neumann proved an upper bound, they were actually solving her conjecture. Apparently they did not know this connection before they proved their theorem. Science, especially mathematics, can be surprising—there are often unforeseen connections. Thus the name “accidental astrophysicists.”

42.4 Open Problems

There are many complexity questions that arise from these theorems. Can we decide efficiently how many roots a harmonic polynomial has? Also can we find them—at least approximately? Also are there applications to complexity theory of these results?

42.5 Notes and Links

Original post:

<http://rjlipton.wordpress.com/2010/10/28/the-accidental-astrophysicists/>

Albert Martinez, *Negative Math: How Mathematical Rules Can Be Positively Bent*, Princeton University Press, 2005.

<http://www.amazon.com/Negative-Math-Mathematical-Rules-Positively/dp/0691123098>

Dmitry Khavinson and Genevra Neumann, “From the Fundamental Theorem of Algebra to astrophysics: A ‘harmonious’ path”. *Notices of the AMS* **55(6)**, June/July 2008, 666–675. Available at <http://www.ams.org/notices/200806/tx080600666p.pdf>

Dmitry Khavinson and Genevra Neumann, “On the number of zeros of certain rational harmonic functions,” *Proc. Amer. Math. Soc.* **134** (2006), 1077–1085. Available at <http://www.ams.org/journals/proc/2006-134-04/S0002-9939-05-08058-5/home.html>

Accidental Astrophysicists:

http://www.sciencenews.org/view/generic/id/33082/title/Math_Trek__Accidental_astrophysicists

Gravitational lensing:

http://en.wikipedia.org/wiki/Gravitational_lens

Picture credits:

1. Roots of system: screen capture from page 5 of the Khavinson–Neumann paper <http://www.ams.org/notices/200806/tx080600666p.pdf>
2. Photo of gravitational lensing cropped from http://en.wikipedia.org/wiki/Abell_2218

Denis Thérien is an expert on low-level complexity, especially the computational theory of groups and related algebraic structures. In a 2005 [press release](#) announcing a promotion at McGill University, it was stated that he is:

an accomplished mathematician, whose research interests include complexity theory, classifying problems in terms of resources required to compute their solution.

Indeed. I like the phrase, “classifying problems in terms of resources required to compute their solution”—very well said.

We will talk about solving equations over groups, and its relationship to complexity theory.

I have two reasons for discussing this. First, there is a chance to solve one of the main open problems in low-level complexity theory, if we could show that certain equations over groups have solutions. The question is the famous one left open by David Barrington: can bounded-width computations over solvable groups compute NC^1 ? Barrington’s famous [theorem](#) showed that such bounded computations can do this over simple groups. Barrington, Straubing, and Thérien worked on this problem proving some partial results, but the main question is unresolved. If solvable groups could, indeed, work in place of simple groups, then the following surprising result would be true:

$$\text{NC}^1 \subseteq \text{TC}^0.$$

This is the last theorem proved in Barrington’s paper: if solvable groups can compute NC^1 , then the above is true.

Second, the theory of equations over groups is potentially useful to computational complexity in other ways. The theory is quite difficult, since groups can behave badly in many ways. But I hope this short introduction to the area will make you want to learn more, and perhaps solve some open problems. Either resolve Barrington’s question conclusively rather than conditionally, or find more applications of the theory of equations. Or something else.

43.1 Equations over a Group

Let's start to look at equations over a group. As usual we will use xy to denote the product of x and y in a group. Also the interesting cases are almost always when the groups are non-commutative, so xy is not the same as yx in general.

What is an equation over a group G ? An example of an equation is:

$$xaxb = 1.$$

Note, a, b are in G and x is the unknown. The equation has a solution over G provided there is an element c in G so that

$$cacb = 1.$$

Pretty natural.

Many times such an equation will not have a solution: that is for all c in G the above is false. This does not stop us. A natural idea is to ask whether there is a group \widehat{G} so that $G \subset \widehat{G}$ and for some c in \widehat{G}

$$cacb = 1.$$

In a sense we have extended the group G and added enough elements to it so that the equation has a solution. Note, this is exactly the issue that bothered early mathematicians, what do you do with the equation

$$x^2 = 2$$

over the rationals? There is no solution, but there is one in a larger world—however you need to adjoin the square root of 2.

The obvious question is: can we always find the larger group \widehat{G} so this is possible?

43.2 No Fundamental Theorem of Groups

The answer, you may have guessed, is that there is no analogue of the Fundamental Theorem of Algebra (FTA). There are group equations that cannot be solved even if we can extend the group and add new elements. This is the reason the theory is so different from solving equations over fields.

Pick any finite group G with elements a and b so that $a^2 = 1$ and $b^2 \neq 1$. This is nothing more than asking for a group with an element of order 2 and another with a different order. To make it interesting let's also assume that a is not the identity element. Now consider the equation

$$xax^{-1}b^{-1} = 1.$$

The unknown is, as usual, the x . I claim that this equation has no solution in any group that contains G . Assume there was an x in some larger group. Rewrite the equation as

$$xax^{-1} = b.$$

Then, square both sides of the equation,

$$xax^{-1}xax^{-1} = xa^2x^{-1} = 1 = b^2 \neq 1.$$

This is a contradiction. Thus, there is no FTA for group equations.

43.3 A Partial Fundamental Theorem

A pretty case where there always is a solution is given by the following [theorem](#) proved in 1962 by Frank Levin:

Theorem 43.1 *Suppose that G is a finite group, and let*

$$g_1xg_2xg_3 \cdots g_mxg_{m+1} = 1$$

be an equation in the variable x , where each g_i is in G . Then, there is a finite group \widehat{G} that contains G as a subgroup, and in \widehat{G} the equation has a solution.

Even better the proof is quite constructive, and actually proves much more. Besides giving \widehat{G} explicitly some properties of G are preserved. For example, if G is a solvable group, then so is \widehat{G} .

I will not have time to explain the proof, but it is interesting that the proof uses the wreath product of groups. Recall the wreath product of groups is very close to the famous zig-zag product of graphs—the wreath product is an older construction. See the great [survey](#) by Shlomo Hoory, Nathan Linial, and Avi Wigderson for an introduction to how products are used in expander graphs and other wonderful things.

Note, Levin's theorem does *not* allow the “bad” equation that has no solution. It does this by simply not allowing x^{-1} anywhere. The following is fine:

$$x^6ax^7bx = 1,$$

since we can get positive powers of x by repeating. But we cannot get negative powers of x .

43.4 Toward the General Case

I am not an expert, not even close, on equations over groups. I do hope the above shows some of the issues that arise in even the most basic cases. I really need theorems that handle systems of equations and more complex situations.

There is one further general comment that I would like to make. There is a beautiful conjecture that is called the Kervaire–Laudenbach [Conjecture](#). It attempts to handle inverses—as we have seen this is not trivial.

The conjecture looks an equation

$$g_1 x^{\epsilon_1} g_2 x^{\epsilon_2} g_3 \dots g_m x^{\epsilon_m} g_{m+1} = 1,$$

where the ϵ_i are in $\{-1, 1\}$. The equation is conjectured to be solvable provided

$$\epsilon_1 + \epsilon_2 + \dots + \epsilon_m \neq 0.$$

Recall the bad equation was

$$xax^{-1}b^{-1} = 1,$$

which violates this constraint: $-1 + 1 = 0$.

43.5 Universal Groups

Let's call a group G *universal* if Barrington's theorem can be proved over this group. We know the following key insights:

- (1) If G is a finite non-Abelian simple group, then G is universal.
- (2) If G is nilpotent, then it is not universal.
- (3) If G is solvable and universal, then NC^1 is contained in TC^0 , making them equal. Perhaps we should call such a group a *Barrington group*, but I hope universal is fine.

The relationship between universal solvable groups and equations over groups is that there are equation systems such that if they have a solution, then there is a universal solvable group. I have many such families and will discuss them in detail another time. The general idea is to encode an “AND” gate by the equations. Here is an example of a system \mathcal{S} : Let x_1, x_2, \dots, x_n be the variables, and let $A \neq 1$ be an element in solvable group G . There are four equations:

$$\begin{aligned} x_1 x_2 \dots x_n &= 1, \\ x_1 A x_2 A \dots x_k A x_{k+1} x_{k+2} \dots x_n &= 1, \\ x_1 x_2 \dots x_m x_{m+1} A x_{m+2} A \dots x_n A &= 1, \\ x_1 B_1 x_2 B_2 \dots x_n B_n &= A, \end{aligned}$$

where B_i is equal to A , or A^2 based on the rules: If i is in $[1 \dots m]$ or $[k + 1, \dots, n]$ then $B_i = A$; otherwise, it is A^2 .

$$B_i = \begin{cases} A, & \text{if } i \in [1 \dots m], \\ A, & \text{if } i \in [k + 1, \dots, n], \\ A^2, & \text{otherwise.} \end{cases}$$

Note, we assume that $m < k$. If \mathcal{S} can be solved over some solvable group then that group is universal.

I've probably made an error in the indices in the above system \mathcal{S} . The idea goes as follows: The product $x_1 \dots x_n$ equals 1. Then, if A is placed in the first k positions or the last m positions the product is still 1. However, if **both** are done then the product is equal to A . The tricky part is working out the indices to reflect the overlap part, since there are two A 's that yield A^2 .

43.6 Open Problems

The main issue, I believe, is to understand the theory of group equations better. I feel that this would help us in complexity theory a great deal. As noted it could even solve some of the main open questions in the area of low-level complexity.

I would like to point out that there is very nice work on the complexity of solving equations over groups. See the [paper](#) of Mikael Goldmann and Alexander Russell for a number of results. For example they proved that the problem of determining if a (single) equation has a solution is NP-complete for all nonsolvable groups G . Perhaps I will discuss these and related results in the future.

43.7 Notes and Links

Original post:

<http://rjlipton.wordpress.com/2010/11/03/equations-over-groups-a-mess/>

McGill press release:

<http://www.istpcanada.ca/WhoWeAre/BoardofDirectors/DenisTherien/index.php>

Referenced post on Barrington's theorem:

<http://rjlipton.wordpress.com/2009/03/02/barrington-gets-simple/>

Paper by Frank Levin:

<http://www.ams.org/journals/bull/1962-68-06/S0002-9904-1962-10868-4/S0002-9904-1962-10868-4.pdf>

Hoory–Linial–Wigderson survey:

http://www.cs.huji.ac.il/~nati/PAPERS/expander_survey.pdf

Kervaire–Laudenbach conjecture:

<http://arxiv.org/pdf/math/0409146v1>

Goldmann–Russell paper:

<http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.28.9799>

Andreas Björklund is a complexity theorist who has many interesting results on the complexity of hard problems. He is a master at algorithms for hard problems such as computing the permanent and computing the number of solutions to #P complete problems.

We describe his paper at the 2010 FOCS conference, titled “Determinant Sums for Undirected Hamiltonicity” (referenced in the end notes). I think it is a beautiful result, and a clever proof. It solved a decades-old problem.

I also think that Björklund’s result is an example of a trend among theory results. Most researchers believe that NP-complete problems not only lack polynomial-time algorithms, but require exponential time in the worst case. Hence there is greater attention to algorithms with running time

$$c^n$$

where $c > 1$ is a constant and n is a parameter of the problem. Note, usually n is not just the number of bits required to represent the problem. An example is SAT itself: there the parameter n is the number of variables. The conjecture, in this case, that the best algorithm is exponential in n is called the [Exponential Time Hypothesis](#). I must add that I find this conjecture very hard to believe. Yet all algorithms to date for SAT do indeed take exponential time—perhaps I am wrong.

44.1 The Result

A *Hamiltonian cycle* is a cycle in an undirected graph that visits each vertex exactly once and also returns to the starting vertex. A graph is said to be Hamiltonian if it contains a Hamiltonian cycle.

The obvious algorithm is try all cycles: this would take $n!$ time at least. One principle I have always told my students is:

No “real” problem’s best algorithm runs in $n!$ time.

I have many times seen initial algorithms that take this time, but there always seems to be a way to get a better bound. The classic result, with the much better bound, is due to Michael Held and Richard Karp in 1970:

Theorem 44.1 *There is a deterministic algorithm that takes an undirected graph on n vertices as input and decides if the graph is Hamiltonian in $O^*(2^n)$ time.*

The key insight of their famous result is to make clever use of dynamic programming. The induction essentially keeps track not of ordered sequences of vertices visited, but of sets of vertices visited. This can be used to reduce $n!$ to 2^n .

Björklund's new [result](#) is a major improvement over this result—the first in 40 years. Again the star means to ignore polynomial factors of n .

Theorem 44.2 *There is a Monte Carlo algorithm detecting whether an undirected graph on n vertices is Hamiltonian or not running in $O^*(1.657^n)$ time, with false positives and false negatives occurring with probability exponentially small in n .*

The major point is that his algorithm runs in exponential time still, but the constant is now $1.657 < 2$. As I stated earlier I believe that we will see many more results like this; results that improve the constant on an exponential algorithm.

44.2 An Overview of the Proof

Ryan Williams was kind enough to supply a summary of the proof. I have made some changes, so any errors are all mine. But first, here is my summary of his summary:

- (1) Replace the graph's adjacency matrix so that each 1 is replaced by a different variable—except that symmetric positions get the same variable.
- (2) Then notice that the determinant of this matrix is the same as the permanent over any finite field of characteristic 2. This follows since $-1 = 1$ in such a field.
- (3) This permanent is, of course, a very complex polynomial with an exponential number of terms. However, it is easy to tell by evaluation at a random point whether or not the polynomial is trivial—has no terms or not.
- (4) The claim we would like to make, but cannot, is that this polynomial is trivial if and only if there is **no** Hamilton cycle. Of course this would imply a randomized polynomial-time algorithm, so it is too much to hope for.
- (5) The claim is therefore false: the polynomial can be trivial for the wrong reasons. The fix to this is the most clever part of the proof. The idea is to add extra labels to the variables to avoid being trivial for the wrong reason. The cost of this last step is exponential. But, it is not too exponential.

44.3 Proof Summary

Here is Ryan’s summary of the proof. We will first start with a broken polynomial-time algorithm for the Hamiltonian cycle problem, and then over a series of steps we will try to fix it.

Let $G = (V, E)$ be an undirected graph. Define a matrix A over variables as follows.

$$A(i, j) = A(j, i) = x_{ij}$$

when $i \neq 1$ and $j \neq 1$ and $\{i, j\}$ in E . Let $A(1, i) = x_{1i}$ when $\{1, i\}$ in E , and let $A(i, 1) = x_{i1}$ when $\{1, i\}$ in E .

That is, the matrix A is similar to an adjacency matrix, and it is symmetric except for the first row and first column. Suppose we plug in random nonzero values for the x_{ij} ’s into A drawn from the field $GF(2^n)$, and evaluate the permanent of the resulting matrix. If the permanent is zero, say “no Hamilton cycle.” If nonzero, say “Hamilton cycle.”

This is a polynomial-time algorithm, since permanent = determinant over characteristic two. And, as the paper states, the permanent sums over all oriented cycle covers in the graph, and for each cover we have the product of all x_{ij} where (i, j) is an arc in the cycle cover. That is, it enumerates all cycle covers along with all possible orientations of the cycles in the cover. One might expect that, over characteristic two, the “bad” cycle covers might all cancel, and so we only get a nonzero evaluation with high probability (whp) if there’s a Hamiltonian cycle. Note that if there *is* a Hamiltonian cycle, then we *do* get a nonzero evaluation whp, because for each Hamiltonian cycle we have two orientations of it which contribute an expression of the form

$$(x_{1v} + x_{v1})P$$

where P is a product of all other $n - 2$ arcs in the cycle to the sum, for some node v . By the Schwartz–Zippel lemma (referenced in the end notes), the sum of these will evaluate to nonzero with high probability, regardless of what the rest of the polynomial looks like.

However the algorithm fails to find a Hamiltonian cycle with high probability, for one reason: these directed cycle covers can have “2-cycles” which go from some vertex i to j then back to i . If the cycle cover is a collection of 2-cycles plus a cycle involving vertex 1, then when you “re-orient” one of these 2-cycles as in Lemma 1 of the paper, you get the *same* cycle cover back, not a different one. Hence you cannot “pair up” the cycle covers that have these 2-cycles so that their corresponding monomials all cancel out modulo 2. For all other cycle covers which are not Hamiltonian, they actually *do* cancel in the evaluation, as the proof of Lemma 1 shows.

The key insight is to assign distinct “labels” to all the arcs in the cycle cover, so that when you reverse the directions of arcs in a 2-cycle, you do get a different 2-cycle. If you have the 2-cycle $(1, 2), (2, 1)$ with label “a” on $(1, 2)$ and label “b”

on $(2, 1)$, then the reverse cycle now has label “b” on $(1, 2)$ and label “a” on $(2, 1)$, which is a “different” assignment of labels.

The good news is that this works. The bad news is there are potentially too many labels needed. A naive method would use n labels and this would yield an algorithm for the Hamiltonian cycle problem that runs in 2^n , which is too slow. But at least we can compute this “labeled cover” problem in $2^k \text{poly}(n)$ time, where k is the number of labels needed.

The next important idea is to figure out a way to use fewer than n labels. We will pick a random bisection of the graph into roughly equal parts V_1 and V_2 . Fix a Hamilton cycle. With decent probability, there are at most $n/4$ edges of this cycle where both endpoints of the edge are in V_1 . The paper explains how to use this partition to reduce the number of labels needed.

44.4 Open Problems

As I said earlier, I believe there is a “trend” among research that speaks to a quotation of Alan Perlis:

“... for every polynomial-time algorithm you have, there is an exponential algorithm that I would rather run.”

I am quite interested in work that is trying to improve the exponential on the running time. There have already been several neat papers, and I expect to see more.

44.5 Notes and Links

Original post:

<http://rjlipton.wordpress.com/2010/09/09/beating-a-forty-year-old-result-hamilton-cycles/>

Exponential-time hypothesis:

http://en.wikipedia.org/wiki/Exponential_time_hypothesis

Andreas Björklund, “Determinant sums for undirected Hamiltonicity,” in the proceedings of the 2010 IEEE Conference on Foundations of Computer Science. Also available at

<http://arxiv.org/abs/1008.0541>

Referenced post with Schwartz–Zippel lemma:

<http://rjlipton.wordpress.com/2009/11/30/the-curious-history-of-the-schwartz-zippel-lemma/>

Referenced post with Perlis quote:

<http://rjlipton.wordpress.com/2009/02/13/polynomial-vs-exponential-time/>

David Hilbert was arguably the greatest mathematician whose career spanned the 19th and 20th centuries. His famous list of 23 open [problems](#) shaped mathematics throughout the last century, and their effect is still felt today. The address in which he presented the problems started:

Who of us would not be glad to lift the veil behind which the future lies hidden; to cast a glance at the next advances of our science and at the secrets of its development during future centuries? What particular goals will there be toward which the leading mathematical spirits of coming generations will strive? What new methods and new facts in the wide and rich field of mathematical thought will the new centuries disclose?

Today, most of our presentations are informal: we use software-generated slides; we do not read from a written script, we get questions during the talk—it is hard to imagine a talk like Hilbert’s ever again. What a beautiful opening: *Who of us would not be glad to lift the veil behind which the future lies hidden. . .* It was an impossible act to follow.

We should talk about a famous theorem of Hilbert’s, called the **Nullstellensatz**. This theorem has played an important role in many aspects of complexity theory. However, I have a somewhat different view of why the theorem is important: it is all about the Greek letter Δ .

I cannot resist telling you a story about the Greek alphabet. When I first got to Georgia Tech, I was working with Mary Jean Harrold on a program testing project. The project had a quite neat idea—all hers—but I was helping a bit. One thing I really did not like was the name of the project: it was not very catchy nor was it descriptive.

One day it occurred to me what to call the project: call it the *delta* project. My reasoning was there already was *alpha* testing and *beta* testing, so the natural next advance had to be *delta* testing. I sent an email to Mary Jean, with my suggested name and why it was the right name for the project. She sent back a short message:

Dick, great idea on the name, but in the Greek alphabet it is: *alpha*, *beta*, **gamma**, . . .

She called the project Gamma. So much for my understanding of the Greek alphabet. Oh well.

45.1 Hilbert's Nullstellensatz

In German, according to [Wikipedia](#), “Nullstellensatz” means the “theorem of zeros,” or more literally, “zero-locus-theorem.”

Theorem 45.1 *Suppose that p_1, \dots, p_m are polynomials in $\mathbb{C}[x_1, \dots, x_n]$. Then, they have no common zero if and only if there are polynomials a_1, \dots, a_m also in $\mathbb{C}[x_1, \dots, x_n]$ so*

$$a_1 p_1 + \dots + a_m p_m = 1. \quad (45.1)$$

As with many “if and only if” theorems, one direction is easy and one is hard. The easy direction is: if (45.1) has a solution, then there can be no common zero. For if

$$p_1 = 0, \dots, p_m = 0$$

have a common zero, then

$$a_1 p_1 + \dots + a_m p_m = 0 = 1,$$

a contradiction. The hard direction is: if the equations

$$p_1 = 0, \dots, p_m = 0$$

have no common zero, why should the polynomials a_1, \dots, a_m exist? This is the core of the theorem, and works because \mathbb{C} is an algebraically closed field.

Further, there are quantitative versions of the Nullstellensatz, in which the degrees of the polynomials a_1, \dots, a_m are bounded by an explicit function of m, n , and the maximum degrees of the p_i 's.

45.2 An Application

Why is this theorem important? Usually a theorem is important if it has many applications. There are other reasons, but this is the main one. There are exceptions: Fermat's Last Theorem is probably more important for the proof technology used by Andrew Wiles, than for applications of the theorem. Indeed, I believe that there are few applications of Fermat's Last Theorem—of course it is still one of the great results of number theory.

Why is this theorem, the Nullstellensatz, important? It is because it has many applications. The application I will present in a moment is a complexity-theory-oriented one.

Consider a polynomial map $f : \mathbb{C}^n \rightarrow \mathbb{C}^n$. An important property the map f may have is being injective. The definition of this property is simple:

$$\forall x \forall y \quad f(x) = f(y) \rightarrow x = y.$$

That is, a map is injective if any two distinct points map to distinct values.

In general there is not much more to say, but in the special case when the map is a polynomial mapping and the field is the complex numbers, one can say much more. Because of the Nullstellensatz, there is a completely different characterization of being injective. A mapping f is injective if and only if

$$\exists \mathbf{a} \Delta_f \mathbf{a}$$

where $\Delta_f(\mathbf{a})$ is a conjunction of polynomial equations that depend only on the variables \mathbf{a} .

I will explain the details of $\Delta_f(\mathbf{a})$, but the key is it uses only existential quantifiers. Thus, the notion of “injective” can be expressed both as a universal formula and as an existential formula. The former is the standard definition; the latter is only possible because we have access to the Nullstellensatz.

The ability to express a universal property by an existential one is very special, and helps me understand why the Nullstellensatz is so powerful. I hope it also helps you.

The Greek letter Δ is used to denote the property of being representable as universal and existential formulas. In complexity theory, for example, problems in both NP and co-NP are special, and this complexity class is sometimes denoted by Δ_1^{poly} .

Proving—in 1975—that primality was unconditionally in Δ_1^{poly} , by Vaughan Pratt, was a beautiful [result](#). Proving—in the future?—that graph isomorphism is in Δ_1^{poly} would be another beautiful result.

45.3 How to Express Injective as an Existential Formula

I owe you the description of $\Delta_f(\mathbf{a})$. To avoid excessive notation let f be the map

$$f(x, y) = (g(x, y), h(x, y))$$

where g and h are polynomials. Then, consider the following equations:

$$g(x, y) - g(u, v) = 0,$$

$$h(x, y) - h(u, v) = 0,$$

$$z(x - u) - 1 = 0.$$

They have a solution if and only if there are values giving $f(x, y) = f(u, v)$ and $x \neq u$. There is a similar set of equations for the event that $y \neq v$.

Let’s restrict our attention to the first case: $x \neq u$. Then, by the Nullstellensatz the equations have no solution if and only if there are polynomials $q(x, y, u, v, z)$, $r(x, y, u, v, z)$, and $s(x, y, u, v, z)$ so that

$$q \cdot (g(x, y) - g(u, v)) + r \cdot (h(x, y) - h(u, v)) + s \cdot (z(x - u) - 1) = 1.$$

The effective version of the Nullstellensatz allows us to write out this equation in terms of the coefficients of the polynomials q, r, s . This finally yields a set of equations that has a solution if and only if the original 1-to-1 condition was true. This defines the formula $\Delta_f(\mathbf{a})$.

45.4 Open Problems

Did this example application help with the understanding of why the Nullstellensatz is important? What are other applications of the Nullstellensatz to problems relevant to complexity theory?

45.5 Notes and Links

Original post:

<http://rjljpton.wordpress.com/2010/02/18/alpha-beta-delta-and-the-nullstellensatz/>

Hilbert's 23 problems:

<http://aleph0.clarku.edu/~djoyce/hilbert/problems.html>

Nullstellensatz:

http://en.wikipedia.org/wiki/Hilbert's_Nullstellensatz

Pratt's result:

http://en.wikipedia.org/wiki/Pratt_certificate

Mary Jean Harrold passed away in September 2013. We wrote a memorial of her at

<http://rjljpton.wordpress.com/2013/09/22/mary-jean-harrold/>

John Hopcroft is one of the great researchers who helped shape much of the foundations of the theory of computation. Some of this he did through his research, some through his great graduate students, some through his early textbooks. He did seminal work on algorithms and on basic complexity theory, leading to a well-deserved Turing Award in 1986, joint with Bob Tarjan.

But let's talk about a much different kind of "result." No theorems were proved—sounds like "no animals were harmed during the filming of this picture." This work was joint with me, but it was never published. Actually, it was not possible to publish the work—yet it has had a real impact on the community.

John's work on algorithms included the start of the modern approach to [graph isomorphism](#)—joint with his then PhD student Merrick Furst and with Eugene Luks—and some of his work on complexity theory is still the best known. For example, with Wolfgang Paul and Leslie Valiant, he [proved](#):

Theorem 46.1 For any $s(n) \geq n$,

$$\text{DTIME}(s(n)) \subseteq \text{DSpace}(s(n)/\log n).$$

This is proved using the famous pebbling argument. If you do not know this result, it is a beautiful proof—take a look, since it is still the best result known about time and space.

Hopcroft has not only proved great results, he has a sweeping vision of the field of computing, a vision that he shared with his students and others. One of his ideas—it is not an exact quote—is:

Work on important problems. They may be as hard as other problems, but if you solve them, then you would have made a real contribution.

This explains why he worked on such hard problems as graph isomorphism, while others stayed away. If you are going to work hard and perhaps solve a difficult problem, then make it an important problem. I like this idea—although I have violated it plenty of times and worked on hard problems that few cared about. I do agree with John: work on important problems. I wish I had listened more to him. Oh well.

46.1 A National Meeting?

John and I created the notion of “Federated Conferences”; we did this back many years ago. Here is the story of how it happened.

I was on an advisory board for NSF chaired by Rich DeMillo in the early 1990s. John was a member of the board. During one of our meetings John asked for a chance to explain what he thought was a national need of the computer science community.

John pointed out that all the other sciences had large national meetings. This included: biology, chemistry, mathematics, physics, and all the rest. Only computer science seemed to have no national meetings. He listed many of the advantages such a meeting would have:

- Bringing all the various parts of computer science together, it would help make the field more of a real community.
- Being large, it would help create excitement.
- Being national, it might attract national-level press coverage.

He added other advantages of such a meeting.

As you would imagine there was lots of discussion, with lots of push back. Computer science had already had national meetings, most of them were weak and not attended by the top researchers. The top people in each area went to their respective top meetings:

- database researchers → SIGMOD;
- programming language researchers → POPL;
- system researchers → SIGOPS;
- theory researchers → STOC;
- ...

The last thing we needed, most argued, was another meeting; there were already too many meetings. Even worse, the last thing we needed was another weak meeting. The central question put to John was:

Why would a theorist send a top paper to this new national meeting, when they could send it to FOCS/STOC and get more “credit”?

Indeed why. John agreed with all these issues, but repeated his position: we needed a national meeting to be on a par with the other classic sciences.

46.2 Federated Conferences

I listened and suddenly had an idea on how to solve John’s problem. I said what if we created a new meeting made up of the old meetings. The point was to simply have existing conferences meet at the same time and at the same place. Just co-locate in time and space.

This caused lots of discussion. Most liked the idea, and as a group we refined it quickly. Each conference would keep its current rules of how to accept papers; how long the talks were; whether the meetings were parallel or not; whether they

had keynotes or not—they would not change anything. The only difference was that they would give up control of when and where the meetings were held.

The meeting ended and we eventually handed the idea off to the Computing Research Association, under whose aegis this was being organized. They followed through and made it happen, and that is how we got [federated meetings](#).

46.3 Open Problems

I hope it was okay to voice this story. Most—almost all—of you probably have no idea where the federated meetings came from, and most probably do not care. I wanted to make a point about problem solving. Problems come in all flavors and from all places, we should be flexible and be prepared to think out of the box.

I am proud to be part of this creation of the federated meetings. I do wish that John and I received a nickel for each person who attends them. Oh well.

The main open problem is: do these federated meetings really help? Do they really get people to talk, who might otherwise not see each other? What do you think?

46.4 Notes and Links

Original post:

<http://rjlipton.wordpress.com/2010/05/23/lets-have-a-meeting/>

Furst–Hopcroft–Luks paper:

<http://portal.acm.org/citation.cfm?id=1398509.1382665>

Hopcroft–Paul–Valiant pebble game:

http://en.wikipedia.org/wiki/Pebble_game

ACM federated meetings:

<http://www.acm.org/fcrc/>

Dick Karp is to blame. Without him we would not have the annoying $P = ?NP$ question in its full regalia, without him we would not have thousands of problems all appearing to be hard to solve, without him modern computational complexity would be completely different. Just to be clear—in case my words do not come across properly—we owe Dick a great debt. Without his work, his leadership, his brilliant insights, the field would be very different, and we owe him a great deal of thanks. Thanks, Dick.

We will talk about an ancient theorem of Ravi Kannan proved in 1982, a beautiful [result](#), yet still one of the few known general circuit lower bounds. There have been some other results proved since then, but Ravi's is a landmark. It is related to a result of Karp and myself, commonly called the [KL theorem](#):

Theorem 47.1 *If SAT has polynomial-size circuits, then the polynomial-time hierarchy (PH) collapses to its second level, Σ_2 .*

This has been used as both a theorem and a research direction over the years. A beautiful [improvement](#), due to Jin-Yi Cai, sharpens the upper bound to a complexity class called S_2^P .

One key application of the KL theorem is in the proof of the beautiful result of Ravi Kannan. I will discuss how to **try** and avoid using our theorem—what am I thinking?—to prove his theorem.

47.1 Kannan's Theorem

A modern statement of Ravi's theorem uses $\text{SIZE}(n^k)$, which is the complexity class of sets S so that for each length n ,

$$C(S_n(x)) \leq n^k,$$

where $C(S_n)$ is the Boolean complexity of the function

$$S_n(x) = 1 \quad \text{if and only if} \quad x \in S.$$

Theorem 47.2 *For any $k > 1$, there is a set S in Σ_2 such that*

$$S \notin \text{SIZE}(n^k).$$

Yes, Ravi actually proves more— $S \in \Sigma_2 \cap \Pi_2$ —but all my comments will apply to the stronger theorem, so I will only discuss Σ_2 here.

47.2 Kannan's Proof

Ravi first shows there is a language L in Σ_4 such that

$$C(L_n) \geq n^k$$

for all n . He then uses KL in the following manner:

- (1) The language SAT has polynomial-size circuits. Hence by KL, the polynomial hierarchy collapses to Σ_2 , and therefore L is in Σ_2 . Thus, in this case his theorem is true.
- (2) The language SAT requires superpolynomial-size circuits. In this case his theorem is also true, since SAT is in NP which is contained in Σ_2 .

This is a very neat proof by disjunction. I pointed out [once](#) the great John Littlewood used a similar trick to prove a theorem in number theory:

- (1) If the Riemann Hypothesis is true, then the primes behave well enough to prove his theorem.
- (2) If the Riemann Hypothesis is false, then the primes behave poorly in a way that implies his theorem.

So Ravi is in good company.

47.3 Kannan's Proof—Can We Do Better?

There is an interesting issue with Ravi's proof of his theorem—it is non-constructive, and only shows there is a language with large circuits for infinitely many lengths n . Let's look at the proof again. The language L is in Σ_4 and

$$C(S_n) \geq n^k$$

is true for all n . Suppose that SAT is in $\text{SIZE}(n^k)$. Then, the polynomial hierarchy collapses to Σ_2 . Thus, L is in Σ_2 and we are done.

However, suppose SAT is not in $\text{SIZE}(n^k)$. Then, we only know that for an infinite number of lengths n ,

$$C(S_n) \geq n^k.$$

So Ravi's proof does not show the existence of the language for all n . It only shows the existence for n infinitely often.

I would like to construct an explicit language L in Σ_2 so

$$C(S_n) \geq n^k$$

for all n . I cannot—yet. Ken Regan and I tried out an approach, but it did not work.

47.4 How to Compare Circuits Fast

One reason Ravi needs Σ_4 is he needs to compare circuits to see if they are equivalent. Various others have since improved this step, getting $\Sigma_3 \cap \Pi_3$ with hardness for all n , or Σ_2 with hardness for infinitely many n , but still short of what we would like. The following idea attempts to meld these improvements, and though it is basically known and gets only the $\Sigma_3 \cap \Pi_3$ result, perhaps it can be extended in a useful manner.

In any event it is a “trick” you may be able to use in other proofs, or to solve other problems. Many years ago I told this trick to Avi Wigderson—in the context of Kannan’s Theorem—and he said immediately to me:

How else could Ravi’s proof go?

Well Ravi’s original proof does not use these ideas—so obviously Avi figured out a proof on his own. Oh well.

Let $A \equiv B$ denote that the two circuits A and B compute the same Boolean function. Say a circuit with x_1, \dots, x_n inputs is an (m, n) -circuit provided only the inputs x_1, \dots, x_m are used in any gate. The other inputs are never used in a gate, and so have no effect on the output of the circuit. The key points of (m, n) -circuits are the following lemmas:

Lemma 47.3 *Let A and B be (m, n) -circuits. Then, there is an algorithm to check if $A \equiv B$ in time polynomial in n and 2^m .*

Proof Clearly, we need only check the 2^m inputs used by the circuits to determine if they compute the same Boolean function. \square

Lemma 47.4 *Let A be an (m, n) -circuit, and let B be a circuit on n inputs. If $A \equiv B$, then there is an (m, n) -circuit B^* such that $B \equiv B^*$, and*

$$C(B^*) \leq C(B).$$

Proof Let B^* be the circuit B with $n - m$ additional inputs

$$x_{m+1}, \dots, x_n$$

set to 0. It is easy to see A and B^* are equivalent. Also the complexity of B^* cannot go up after setting variables to 0. \square

If we go over Ravi's proof and only consider (m, n) -circuits with $m = ck \log n$, then equality will be checkable in polynomial time, thus saving an alternation. This will also avoid a technical lemma Ravi needs: he had to prove a special version of the classic Claude Shannon [theorem](#) that most Boolean functions have exponential complexity. Now we can use the original theorem on m variables.

One could view the restriction to (m, n) -circuits as a "padding" trick, which helps us compare circuits in polynomial time. This brings down the complexity of the whole operation to Σ_3 instead of Σ_4 . A similar method defines the complement of the language L , thus achieving $\Sigma_3 \cap \Pi_3$.

47.5 Open Problems

The main open problem I would like to see solved is the construction of hard Boolean functions in weak complexity classes. Also, I would like to see a stronger form of Ravi's theorem that would show the existence of a language with large circuit complexity in Σ_2 proved **for all** n —if this is possible.

47.6 Notes and Links

Original post:

<http://rjlipton.wordpress.com/2010/03/31/a-problem-with-proving-problems-are-hard/>

Kannan paper:

<http://www.sciencedirect.com/science/article/pii/S0019995882903825>

Karp–Lipton theorem:

http://en.wikipedia.org/wiki/Karp-Lipton_theorem

Cai paper:

<http://portal.acm.org/citation.cfm?id=1221875>

Referenced Littlewood post:

<http://rjlipton.wordpress.com/2009/09/18/why-believe-that-ntp-is-impossible/>

Referenced post for Shannon theorem:

<http://rjlipton.wordpress.com/2009/05/06/worst-case-complexity/>

Nick Howgrave-Graham and Antoine Joux are experts in the area of computational number theory and cryptography.

We will talk about their new algorithm for the knapsack problem. It is an exponential-time algorithm with a much smaller exponent than any previous algorithm. The knapsack problem is one of my favorite problems, and their improvement is clever and surprising.

Nick has done some beautiful work in computational number theory. One of my favorites is his fast algorithm for factoring numbers of the form $p^r q$, where as usual p and q are primes. He has [shown](#), with Dan Boneh and Glenn Durfee, that if r is large enough—order logarithmic in p —then such numbers can be factored in polynomial time.

Antoine has done some wonderful work on secure hash functions. For example, his team found the first collision for [SHA-0](#). In cryptography hash functions require a variety of special properties to be useful. Distinct messages with the same hash value must exist by pigeon-hole, but they should be hard to find. If collisions were easy to find, then it might be possible to “fool” someone into accepting the wrong message as the intended message. Here are two messages, found by Antoine, that collide for SHA-0:

First message (2048 bits represented in hex):

```
a766a602 b65cffe7 73bcf258 26b322b3 d01b1a97 2684ef53 3e3b4b7f 53fe3762
24c08e47 e959b2bc 3b519880 b9286568 247d110f 70f5c5e2 b4590ca3 f55f52fe
efffd4c8f e68de835 329e603c c51e7f02 545410d1 671d108d f5a4000d cf20a439
4949d72c d14fbb03 45cf3a29 5dcda89f 998f8755 2c9a58b1 bdc38483 5e477185
f96e68be bb0025d2 d2b69edf 21724198 f688b41d eb9b4913 fbe696b5 457ab399
21e1d759 1f89de84 57e8613c 6c9e3b24 2879d4d8 783b2d9c a9935ea5 26a729c0
6edfc501 37e69330 be976012 cc5dfe1c 14c4c68b d1db3ecb 24438a59 a09b5db4
35563e0d 8bdf572f 77b53065 cef31f32 dc9dbaa0 4146261e 9994bd5c d0758e3d
```

Second message:

```
a766a602 b65cffe7 73bcf258 26b322b1 d01b1ad7 2684ef51 be3b4b7f d3fe3762
a4c08e45 e959b2fc 3b519880 39286528 a47d110d 70f5c5e0 34590ce3 755f52fc
6ffd4c8d 668de875 329e603e 451e7f02 d45410d1 e71d108d f5a4000d cf20a439
4949d72c d14fbb01 45cf3a69 5dcda89d 198f8755 ac9a58b1 3dc38481 5e4771c5
796e68fe bb0025d0 52b69edd a17241d8 7688b41f 6b9b4911 7be696f5 c57ab399
a1e1d719 1f89de86 57e8613c ec9e3b26 a879d498 783b2d9e 29935ea7 a6a72980
6edfc503 37e69330 3e976010 4c5dfe5c 14c4c689 51db3ecb a4438a59 209b5db4
35563e0d 8bdf572f 77b53065 cef31f30 dc9dbae0 4146261c 1994bd5c 50758e3d
```

The exact messages are not important; the great result is that they can be found.

Their [paper](#) appeared in the 2010 EUROCRYPT conference. I like one of the traditions of this conference, which is an evening “rump” session. The session consists of short presentations that range from: “Here is my paper that you did not accept,” to “Here is my great new paper that you will accept,” to jokes. The rump session presentations are helped by a generous supply of beer and by general good company.

One year, while I was eating dinner with a large group right before the rump session, we decided we had to present a talk during the evening session—but on what? After some discussion we agreed on an idea I suggested; I did not present the idea but one of my colleagues did. The idea was a new protocol for program committees; the protocol would allow committees to have provable *deniability*. Our protocol was simple:

After the committee was done deciding which papers to accept and which to reject, they would do the following. They would randomly select one accepted paper to reject and one rejected paper to accept.

The protocol would give each program committee member the ability to handle the two most dreaded questions they get:

- “Why did you reject **my** paper?”
- “Why did you accept **their** paper?”

Each of these questions could now be answered with a smile. As far as I know the program committee does not currently use our protocol, but I am not sure.

Let’s turn to the knapsack problem.

48.1 The Problem

Recall the [knapsack problem](#) is the question of whether or not given the positive integers a_1, \dots, a_n , there is a subset I of the indices 1 to n so that

$$\sum_{i \in I} a_i = b$$

where b is another given positive integer. The obvious algorithm would try all subsets I . Since there are $2^n - 1$ non-empty subsets the algorithm takes a long time.

The earliest [algorithm](#), due to Ellis Horowitz and Sartaj Sahni in 1974, made one key observation. The above equation can be rewritten as follows:

$$\sum_{i \in I} a_i = b - \sum_{j \in J} a_j$$

where $I \subseteq \{1, \dots, n/2\}$ and $J \subseteq \{n/2 + 1, \dots, n\}$. (Yes, we assume that n is even, but if n is odd the method can easily be fixed.) Then, the algorithm computes two sets. The first consists of all the values of

$$\sum_{i \in I} a_i$$

where $I \subseteq \{1, \dots, n/2\}$; and the second consists of all values of

$$b - \sum_{j \in J} a_j$$

where $J \subseteq \{n/2 + 1, \dots, n\}$. Then simply check to see if these two have a value in common. The point is they have a value in common if and only if the knapsack problem has a solution. The punch-line is the cost for this is dominated by the number of values in each set: $2^{n/2}$. Note, checking if they have a common value can be done either by hashing or by sorting. But, in either case, this takes time just slightly larger than the sum size of the sets.

48.2 The New Algorithm

Nick and Antoine have proved the following:

Theorem 48.1 *There is a randomized algorithm for the knapsack problem that runs in time $\bar{O}(2^{0.311n})$ and uses space $\bar{O}(2^{0.256n})$.*

Note, $\bar{O}(2^{cn})$ is just the usual O-notation except the missing constant is allowed to be a missing polynomial. There is always a chance this could be abused—what if the missing polynomial is n^{1000} , but in their case it is not. See their paper for the actual bounds. Another way to see they have not “cheated” is that they have a running implementation with reasonable performance.

I think this is a terrific result, since the bound they beat is almost 30 years old. Further, one wonders whether or not there could be further improvements to the exponent on the running time. Of course, I believe $P = NP$ is possible, so even more surprising bounds could exist. In any event, it is a great result.

48.3 The Schroepel and Shamir Algorithm

How do Nick and Antoine prove their theorem? I will try to sketch the main ideas for you, but please look at their paper for the details.

Their algorithm is based on an earlier [algorithm](#) of Richard Schroepel and Adi Shamir. This algorithm uses the same method I just outlined, with a key twist. The previous algorithm created two sets S_L and S_R . The first S_L consists of all the values of

$$\sum_{i \in I} a_i$$

where $I \subseteq \{1, \dots, n/2\}$; and the second S_R consists of all values of

$$b - \sum_{j \in J} a_j$$

where $J \subseteq \{n/2 + 1, \dots, n\}$. What Schroeppe and Shamir focused on was not reducing the running time, but the storage of the algorithm. Their twist is to create a method that takes time still $2^{n/2}$, but now only uses memory $2^{n/4}$. This was a huge improvement, since even for modest size n the memory requirement of the old algorithm of $2^{n/2}$ made it impractical.

The key to reducing the memory needed is not to store all of S_L and S_R . Instead, create a subroutine that creates the elements from these sets in sorted order, and this reduces the storage required. This is achieved by viewing S_L and S_R as coming from other sets. For example, S_L is viewed as the set of all sums of S_{LL} and S_{LR} where: S_{LL} is the set of all sums from a_1 to $a_{n/4}$ and S_{LR} is the set of all sums from $a_{n/4+1}$ to $a_{n/2}$. In a like manner S_R is viewed as the set of all sums of two sets S_{RL} and S_{RR} .

This method, as I stated already, reduces the storage required to $2^{n/4}$, but leaves the running time the same. The algorithm must still look at $2^{n/2}$ values to determine whether or not the knapsack problem has a solution.

48.4 Add Hashing

The new clever insight is to add another twist to reduce the time to $2^{0.311n}$ and very slightly increase the storage requirements. Again consider finding a sum of $n/2$ elements in

$$a_1, \dots, a_n$$

that sum to b . Of course they must handle all cases, but the case where the answer uses $n/2$ elements is the case that takes the longest. Define $\mathcal{S}_{n/4}$ as the set of all sums of $n/4$ elements. There are elements σ_1 and σ_2 that lie in $\mathcal{S}_{n/4}$ so

$$\sigma_1 + \sigma_2 = b, \tag{48.1}$$

if the knapsack has a solution. Moreover, there will be **many** pairs that solve (48.1), when the problem has a solution. Therefore, the entire set of solutions to (48.1) need not be examined.

There is a general trick for searching a large space of potential solutions provided there are many solutions. Pick a hash function and a random hash value, and only search those potential solutions with the given hash value. In their algorithm they pick a number M of size about the number of solutions:

$$\binom{n/2}{n/4}.$$

Then, they pick a random R , and try to find solutions to the equations:

$$\sigma_1 \equiv R \pmod{M},$$

$$\sigma_2 \equiv b - R \pmod{M}.$$

The time to search for such a solution is bounded roughly by

$$\binom{n}{n/4} / M \approx 2^{0.311n}.$$

The details involve a clever use of the SchroeppeI–Shamir algorithm to make their hash search work correctly.

I hope this captures some of the flavor of what they do. See their well-written paper for the full details.

48.5 Open Problems

The obvious question is how low can the exponent be? Can knapsack be done in time

$$2^{o(n)}?$$

Another question is can their ideas be used to solve other related open problems?

48.6 Notes and Links

Original post:

<http://rjlipton.wordpress.com/2010/02/05/a-2010-algorithm-for-the-knapsack-problem/>

Boneh–Durfee–Howgrave–Graham paper:

http://link.springer.com/chapter/10.1007/3-540-48405-1_21#page-1

SHA-0 hash function:

http://en.wikipedia.org/wiki/SHA_hash_functions

Howgrave–Graham–Joux paper:

<http://www.joux.biz/publications/Knapsacks.pdf>

Referenced post on the knapsack problem:

<http://rjlipton.wordpress.com/2009/02/13/polynomial-vs-exponential-time/>

Horowitz–Sahni paper:

<http://portal.acm.org/citation.cfm?id=321823>

SchroeppeI–Shamir paper:

<http://portal.acm.org/citation.cfm?id=1382632>

Picture Credit:

Screen grab from Chabaud–Joux paper

<http://fchabaud.free.fr/English/Publications/sha.pdf>

Hedy Lamarr was, of course, not a mathematician nor a complexity theorist. She was one of the great movie actresses of all time, and was once “voted” the most beautiful woman in the world. She was also an inventor.

We will talk about barriers that amateurs face in working in science and technology.

Lamarr is a great example of how an amateur can both overcome and be stopped by barriers. She is the answer to the following question:

What Hollywood actress has a patent?

There may be others now, but I believe she was the first superstar actress to get a patent. She made an important contribution to technology, and faced all the problems that amateurs face when they attempt to make contributions.

Let’s look at what she did, and what barriers she faced.

49.1 Spread Spectrum

- **Lamarr knew about a real problem.** It was 1940 and World War Two had started, with Britain and Germany locked in combat. Lamarr knew that an important problem was: how can one safely control a torpedo with a radio signal? This was important, since torpedos were not very accurate and the ability to remotely control them could be immensely valuable. The story of how she knew about this problem is long, but her previous husband was an arms manufacturer. She had sat in on his corporate meetings—at his insistence—and there she apparently learned about the torpedo problem.

The difficulty in using a radio signal to control a torpedo is essentially the problem of *jamming*. If you tried to control your torpedo by a signal, eventually the enemy will find out the frequency you are using. Once this is known they could jam your control signal by putting out a strong noise signal on the given frequency.

- **Lamarr had a solution.** Her brilliant idea was to use *frequency hopping*—her invention. The transmitter on the ship and receiver in the torpedo would synchronously hop from one frequency to another. This would make jamming very hard, if not completely impossible. The jammer could try to jam all frequencies, but this would require too much equipment and power. Or the jammer could try and guess the hopping schedule used, but that would be also very difficult.
- **Lamarr found a coinventor.** The story is that at a Hollywood dinner party in 1940 she met the composer George Antheil. George had the experience of using a player piano in his work in the film industry. They agreed to work together, and used the electromechanical technology of player piano rolls to work out how to implement her hopping idea.

By 1942 they had received a patent for their invention:

Patented Aug. 11, 1942

2,292,387

UNITED STATES PATENT OFFICE

2,292,387

SECRET COMMUNICATION SYSTEM

Hedy Kiesler Markey, Los Angeles, and George Antheil, Manhattan Beach, Calif.

Application June 10, 1941, Serial No. 397,412

6 Claims. (Cl. 250—2)

This invention relates broadly to secret communication systems involving the use of carrier waves of different frequencies, and is especially useful in the remote control of dirigible craft, such as torpedoes.

An object of the invention is to provide a method of secret communication which is relatively simple and reliable in operation, but at the same time is difficult to discover or decipher.

Briefly, our system as adapted for radio control of a remote craft, employs a pair of synchronous records, one at the transmitting station and one at the receiving station, which change the tuning of the transmitting and receiving apparatus from time to time, so that without knowledge of the records an enemy would be unable to determine at what frequency a controlling impulse would be sent. Furthermore, we contemplate

Fig. 2 is a schematic diagram of the apparatus at a receiving station;

Fig. 3 is a schematic diagram illustrating a starting circuit for starting the motors at the transmitting and receiving stations simultaneously;

Fig. 4 is a plan view of a section of a record strip that may be employed;

Fig. 5 is a detail cross section through a record-responsive switching mechanism employed in the invention;

Fig. 6 is a sectional view at right angles to the view of Fig. 5 and taken substantially in the plane VI—VI of Fig. 5, but showing the record strip in a different longitudinal position; and

Fig. 7 is a diagram in plan illustrating how the course of a torpedo may be changed in accordance with the invention.

49.2 Why Did She Fail?

What happened to their invention? At the time, during the war, nothing. Some of the reasons, I believe, she did not succeed immediately are the following:

- **Other Priorities:** We were at war, and there were many other R&D projects that took up all of the US's resources—and more. These included radar, sonar, code-breaking, and the atomic bomb. Even if the Navy had wanted to support her ideas, it seems likely there were not enough resources to spare. Not enough resources to make a real effort to bring her invention to reality.
- **Not the Usual Inventor:** This is one of the fundamental barriers that anyone outside the mainstream research community faces. I can imagine the reaction to hearing that the world's most beautiful actress and a musician had invented a method to help win the war. The obvious reaction would have been, to quote basketball commentator Jeff Van Gundy:

Are you kidding me?

Jeff says this a lot, especially when he sees a player attempt a shot that they usually never make. A translation to this cry of Jeff is: stick to what you are supposed to do. The trouble is that sometimes the ball goes in, even as Jeff says, “are you kidding me?”

- **Ahead of Technology:** This is another standard barrier. Often inventions are created before the technology is ready. Spread-spectrum requires a fairly powerful digital computational ability. The technology that was available in the 1940s was very crude, and it is likely that it was essentially impossible to make her ideas work.

Lamarr's brilliant idea is used today in wireless communication—not exactly as she envisioned in her original patented work, but nevertheless in ways that are clearly traceable to her ideas. While it failed initially, Lamarr eventually got the recognition she deserved. She and her coinventor Antheil won the 1997 Electronic Frontier Foundation Pioneer Award. She also won the **BULBIE** Award, which is sometimes called the “Oscar” of inventing.

49.3 Open Problems

Are there ways to lower the barriers so amateurs can make contributions to science and technology? My discussion in Chap. 28 on Srinivasa Ramanujan raised many interesting ideas. I would like to think we can do a better job today to make sure that inventions from outside the usual circles of researchers are taken seriously.

I look forward to hearing some additional ideas on the best ways to do this.

49.4 Notes and Links

Original post:

<http://rjlipton.wordpress.com/2010/07/25/hedy-lamarr-the-inventor/>

Referenced post on amateur research:

<http://rjlipton.wordpress.com/2010/07/01/can-amateurs-solve-pnp>

BULBIE Award:

<http://www.inventionconvention.com/ICArchives/>

For more on Lamarr and other women inventors, see

[http://inventionconvention.com/
americaninventor/dec97issue/section2.html](http://inventionconvention.com/americaninventor/dec97issue/section2.html)

Picture credits:

Screenshot of patent from

<http://www.google.com/patents?vid=USPAT2292387>

Nicolas Courtois is one of the leading experts on attacks on practical crypto-systems. He has worked on trying to break, for example, various symmetric cipher systems, including AES. Since 2000, AES, the Advanced Encryption Standard, is the symmetric cipher approved by NIST. It is claimed that it will be safe for the next 200 years. Courtois maintains a webpage (referenced in the end notes) for up-to-date information on AES and related issues.

We discuss a heuristic technology that has been quite powerful in attacking crypto-systems—called *linearization*. There is a great new result that there is a class of problems that can be provably solved by this heuristic.

There are two cultures in cryptography: the practical cryptography researchers and the theoretical cryptography researchers. Let's call them the \mathcal{P} 's and \mathcal{T} 's. Both are important, both study related problems, they share techniques, and many researchers work in both areas. For example, the Siegenthaler [inequality](#) has been used by both communities—named for Thomas Siegenthaler. It relates the degree of a Boolean function and the size of its smallest non-zero Fourier coefficient.

Courtois is certainly one of the leaders in \mathcal{P} , and Sanjeev Arora is one of the leaders in \mathcal{T} . Our discussion is about a method that both groups have used in the past. However, I believe it is fair to say that the method, called linearization, has been mostly used by those in \mathcal{P} . What is exciting to report is that Arora and his student Rong Ge have a paper that shows how researchers in \mathcal{T} can add to our global understanding of this important method. Perhaps their ideas, which are interesting on their own, can help shed light on \mathcal{P} . We will see.

50.1 Linearization Method

The linearization method is based on the simple idea of converting a quadratic equation into a linear equation. For instance,

$$\sum_{i,j} a_{ij}x_i y_j = b$$

is mapped to

$$\sum_{i,j} a_{ij} V_{ij} = b.$$

There is good news and bad news here: The good news is that the new equation is linear in the variables V_{ij} ; the bad news is that the new equation has lost some information. The following is now possible,

$$V_{13} = 1, \quad V_{24} = 1, \quad V_{12} = 0.$$

If we go back to the actual variables this is impossible, since the first two equations make x_1 and x_2 non-zero, yet the last equation means that one of these must be zero. This is a linear equation, and since linear is “good” this has the potential to change a hard problem into a simple one.

The method typically takes a set of quadratic equations and converts them into a set of linear equations. Then these equations are solved by Gaussian elimination. Any solution of the quadratic equations is of course a solution of the linear ones; however, the linear equations may have many extra solutions. The method works provided the linear equations have few extra solutions.

We can try the same idea on equations of any degree d . We replace terms of degree at most d by variables V_S where S is a set of size up to d . Of course, for greater d we have many more ways to lose information. To keep the number of extra solutions down in proportion, we need many more equations. Note also that we could rewrite our higher-degree equations as quadratic by introducing new variables, but doing so might not improve the situation.

Heuristically, we can judge how many initial equations we need to have a good chance of making this work. In practice, in \mathcal{P} this works sometimes. In theory, in \mathcal{T} there are certain cases where this can be *proved* to work. That is the insight of Arora and Ge.

50.2 Linearization in Practice

The key technical insight is that AES has a nice algebraic structure. Nice algebraic structures can hurt crypto-systems, especially symmetric ciphers. Bruce Schneier and Niels Ferguson have written:

We have one criticism of AES: we don't quite trust the security . . . What concerns us the most about AES is its simple algebraic structure . . . No other block cipher we know of has such a simple algebraic representation. We have no idea whether this leads to an attack or not, but not knowing is reason enough to be skeptical about the use of AES.

One problem is that there are attacks on AES that reduce to solving many quadratic equations—apply linearization. Solving such systems of equations in general is NP-hard, and so the hope is that AES is safe. But in practical cryptography

one must look at the actual systems. In this case there are about 8000 equations with 1600 variables for the 128-bit AES.

These are apparently not enough equations to make linearization work directly on AES. An extension of linearization called the “XSL attack” was created in 2002 by Courtois and Josef Pieprzyk to try to overcome this roadblock. The XSL attack is based on linearization and some further tricks. There is an ongoing debate whether these or related methods will one day break AES.

Enter Arora and Ge. They have a pretty new paper on using linearization methods to solve a learning problem. I think this paper and their results are important. They can prove that linearization actually works on their learning problem. This is in sharp contrast to the practical work on ciphers like AES where there are no proofs yet. There are ample experiments to suggest that certain attacks will or will not work. But no proofs.

50.3 Learning Noisy Parity

Arora and Ge work on the problem of learning parities with noise. This is a classic problem, well studied, and extremely difficult. The standard model is that there is a secret Boolean vector s of some length n . There is a “box” so that when you press the “button” the following happens:

- The system generates $x \in \{0, 1\}^n$ uniformly at random and shows you x .
- The system generates a hidden “noise bit” r under a non-uniform distribution, such as $r = 1$ happening with probability 0.30.
- The system shows you the bit $b = x \cdot s + r$.

After many button presses, one gets a system of linear equations, most of which are correct. However, a substantial fraction of them are incorrect, and this makes solving the linear system very difficult.

Arora and Ge modify the problem in an interesting manner. Each time you press the button you get, say, ten x 's and ten corresponding b 's. You are promised that at most 3 are incorrect. Note this is the same on average, but it is not exactly the same as the original problem.

The surprise is that they can solve this new problem, which they call structured noise. The problem reminds me some of list decoding, and it seems to be related to that in some way.

Each time the button is pressed we get m values. The d incorrect solutions collectively satisfy a certain *noise polynomial* P . For example, they may satisfy a certain polynomial P_3 that is zero on all binary m -tuples of Hamming weight at most 3. Note that P_3 can have degree 4. The one condition on their theorem is that the correct pattern (or some correct pattern) cannot be written as the GF(2) sum of two patterns on which P is zero. This rules out arbitrary noise patterns.

In their model the original variables stand for bits s_i of the secret vector s . The new variables after the linearization have the form s_J for sets J of size up to $d + 1$. A solution to those variables is *good* provided $s_{\{i\}} = s_i$ for all i . In this model they give a randomized algorithm A and prove:

Theorem 50.1 *If there are m values and n variables, then A runs in $O(n^m)$ time and finds a good solution with high probability.*

Of course for constant m —which entails constant d —this yields a probabilistic polynomial-time algorithm.

I really like this paper, not only for proving a nice result, but for the type of result that they get. We need to prove more results of the form: the following heuristic method can be proved to work in the case of . . . One of the mysteries of complexity theory is that some heuristics work very well in practice, and we do not understand why. I think that their result helps unravel this.

50.4 Open Problems

Can we prove that linearization works on more problems? Can we prove that other heuristics also work?

Is there a connection between the beautiful work of Arora–Ge and cryptanalysis? Does the structure in AES allow an attack in which a certain proportion of “hits” are bound to be correct, and the non-hits satisfy a noise polynomial of suitably low degree? Can \mathcal{T} help \mathcal{P} ?

50.5 Notes and Links

Original post:

<http://rjlipton.wordpress.com/2010/12/13/making-a-heuristic-into-a-theorem/>

AES page:

<http://www.cryptosystem.net/aes/>

Siegenthaler inequality:

<http://www.springerlink.com/content/jxjbx75v66fvhfw4/>

XSL attack:

http://en.wikipedia.org/wiki/XSL_attack

Neal Koblitz is a strong number theorist who is famous, among other things, for inventing elliptic curve cryptography. This was independently invented by Victor Miller at the same time—1985. Neal, during his career, has also worked on critical reviews of applications of mathematics to other areas. Mathematics is a perfect fit for the physical sciences, but it is less clear how it fits with the social sciences. Neal was, for instance, involved in a potential misuse of mathematical methods to social science, in a [controversy](#) about Serge Lang and his book “The File.”

We shall discuss a recent [article](#) he wrote with Alfred Menezes for the AMS Notices entitled: *The Brave New World of Bodacious Assumptions in Cryptography*. This is not the usual math title—I have never seen a title like this in an AMS publication, or in any other publication. According to [Wikipedia](#) *bodacious* means:

- Remarkable, courageous, audacious, spirited;
- in CB radio jargon, a general-purpose word of praise;
- a variety of iris (plant);
- voluptuous, attractive, “hot,” appealing to the eye;
- Bo-Day-Shus!!!, an album by Mojo Nixon and Skid Roper.

Neal has been critical of theorists’ approach to cryptography, and has written some very critical [pieces](#). I do not agree with him, but I do think he raises some important issues.

51.1 Early Days of Cryptography

I have two background stories I wish to share—hopefully they will help explain my views on security, and why I think Neal has raised some interesting issues. It will also show how stupid I was, but I will get to this in a moment.

- **Cheating at Mental Poker:** Adi Shamir, Ronald Rivest, and Leonard Adleman (SRA) in 1979 wrote a [paper](#) on how to play “mental poker.” They described a protocol for playing two-person poker over a communication channel, without using any trusted party. One player acted as the dealer and handed out the cards, and both then played poker.

When I got the actual paper—this is way before the Internet and PDF files—I immediately realized there was a problem with their protocol. Their method used a family of maps

$$E_a : x \rightarrow x^a \pmod q$$

where q is a large prime and a was odd. The key to their method was the simple observation:

$$E_a(E_b(x)) \equiv E_b(E_a(x)) \pmod q.$$

I sent Rivest a letter outlining an attack based on getting at least one bit of information: I used the observation that their maps preserved whether a number is a quadratic residue or a non-residue. This allows the player who is “dealing” the cards to cheat and control who gets, for example, the aces.

Ron replied quickly, by return mail, with a fix: this stopped my attack, the dealer no longer could decide who got the aces. However, it still had a problem, the non-dealer could still see a few bits of information about the dealer’s hand. This is plenty of information to win at poker—in the long run. I should have challenged Ron to actually play for money, but instead wrote a [paper](#) on how to cheat at mental poker. The paper was not widely read, alas, even though it was published as part of an AMS workshop.

Shortly after this Shafi Goldwasser and Silvio Micali wrote a beautiful [paper](#) on how to use probabilistic methods to construct a protocol that avoided the problems I had pointed out. I was stupid, since I did not try to find such a provable protocol. At the time, I was working on complexity theory, and did not follow up on my crypto ideas. Too bad.

A few years ago, at a very pleasant lunch in San Francisco with Dick Karp, Shafi and Silvio pointed out to Dick how much they were influenced by my cheating paper. Even though I usually get little—no credit?—for my work, they were quite adamant how important it was to their thinking about semantic security. I thanked them.

In the SRA paper, they did not realize their protocol leaked at least one bit—more than enough to win at poker. This simple example, in the hands of Shafi and Silvio, led eventually to important new ideas in cryptography. The second story is also about how hard it is to define what is security.

51.2 Proving an OS Kernel Is Secure

In the summer of 1976, Anita Jones and I were part of a RAND summer study group organized by Stockton Gaines on security. Anita and I eventually wrote a fun [paper](#) during the summer: a paper with a long, complex, and crazy story; I will share it with you another day.

During the workshop, Gerry Popek visited us one day, and gave a talk on his then-current project to prove a certain OS kernel was secure. This was a large project, with lots of funding, lots of programmers, and he said they hoped in two years to

have a proof of the OS's correctness. What struck me during his talk was he could write down on the board, a blackboard, a relatively simple formula from set theory. The formula, he stated, captured the notion of data security: if a certain function f had this property, then he would be able to assert his OS could not leak any information. Very neat.

At the end of his talk I asked him if he wanted a proof *now* that his function f satisfied the formula. He looked at me puzzled, as did everyone else. He pointed out his f was defined by his OS, so how could I possibly prove it satisfied his formula—the f was thousands of lines of code. He added they were working hard on proving this formula, and hoped to have a full proof in the next 24 months.

I asked again would he like a proof today? Eventually, I explained: the formula he claimed captured the notion of security was a theorem of set theory—any function f had the property. Any. He said this was impossible, since his formula meant his system had no leaks. I walked to the board and wrote out a short set theory proof to back up my claim—any f had his property. The proof was not trivial, but was not hard either.

The point of the story is: the formula captured nothing at all about his OS. It was a tautology. What surprised me was his response: I thought he would be shocked. I thought he might be upset, or even embarrassed his formula was meaningless. He was not at all. Gerry just said they would have to find another formula to prove. Oh well.

51.3 Provable Security

I hope I do not lose my theory membership card, but I think Neal has made some serious points in his recent paper. But, first let's look at the foundations of *modern cryptography*. This is based on turning “lemons” into “lemonade”: turning hardness of certain computations into security of certain crypto-protocols. You probably all know the framework, but here is my view of it:

- Let \mathcal{P} be a protocol between Alice and Bob. This is nothing more than a fancy way of saying Alice and Bob have an algorithm—almost always random—they plan to execute together. The goal of the algorithm is to send a message, to share a secret, to agree on a value, to do a computation together, and in general to do some interesting information processing. Of course, Alice and Bob are using the protocol to avoid others from learning all or even parts of their information.
- Let \mathcal{H} be some computational problem we believe is “hard.” The assumption that \mathcal{H} requires a great deal of computation can vary greatly: it can mean worst-case cost, average cost, and can, today, mean the cost on a quantum computer.
- Let \mathcal{A} be the class of “attacks” allowed against the protocol. Usually these have to be explicitly stated and carefully defined. Sometimes, we allow simple attacks, sometimes we allow more powerful attacks. For example, allowing the attacker to influence Alice or Bob enables a much more powerful attack than one by an attacker who passively watches their protocol.
- Finally, the **proof of security** is a mathematical theorem of the form:

Theorem 51.1 *Any efficient attack of type \mathcal{A} against the protocol \mathcal{P} implies the problem \mathcal{H} is not hard.*

Of course, such theorems would have more quantitative bounds, but this is their general form.

51.4 Provable Security?

The framework is very pretty, but can fail in multiple ways—as Neal points out in his papers. Let’s take a look at what can go wrong. It is convenient to do this in reverse order of the parts of the framework: the theorem first, then the attacks, then the hardness assumption, and finally the protocol.

- **The Theorem:** Any theorem, especially a complex theorem, can have an incorrect proof. We have talked about this in several previous discussions. Even great mathematicians make mistakes, so it should come as no surprise if cryptographers make mistakes.

There are two reasons errors here are perhaps more likely than in some other areas of mathematics. First, there may be much less intuition about the behavior of a complex protocol. No matter how precise you are in your reasoning in any proof, intuition is a great guide. Sometimes, I have thought I had proved something, but it did not fit with my intuition. Usually, this led, after more careful thought, to the discovery of an error in the proof.

Second, as Rich DeMillo, Alan Perlis, and I have pointed out in our famous—infamous?—paper on social processes, the social network of mathematicians is a requirement for confidence in proofs. Without others checking your theorem, teaching it in their class, using it in their own work, finding new proofs of your theorem, we would be flooded with errors. The trouble with crypto theorems is they do not always have this social network to back them up. Many crypto papers prove theorems used by others, but many do not.

- **The Attacks:** This is in, in my opinion, the main issue: do the attacks include all the possible attacks? My examples on mental poker and OS kernels show two simple examples where attacks were not properly defined. There are many attacks on crypto protocols that were not envisioned initially. These include: timing attacks, fault based attacks, composition of protocols, and many others.
- **The Hardness Assumption:** There are hardness assumptions, and there are hardness assumptions. Clearly, if the \mathcal{H} problem is not as hard as you thought, then even if the theorem and attacks are correct, the theorem is meaningless.

The failure of a hardness assumption \mathcal{H} to really be hard has happened over and over in modern cryptography. Famous examples include:

- The assumption about hardness of knapsack problems;
- The assumption in the original RSA paper on the cost of factoring;
- The assumptions about solving certain Diophantine equations—especially the work of John Pollard and Claus Schnorr on [solving binary](#) quadratic forms, and Don Coppersmith on [low exponent](#) RSA.

- **The Protocol:** Finally, the protocol itself is a potential failure point. If the protocol is implemented incorrectly, then all the other parts of the framework are useless.

51.5 Neal's Main Attack

Neal's main attack is on a paper written by one of our own faculty at Georgia Tech, Sasha Boldyreva. She and her colleagues wrote a [paper](#) on a certain crypto protocol in 2007. In order to prove their protocol was secure they needed to assume a certain technical problem \mathcal{H} was intractable. They argued their assumption was reasonable: they even could prove it was hard provided the groups involved were *generic*.

I will not explain exactly what this means, but in my view it is equivalent to really restricting the attacks allowed. Roughly, if a group is generic, then there are very limited operations an attacker can perform on the group. Neal quotes them as saying:

This has become a standard way of building confidence in the hardness of computational problems in groups equipped with bilinear maps.

The problem is that the attacker can use “burning arrows.” For instance, the attacker can use the structure of the group in question, and is *not* limited to treat it as a generic group. The attackers can do whatever they want, as long as the computations they perform are efficient. This is exactly what happened two years later: Jung Hwang, Dong Lee, and Moti Yung [showed](#) Sasha's protocol could be easily broken. Neal—to hammer home his point—gives the attack in full, since it is so simple. Not simple to find, but simple since it involves only solving a few equations over the group.

The lesson here is a valid one I believe: we must be very careful in making assumptions about what an attacker can do. An equivalent way to look at this situation is: the hardness assumption was not true, if sufficiently general attacks against it are allowed. Neal then adds a comment:

The 4-page proof . . . is presented in a style that is distressingly common in the provable security literature, with cumbersome notation and turgid formalism . . .

This is unfair—I always have felt whether a proof is clear is a subjective statement. But, there is a lesson here. Just as SRA did not allow a simple attack that defeated their mental poker protocol, Sasha and her colleagues did not allow non-generic attacks against her protocol.

51.6 Open Problems

Rich DeMillo, Alan Perlis, and I, in our social process paper, were most concerned with implementations of complex software. The verification community idea, at the time, was to use mathematics to prove the correctness of such systems.

But, perhaps one way to summarize is this: the whole framework of provable cryptosystems needs the same social processes we outlined in our old paper. The exact specification of attacks, and the proof of the security of a protocol are just as vulnerable to problems as the implementation of any large software.

An open problem is to see more proofs of the completeness of attacks. I am unsure how we can do this, but I think this would be very confidence building for the field.

51.7 Notes and Links

Original post:

<http://rjlipton.wordpress.com/2010/03/13/breaking-provably-secure-systems/>

Referenced post on book by Lang:

<http://rjlipton.wordpress.com/2009/12/21/some-of-my-favorite-books/>

Koblitz–Menezes article:

<http://www.ams.org/notices/201003/rtx100300357p.pdf>

Bodacious:

<http://en.wikipedia.org/wiki/Bodacious>

Koblitz pieces:

<http://www.ams.org/notices/200708/tx070800972p.pdf>

RSA poker paper:

<http://people.csail.mit.edu/rivest/ShamirRivestAdleman-MentalPoker.pdf>

Collection with Lipton's poker paper:

<http://www.amazon.com/Cryptology-Computational-Proceedings-Symposia-Mathematics/dp/0821801554>

Goldwasser–Micali paper:

<http://portal.acm.org/citation.cfm?id=802212>

Jones–Lipton paper:

<http://portal.acm.org/citation.cfm?id=806538>

Referenced post on surprises:

<http://rjlipton.wordpress.com/2009/09/27/surprises-in-mathematics-and-theory/>

Pollard–Schnorr paper:

<http://portal.acm.org/citation.cfm?id=41075.41085>

Coppersmith paper:

<http://www.springerlink.com/content/2r2t67alntpv55w6/>

Boldyreva paper:

<http://www.cc.gatech.edu/~aboldyre/papers/bgoy.pdf>

Hwang–Lee–Yung paper:

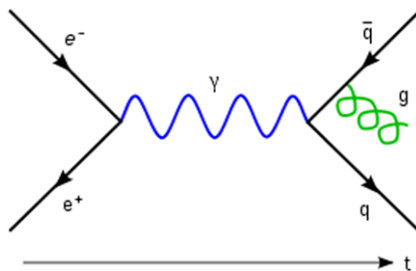
<http://portal.acm.org/citation.cfm?id=1533080>

DeMillo–Lipton–Perlis paper:

<http://portal.acm.org/citation.cfm?id=512970>

Richard Feynman was one of the greatest physicists of all time, and was probably one of the most interesting. He shared the 1965 Nobel Prize in Physics with Julian Schwinger and Sin-Itiro Tomonaga for their work on QED, which stands for quantum electrodynamics. Feynman used his now famous Feynman diagrams to understand how electrons and photons interacted, instead of using complex calculations. He is of course known to the general public for his tremendously popular series of books and textbooks—such as *Surely You're Joking, Mr. Feynman!* He could make hard concepts accessible in a way that no one else could, and there is a noted video of him playing bongo drums too.

Below is one of his diagrams.



Our drumbeat is for some strange numbers that arise in mathematics and theory. These numbers are usually connected to results that seem like miracles. I will also present a new polynomial-time algorithm whose exponent is very large—another strange result.

Such numbers occur in physics too. One such miracle number is approximately 137 and is called the *fine structure constant* in physics. In the right units the constant is

$$\alpha = \frac{e^2}{\hbar c}$$

where e is the elementary charge, \hbar is the reduced Planck's constant, and c is the speed of light in a vacuum. Feynman spoke about it this way:

Immediately you would like to know where this number for a coupling comes from: is it related to pi or perhaps to the base of natural logarithms? Nobody knows. It's one of the greatest damn mysteries of physics: a magic number that comes to us with no understanding by man. You might say the "hand of God" wrote that number, and "we don't know how He pushed his pencil." We know what kind of a dance to do experimentally to measure this number very accurately, but we don't know what kind of dance to do on the computer to make this number come out, without putting it in secretly!

Let's look at some miracle numbers.

52.1 Some Numerology

I will call a number a miracle number if there is some result attached to it that is surprising, unexpected, or just strange.

- **The number two.** Okay this is not exactly the fine structure constant. But there are many examples in mathematics where 2 is special. I guess being the lone even prime number makes you special. A simple example of this that plays an important role in complexity theory is: for any square matrix A over a field of characteristic two,

$$\det(A) = \text{perm}(A).$$

This relies on the trivial but important fact that in such a field,

$$-1 = 1.$$

- **The number one hundred and sixty three.** This number has many miracles associated with it. There is the famous polynomial, due to Leonhard Euler,

$$n^2 + n + 41,$$

which is prime for $n = 0, \dots, 39$. Note, any polynomial with a constant coefficient $c > 1$ is periodic modulo c . So having one be prime for 40 values with a constant coefficient of 41 is pretty impressive. There is a deep reason for this polynomial's behavior, which is related to a notion of Kurt Heegner. The key is that

$$163 = 4 \cdot 41 - 1$$

is a *Heegner Number*.

The number 163 also was used by the renowned Martin Gardner who claimed that

$$e^{\pi\sqrt{163}}$$

was an integer in an April Fool’s column. It is not. It is equal to

$$262,537,412,640,768,743.99999999999925\dots$$

Pretty close, but not an integer.

- **The number one thousand seven hundred and twenty nine.** The story of Godfrey Hardy and Srinivasa Ramanujan and the “[taxicab number 1,729](#)” is well known, in regard to its being the smallest number that can be written as the sum of two integer cubes in two different ways ($1^3 + 12^3 = 9^3 + 10^3$). But this is not its only notable property.

In the digits of π , a sequence of ten consecutive places that have all the digits occurs fairly quickly, at the end of

$$3.141592653589793238462643383279502884197169399375105820974944592307816$$

If you look for the first such sequence in the digits of e , however, you will not find one until . . . the 1,729th decimal place.

The number 1,729 is also remarkable as the third Carmichael number, after 561 and 1,105.

One might rebut the idea of “miracle numbers” by asserting that by looking hard enough one can find remarkable facts about any number. On the other hand, from the probabilistic viewpoint one might expect the distribution of short, easy-to-state facts to be “lumpy,” favoring some numbers while shunning others. If the number of facts worth noting is about \sqrt{N} to N where N is the largest integer under consideration, then this is like a form of the “birthday paradox.” Of course, the distribution of facts about integers is not really random—so all we can do is appreciate those “miracles” that we observe. One more such observation is what prompted this post:

- **The number fifty six.** This number arises in a relatively recent result on group theory. The following wonderful [result](#) of John Wilson was proved in his 2005 paper, “Finite Axiomatization of Finite Soluble Groups” (referenced in the end notes):

Theorem 52.1 *Let G be a finite group. Then G is soluble if and only if it satisfies the following: that no non-trivial element g is a product of 56 commutators $[x, y]$ with entries x, y conjugate to g .*

His proof uses John Thompson’s [classification](#) of the minimal simple groups. This is a deep result, but is less than the full classification theorem for simple groups. See the paper “Finite Axiomatization of Finite Soluble Groups” by Ron Solomon for Thompson’s technique, and the paper “Definability of Group Theoretic Notions” by Ari Koponen and Kerkko Luosto for related results on other properties of groups.

52.2 A Surprising Result

Wilson’s theorem allows us to prove the following surprising theorem:

Theorem 52.2 *Given a finite group, there is a uniform AC^0 algorithm that determines whether the group defines a solvable group or not.*

Recall the notion of *solvable*—some like Wilson say *soluble*. Consider a group G . The commutator subgroup G' of G is the subgroup generated by all the commutators of the group: elements of the form

$$[x, y] = xyx^{-1}y^{-1}.$$

G is solvable if the action of taking successive commutator subgroups results in the trivial subgroup in a finite number of steps.

Checking a group to see if it is solvable is clearly computable in polynomial time. Form all the commutators and build up the subgroup they generate. If this is G reject; if it is the trivial group accept. Otherwise, continue. This easily is seen to run in polynomial time, but it is not obvious how to do it in small space, let alone in AC^0 . But, as we will see, that is possible.

52.3 The First-Order Connection

The title of Wilson's paper references the consequence that solvable groups have a finite definition in first-order logic. A group can be represented as a *relational structure* (G, M) where M is the graph of the group operation: $M(x, y, z)$ means $x \cdot y = z$. The structure defines a group if and only if it *satisfies* the following axioms:

- (1) $(\forall x)(\forall y)(\exists!z) : M(x, y, z)$ (multiplication is a function and the group is closed under it).
- (2) $(\exists!e)(\forall y) : M(e, y, y) \wedge M(y, e, y)$ (identity).
- (3) $(\forall t, u, v, w, x, y, z) : (M(x, y, t) \wedge M(t, z, u) \wedge M(y, z, v) \wedge M(t, v, w)) \longrightarrow w = u$ (associativity).
- (4) $(\forall x)(\exists y) : (M(x, y, e) \wedge M(y, x, e))$ (inverses).

One can use axiom 2 to quantify the reference to e in axiom 4, or even combine all four axioms into one statement. What Wilson did is find one more axiom that is satisfied by, and only by, those groups that are solvable.

There is a well-known general theorem, by David Mix Barrington, Neil Immerman, and Howard Straubing building on work by Miklos Ajtai and others, to the effect that a language of finite relational structures is first-order definable if and only if it is in AC^0 . This would prove our result about solvable groups. However, the theorem itself does not immediately answer some important questions:

- (a) How must we represent the languages of relational structures, in order to use the theorem?
- (b) How big are the AC^0 circuits that we get—what degree polynomial bounds their size?

Regarding (a), note how cumbersome our associativity axiom is. We would rather use a functional symbol \bullet for the group operation and write the axiom simply as $(\forall x, y, z) : (x \bullet y) \bullet z = x \bullet (y \bullet z)$. For logical *meaning* this makes no difference,

but for complexity and the mechanics of proofs the representation is surprisingly important.

52.4 How to Represent the Group

A way to represent a finite group is as a multiplication table. Such a table is called a *Cayley table*, after Arthur Cayley. Here is a simple example, the table of addition modulo 4:

	0	1	2	3
0	0	1	2	3
1	1	2	3	0
2	2	3	0	1
3	3	0	1	2

A Cayley table can be written as a string t of values in row-major order, say separated by commas. Given x and y , one must look up the entry in the table corresponding to $x \cdot y$.

The main alternative is to present the group as a series of triples

$$(x, y, z)$$

where $x \cdot y = z$, basically writing out the relation M above. The key to this encoding is that an AC^0 circuit can easily *check* whether or not $x \cdot y = z$ in polynomial time and constant depth.

I believe that this encoding facilitates the construction of circuits that come from John Wilson's theorem.

52.5 The Proof

The proof that AC^0 can determine whether a group is solvable is now clear. We “simply” have to test the assertion in Wilson's theorem. Hence we build a circuit that fans out over all g in the group, and further over all sequences of 112 elements among those that are conjugate to g , and test whether the 56 commutators made from the latter multiply out to g . Since the number of conjugates may be linear in the size n of the group, we may be dealing with n^{112} such sequences. Taking in g gives order-of n^{113} cases to check. The size of the sub-circuit for each case adds 1 or 2 to the overall exponent. Thus the final algorithm is constant depth and is polynomial, but not any polynomial I would like to use.

Wilson notes in his paper that “56” might be reducible after all, and that this would follow if certain combinatorial problems arising in his proof can be solved. Independent of this, we can ask whether the degree of the polynomial size of AC^0 circuits deciding solvability can be reduced from 113 or so to a reasonable number, like 3 or 4.

52.6 Open Problems

What applications of solvability being in AC^0 can we find in complexity theory?
Can we reduce the degree of the polynomial bounding the circuit size?

What are some other great numbers with miracle results attached to them?

52.7 Notes and Links

Original post:

<http://rjlipton.wordpress.com/2010/10/19/miracle-numbers-such-as-56/>

Bongo drum video of Feynman:

<http://www.youtube.com/watch?v=qWabhnt91Uc>

Feynman biography:

http://en.wikipedia.org/wiki/Surely_You're_Joking,_Mr._Feynman!

Heegner numbers:

http://en.wikipedia.org/wiki/Heegner_number

Hardy–Ramanujan story:

<http://mathworld.wolfram.com/Hardy-RamanujanNumber.html>

John Wilson, “Finite axiomatization of finite soluble groups,” *Journal of the LMS* **74(3)**, 2006, 566–582. Available at

<http://jlm.oxfordjournals.org/content/74/3/566.abstract>

Ron Solomon, “On finite simple groups and their classification,” *Notices of the AMS* **42(2)**, February 1995, 231–239. Available at

<http://www.ams.org/notices/199502/solomon.pdf>

Ari Koponen and Kerkko Luosto, “Definability of group theoretic notions,”

<http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.34.9486>

Cayley table:

http://en.wikipedia.org/wiki/Cayley_table

Picture credits:

1. Feynman diagram:

http://en.wikipedia.org/wiki/Feynman_diagram

http://en.wikipedia.org/wiki/File:Feynmann_Diagram_Gluon_Radiation.svg

2. Cayley table: Was taken from

http://en.wikipedia.org/wiki/Cayley_table

article now has other examples.

Patrick Fischer is one of the founders of complexity theory. He played a critical role in starting the field, in proving some of the early results, and in making theory as vibrant an area as it is today. We owe him a great deal of thanks.

We will discuss two central results of complexity theory. They are old, they are important, and they have not been improved for years. I believe that the time may be right to finally make progress on them.

One of the curious features of theory, especially results concerning Turing Machines (TMs), is that their proofs often depend on the ability to program TMs in clever ways. Programming a TM well is a skill that requires a programmer to understand how linear memory—Turing Tapes—can be manipulated. Linear memory is not as easy to use as random access memory, and this is why TM programmers get paid so well. Okay, you probably can make much more money programming iPhone apps, but being a master TM programmer is hard.

I recall one conference, not FOCS or STOC, where a speaker gave a talk on TMs that did not seem correct to me, but I was not sure. Pat was also in the audience and said nothing during the talk. Immediately after the talk was over and a coffee break was in progress, Pat walked up to the speaker and said:

Son, what you said about Turing Machines is not right. I program them for a living, and you cannot do what you claim.

I have always liked the phrase “I program them for a living.” Pat has been doing that for almost half a century. His first paper that I can locate was titled “On computability by certain classes of restricted Turing machines” and appeared in the proceedings of the 1963 Fourth Annual Symposium on Switching Circuit Theory and Logical Design—the conference that became the conference that became FOCS.

Let’s turn to the two theorems.

53.1 The Two Theorems

The first theorem is in a paper by Fred Hennie and Richard Stearns in 1966 that was titled “Two-Tape Simulation of Multitape Turing Machines” (referenced in the end notes):

Theorem 53.1 *Let M be a multi-tape TM that runs in time $t(n)$. Then, there is a two-tape TM M^* that accepts the same language as M and runs in time $O(t(n) \log t(n))$.*

For the next theorem we need to define what it means for a TM to be *oblivious*. A TM is called oblivious provided for all n and for all inputs x of length n , the heads of the TM move in exactly the same way. That is the tape motions do not depend on the input x —they only depend on its length.

The second is by Nicholas Pippenger and Michael Fischer in 1979 and is titled “Relations Among Complexity Measures” (referenced in the end notes). They extended the Hennie–Stearns result to make the simulation oblivious. An aside: Michael and Pat are brothers. They probably are one of the first brothers who both worked in complexity theory.

Theorem 53.2 *Let M be a multi-tape TM that runs in time $t(n)$. Then, there is a two-tape TM M^* that accepts the same language as M and runs in time $O(t(n) \log t(n))$. Further, the machine M^* is oblivious.*

53.2 Applications

The reason I wanted to raise the two theorems together is that they really show how programming TMs is a skill that we are still learning how to do well.

The main application of the Hennie–Stearns theorem is to prove efficient deterministic-time *hierarchy theorems*. The main application of the Pippenger–Fischer theorem is to prove efficient circuit simulations of languages computable by Turing Machines.

The best-known separation theorems for nondeterministic time are much stronger than those known for deterministic time, which are based on Hennie–Stearns. The best is still, I believe, due to Joel Seiferas, Michael Fischer, and Albert Meyer in their paper, “Separating Nondeterministic Time Complexity Classes” (referenced in the end notes):

Theorem 53.3 *Suppose that $t(n)$ and $T(n)$ are time-constructible functions such that $t(n + 1) = o(T(n))$, then*

$$\text{NTIME}(t(n)) \subsetneq \text{NTIME}(T(n)).$$

53.3 Proofs of the Theorems

I will give the flavor of how these theorems are proved—as usual see the papers for the actual details.

What is interesting is that while most—many?—know the theorems, I believe that most do not know the proofs. I did know the proofs at one time, but have long forgotten the details.

Somehow the results are important, but their proofs are no longer well known. I think that this is a shame for two reasons. First, they are beautiful non-trivial theorems. Second, if we forget how they are proved, then they will never get improved. This would be too bad, since an improvement to either of the theorems would have important consequences.

The main idea both use is what we call today *amortized complexity*. Both proofs use amortization, but do not call it that. I am not a historian, but I believe that the notion of amortized complexity is usually associated with the work of Danny Sleator and Bob Tarjan in their famous work on [splay-trees](#). For example, they won the prestigious Paris Kanellakis Award in 1999 for this brilliant work.

Both theorems, Hennie–Stearns and Pippenger–Fischer, essentially have to use a kind of amortized simulation. The critical issue is this: suppose that you want to simulate multiple Turing tapes. The obvious method is to store their contents on one tape, and use your other tape for “working” storage. This is exactly what they both do.

The problem is that one linear device does not seem well suited to store two linear sequences. Suppose that you store the sequences a_1, \dots, a_m and $b_1, \dots, b_m \dots$ as one sequence

$$a_1, b_1, \dots, \underbrace{a_k, b_k}_{\Delta}, \dots, a_m, b_m.$$

Here Δ is the location of the tape head, and a_k and b_k are the contents of the two tapes that are being simulated.

The simulation is easy. Just look at the two symbols, decide what to do next, and do it. No problem. Wrong. The difficulty is, what if the simulation needs to keep one tape head stationary and move the next one to the right? Then the next state should be

$$a_1, b_2, \dots, \underbrace{a_k, b_{k+1}}_{\Delta}, \dots, a_m, b_{m+1}.$$

The trouble is that to get to this global state is very expensive: it requires the simulator to move $O(m)$ symbols. Since m can be huge, this would be a very inefficient simulation.

What to do? The details are complex, but the high-level idea is to be conservative. Why move everything? The tape heads might go back to where they were previously at the very next step, or they might continue to diverge—move further and further

apart. The clever idea is to think of the linear storage as a series of blocks that double in size. Then, as the two heads move the simulator only moves at first a block of size 2, then a block of size 4, and so on.

The insight is that there is *no* bound on the cost of any simulation step: some take constant time, and others take time 2^i where i is the size of the block that they need to move. But the total cost over the whole simulation is bounded by the “average” cost of each step times the number of steps, which is shown to be

$$O(\log t(n)) \cdot t(n).$$

Very neat. Again see the papers for the details.

53.4 Open Problems

The obvious open problems are to try and improve these classic theorems. I conjecture that it should be possible to improve them, but I have no clear idea how to proceed. I hope that you can find a trick and program TMs better than I can.

A final comment. I wonder what would have happened if the proofs, not the results, of these early TM theorems had become well known. The proofs both used the key idea of amortization that today plays such an important role in the theory of algorithms.

A related question is: are there tricks in complexity theorems that could be used in non-complexity situations? More bluntly: is there some fundamental programming trick that is used in complexity theory today that could be used in other parts of theory or even computer science in general? I wonder.

53.5 Notes and Links

Original post:

<http://rjlipton.wordpress.com/2010/01/27/programming-turing-machines-is-hard/>

Patrick Fischer, “On computability by certain classes of restricted Turing machines,” Proceedings of the Fourth Annual Symposium on Switching Circuit Theory and Logical Design, 1963, pp 23–32. Available at

<http://portal.acm.org/citation.cfm?id=1449620>

Fred Hennie and Richard Stearns, “Two-tape simulation of multitape turing machines,” *Journal of the ACM* **13(4)**, 1966, 533–546. Available at

<http://portal.acm.org/citation.cfm?id=321362>

Nicholas Pippenger and Michael Fischer, “Relations among complexity measures,” *Journal of the ACM* **26(2)**, 1979, 361–381. Available at

<http://dl.acm.org/citation.cfm?id=322138>

Joel Seiferas, Michael Fischer, and Albert Meyer, “Separating nondeterministic time complexity classes,” *Journal of the ACM* **25(1)**, 1978, 146–167. Available at <http://portal.acm.org/citation.cfm?id=322061>

For more background on the theorems, see the related 2009 posts

<http://rjlipton.wordpress.com/2009/07/28/oblivious-turing-machines-and-a-crock/>

and

<http://rjlipton.wordpress.com/2009/02/26/guessing-simulations/>

Splay trees:

http://en.wikipedia.org/wiki/Splay_tree

Roger Apéry was a mathematician who solved a beautiful problem that had been open for hundreds of years. He was not an amateur in any sense, but because his result was so surprising there was initial doubt about its correctness. Even professional mathematicians have some of the same problems that [amateurs](#) do.

We will give pointers on how to know if your proof is a proof. And further how to convince others your proof is a proof. I have no magic single insight, but I have a few suggestions that I find useful, perhaps you will too.

Apéry proved that $\zeta(3)$ is irrational. In general

$$\sum_{k=1}^{\infty} \frac{1}{k^m}$$

is the value of the famous “zeta” function $\zeta(m)$ at $m > 1$. Leonhard Euler found a closed form for $\zeta(2m)$ for any integer $m \geq 1$. This expression shows that for all integers $m \geq 1$, the value of $\zeta(2m)$ is irrational—actually it is a transcendental number. For example,

$$\sum_{k=1}^{\infty} \frac{1}{k^2} = \frac{\pi^2}{6}.$$

Before Apéry’s work the status of $\zeta(3)$ was unknown. Then in June 1978 he gave a talk, during which he presented a proof that $\zeta(3)$ was indeed an irrational number.

54.1 Apéry’s Proof

Ron Graham told me a story about the talk. It seems likely Apéry had a proof, but his presentation style and some of his comments—jokes?—did not make the audience feel confident about his claim.

Apéry's proof depended on the following identity

$$\zeta(3) = \frac{5}{2} \sum_{n=1}^{\infty} \frac{(1)^{n-1}}{n^3 \binom{2n}{n}}.$$

At the time no one had seen such an identity. It was new. And it seemed magical—even the factor $5/2$ is strange. Apparently someone in the audience asked Apéry how did he find this identity? Apéry answered,

They grow in my garden.

You can imagine the effect such a comment might have on a group of skeptical mathematicians. Suppose he had said instead:

The identity follows by using shifted Legendre polynomials. I can prove that all . . .

Then it would have been okay. But “they grow in my garden” caused, according to Ron, many to start to leave the talk. However, someone in the back of the room had a programmable HP calculator, and quickly wrote a tiny program to check the identity. They interrupted the talk and said that the identity checked to the allowed accuracy of the calculator. Ron says this got everyone's attention, and people sat back down and listened to the rest of Apéry's talk.

In the years after the talk other mathematicians found proofs that were based on Apéry's work. They were done and explained in a more conventional manner, and $\zeta(3)$ is indeed irrational. See Wikipedia's [item](#) “Apéry's theorem: Later proofs” for more discussion on the state of further work.

54.2 Some Suggestions

I have a small collection of suggestions on how to know your *proof is a proof*. The same suggestions can be used to convince others, so I will concentrate on convincing oneself. In my opinion the *last* way to convince oneself or anyone else is to use formal methods. I have seen some comments in our discussions on using this or that formal system. I think that they have a role in mathematics, but not in the checking of a new result. The old paper of mine with Richard DeMillo and Alan Perlis on the nature of proof, titled “Social processes and proofs of theorems and programs” and mentioned in Chap. 51, has my full thoughts on this topic.

- **Calculate:** Many of the great mathematicians were tireless calculators. Euler, Carl Gauss, Bernhard Riemann, and many others did extensive calculations of all kinds. It is even more remarkable they did this, when they did not even have calculators—let alone computers. Trying small examples, checking identities to high precision, and building tables is one of the ways to get a feel for the problem at hand.

I am suggesting that this be used to check some of your “lemmas” and claims. This approach does require a caution. Nature, especially in number theory, often will not reveal her secrets in small cases. In a post from 2009 (referenced in the end

notes), I discussed a classic theorem of John Littlewood that showed that calculations can be misleading. But calculations can *disprove* a claim, even if they cannot prove a claim.

I have more than once thought that a false X was true and tried to prove it. In one case, after trying quite a while to prove such an X , I asked the late Philippe Flajolet for help. Philippe was a world expert on such problems. He used a combination of a technical insight I did not know and Maple to show why I could not prove X . The simple difficulty was there is a counter-example to X .

- **Check Concrete Cases:** A related idea to calculation is to check all lemmas and claims in small cases. For example, suppose you have “proved”:

Lemma 54.1 *For each integer $d \geq 1$, the Euclidean space \mathbb{R}^d has the red property.*

I would suggest you check that \mathbb{R}^1 and \mathbb{R}^2 actually have this property. If you can show that the line and the plane have the red property, that will help. Even better is if your argument in these special cases is different from the general argument used to prove the lemma.

Also in a previous post (referenced in the end notes), I discussed work I did on the *Vector Addition Problem*. One false claimed proof that the problem was decidable had a lemma like the above. The lemma stated some property of Euclidean space \mathbb{R}^d . The difficulty was that the property failed even for the one-dimensional case. My geometric intuition is terrible, but the authors should have checked that their lemma at least worked on the line. It did not—there was a simple counterexample.

- **Watch Case Arguments:** The easiest error to make in any proof is to forget a case. This is especially true when there are many cases, or when the cases have sub-cases and so on. In geometry, for example, there are great paradoxes based on “proofs” that leave out a case.

A related issue is check any argument that handles a case X by saying: “the proof follows in the same way as case Y .” This type of argument often relies on a symmetry between X and Y . But if the symmetry is missing, then X has not been proved. This happened years ago in a claimed proof that SAT needed superpolynomial-size circuits. The claim was made by a world expert, and the bug was exactly of this type.

- **Know the Obstacles:** If you have proved something that others could not, you should know the barriers they faced. Further you should be prepared to explain how you overcame those barriers.

Imagine that Grigori Perelman had claimed the Poincaré Conjecture based on the Ricci Flow methods of Richard Hamilton, and when asked how he handled the “cigar” singularity, had said:

The what singularity?

That would have raised lots of doubt about Perelman’s work. If you solve a problem, even a much more modest problem than Poincaré, you should know the difficulties that others have faced. And be prepared to explain how you overcame them, went around them, or through them—in general how you avoided them.

- **What is Your New Idea?** I believe that anyone who solves an open problem needs to have some new idea. An idea that was missed by previous researchers. The idea can be small, but it is essential.

If someone says here is how I proved the Riemann Hypothesis: “it is a careful induction . . .” I am pretty pessimistic. If instead they say:

Say a set of complex numbers is a k -set if it can be covered by k lines. I eventually show that the zeroes of the zeta function are a 2 set. First I show they are a k -set for some finite k . I noticed if this was false, then by . . .

This sounds like it has a new idea, and while I need to see the details, I will listen. Even if the proof fails, there is some chance that the new ideas could be useful in solving other problems.

- **Don’t Prove Too Much:** This is one of my favorite tools when I am searching for a proof. I have not heard it discussed before, but perhaps it is known.

Suppose I am trying to prove some statement X . I may see an outline that looks like it will eventually lead to a proof of X . Often there are many details that have to be checked and lots of work to be done. However, sometimes I can see that if my method can prove X it could also prove another statement Y . This is a kind of “meta” argument, but I often find if a method proves X it will prove a related Y .

The key is then what is Y ? If the statement Y is known to be false, then my method must fail. In this case I have avoided a great deal of work, and can use my time to refine my approach to X . If the statement Y is open, but considered likely to be false, then I may still continue. But now I am much more careful.

Another possibility is that the statement Y is a major open problem. Of course if my method will work this is great news, but usually I view it as showing my method is suspect. This does not make me abandon my approach; it does again make me very careful about each step. Almost always I find that I have a hidden issue in my approach in this situation.

- **Explain It to Others:** One way to see if your argument is reasonable is to try to explain it to others. The best situation is to try to do this to an interested and expert audience. This is not always possible. I have found, however, that even explaining your ideas to non-experts is quite useful. More than once in the middle of a detailed explanation of a “proof” I realized the “bug” myself. I think forcing one to say all the details forces you to see all the nooks and crannies where your proof could go wrong. It is a bit like reading text out loud to better proof-read it.
- **Work With Others:** Yes Perelman worked alone. But even Andrew Wiles needed the [help](#) of Richard Taylor to finish his great proof of Fermat’s Last Theorem. Wiles rightly deserves the credit, but there are two papers that were needed to resolve this famous problem.

Most of the false claims of famous open problems—I believe—have been singly-authored papers. Perhaps that is a lesson for us all.

Harry S. Truman once said:

It’s amazing what you can accomplish if you do not care who gets the credit.

54.3 Open Problems

Did these suggestions help? Do you have some additional ideas to help check proofs?

54.4 Notes and Links

Original post:

<http://rjlipton.wordpress.com/2010/07/11/proving-a-proof-is-a-proof/>

Referenced post on research by amateurs:

<http://rjlipton.wordpress.com/2010/07/01/can-amateurs-solve-pnp/>

Later proofs of Apéry's theorem:

http://en.wikipedia.org/wiki/Apery's_theorem#Later_proofs

DeMillo–Lipton–Perlis paper:

<http://portal.acm.org/citation.cfm?id=512970>

Referenced post on Littlewood:

<http://rjlipton.wordpress.com/2009/09/18/why-believe-that-pnp-is-impossible/>

Referenced post on the Vector Addition Problem:

<http://rjlipton.wordpress.com/2009/04/08/an-expspace-lower-bound/>

For more on proof ideas, see

<http://rjlipton.wordpress.com/2009/02/16/proofs-and-elevator-rides/>

Wiles' proof of Fermat's Last Theorem:

http://en.wikipedia.org/wiki/Fermat's_Last_Theorem

Ron Rivest is one of the most famous computer scientists in the world. While he has done many important things in many aspects of computer theory, he is best known as the “R” in RSA. This is the renowned public-key cryptosystem based on the hardness of factoring that was created by Ron and Adi Shamir and Leonard Adleman. They won the 2002 ACM Turing Award for this terrific achievement.

We have some gifts to exchange with you. These are simple mathematical facts and items that are fun.

One of the “gifts” is from Ron, and the others are from various places and people. I wrote this originally in the blog toward the end of the year, close to the winter solstice. Various cultures also have holidays, festivals, and rituals that celebrate this time. Of course it is the time also for gifts of all kinds, since many holidays involve gifts or celebrations: Chinese New Year, Christmas, Hanukkah, Kwanzaa, New Year, St. Lucia’s Day, and others.

In this spirit I hope you enjoy a small number of fun items that I view as gifts from mathematics to all of us.






55.1 Gift I

Daniel Kirsch’s gift.

One of the coolest sites I have seen in years is [Detexify²](#). Go to the site and then draw with your mouse any symbol at all, anything at all, and magically the site lists a number of LaTeX commands that it thinks are what you want. I drew this



and got this:

	Score: 0.0772683748856403 <code>\phi</code> mathmode
	Score: 0.0924813117961339 <code>\o</code> textmode
	Score: 0.094272194291775 <code>\usepackage{ wasysym }</code> <code>\diameter</code> textmode & mathmode
	Score: 0.0971273357562433 <code>\emptyset</code> mathmode
	Score: 0.102276017179725 <code>\usepackage{ amssymb }</code> <code>\varnothing</code> mathmode

Very cool. Does everyone know about this great site—am I the last to discover it? Ken adds, however, that it is not foolproof. His best examples of symbols that are not covered—meaning they are not in standard L^AT_EX either—are semicircular wedges, like a half-moon. With the round part facing up, it is the symbol used in Chess Informant [pictographs](#) to suggest a better move than the one that was played.

(Let us not forget that T_EX itself was a gift. And L^AT_EX and AMS-T_EX and all the rest too. That is on a scale beyond the margins of this chapter.)

55.2 Gift II

Ron's gift.

Ron had a brilliant idea that helped to solve an important open problem called the Aanderaa–Karp–Rosenberg [Conjecture](#). It is still not completely resolved, but I want to highlight his idea—actually the work was joint with Jean Vuillemin. Suppose that $f(x_1, \dots, x_n)$ is a Boolean function. There are many ways to define the complexity of such a function:

- (1) the size of its smallest formula;
- (2) the size of its smallest general circuit;
- (3) the size of its smallest circuit of a given constant depth;
- (4) and so on.

Another interesting measure is to consider is based on [decision trees](#) that compute the Boolean function. A decision tree starts at a root and at each node branches left or right based on the value of a single variable x_i . At a leaf the tree decides whether to accept or reject: each leaf is labelled with 0 or 1.

The *depth* of the decision tree is the length of the longest path in the tree from the root to a leaf. This is the measure that Ron's idea works on—note the size of the tree is also quite interesting, but that is for another day.

Call a Boolean function $f(x)$ *elusive* provided any decision tree for the function, of any size, has some path of depth n . Since it is unnecessary to evaluate a variable more than once on any path, an elusive function has the maximum depth possible.

The natural question is how does one prove that various functions are elusive? In general this can be quite hard, but Ron's beautiful idea resolves it for many functions.

Theorem 55.1 *Let $f(x)$ be a Boolean function. Suppose that the number of x such that $f(x) = 1$ is odd. Then the function $f(x)$ is elusive.*

The proof is really wonderful. Consider a decision tree for $f(x)$ that is not elusive. For a leaf l define $N(l)$ to be the number of x 's so that on that input to the tree the path taken goes to the leaf l . There are two simple observations. Since the tree is not elusive, all paths are less than n , so $N(l)$ is always even. This follows because some x_i was not examined on the path to the leaf l and so the number of inputs that go to this leaf is even.

Next we note that the following identity is true:

$$A = \sum_{l \text{ is an accept leaf}} N(l)$$

is the total number of inputs that make the Boolean function 1. We now have a contradiction, since A is even, but by assumption it is odd.

This is one of the most elegant counting arguments, in my opinion. Perhaps in each case there is an alternative proof, perhaps one based on an adversary. But this counting is so nice. As a simple example consider the majority function on 4 inputs: it is 1 when there are two or more 1's in its input. This happens in 9 ways, so the function is elusive.

55.3 Gift III

Bettina Richmond's gift.

Imagine that you are given a black box \blacksquare . It has a slot where you can put any real number. After you place one, it makes some sound, vibrates a bit, and returns a value. Inside the box is a secret polynomial $p(x)$: if you put in x , you get out $p(x)$.

Your job is to find the exact coefficients of the polynomial. To make it especially hard, you are only allowed to put in two numbers—one after the other. So far the problem is impossible: it is easy to show that if the black box \blacksquare says 0 each time,

you still have no way to get the whole polynomial. Hence we will restrict the polynomial in a simple way: all the coefficients of the polynomial are natural numbers:

$$0, 1, 2, 3, \dots$$

The claim now is the problem is solvable: only two numbers need be evaluated by the black box ■.

The proof of this is quite neat. Ask the black box ■ for the value $R = p(1)$. Then ask it for the value at $R + 1$. A moment's thought will show that this essentially yields the coefficients of the polynomial: one just converts the answer to radix $R + 1$ and reads off the coefficients. Note it works precisely because the coefficients are all non-negative natural numbers.

This is from her [article](#) called “On a Perplexing Polynomial Puzzle.” By the way I have seen this puzzle elsewhere on the Web, and some sites ask money for the solution. No regrets here—it’s a gift.

55.4 Gift IV

Pythagoras’ gift.

Well not quite. There is an interesting [thread](#) on MathOverflow whether or not there is a proof that $\sqrt{2}$ is not rational that does not use “proof by contradiction.” See also Tim Gowers’ interesting [discussion](#) on this and more.

Our goal is to try and get such a proof. A proof that $\sqrt{2}$ is not rational—without using proof by contradiction. Suppose we examine the polynomial

$$f(x, y) = 2 \cdot x^2 - y^2.$$

We will show that this polynomial over the natural numbers is only zero if both x and y are 0.

The proof is based on the magic number 3, not 2, and is unfortunately a series of cases. Well it’s a gift—if you do not like it, then return it.

I need two basic facts about integers:

- (1) Any integer is congruent to one of 0, 1, 2 modulo 3.
- (2) Any non-zero integer is of the form $3^k w$ where $k \geq 0$ is a natural number and w is a non-zero integer that is not divisible by 3.

I claim these both are provable by induction. To prove (1) use simple induction. It is clearly true for 0. Suppose that $n > 0$. Then $n - 1$ is congruent to one of 0, 1, 2. By case analysis we can show the same is true for n . For example, if $n - 1 \equiv 2 \pmod{3}$, then $n \equiv 0 \pmod{3}$. To prove (2) use complete induction and prove it for positive integers, which is enough. It is clearly true for 1. Suppose that $n > 1$. If n is not divisible by 3, then the statement is true. If 3 divides n , then $n = 3m$, and the proof follows again by induction.

Let’s start the proof. Fix x and y , not both zero, and let $z = f(x, y)$. Our goal is to show z is not zero.

- Suppose $x = 0$ and $y \neq 0$. Then z is clearly non-zero.

- Suppose $x \neq 0$ and $y = 0$. Then z is clearly non-zero.
So we can assume that both x and y are not zero. So far no proof by contradiction—right? Now on to the more interesting cases.
- Suppose that both x and y are not divisible by 3. Then z modulo 3 is $2 \cdot 1 - 1$ which is not zero. Thus z is not zero. This uses (1) and the fact that $1^2 \equiv 2^2 \equiv 1 \pmod{3}$.
The above really is the main part of the proof. The other cases have to do with one or both of x and y being divisible by some power of 3.
- Suppose that $x = 3^k \cdot a$ and $y = 3^\ell \cdot b$ by (2) above. Then,

$$z = 2 \cdot 3^{2k} \cdot a^2 - 3^{2\ell} \cdot b^2.$$

There are three cases. If $k = \ell$ then z is equal to $3^{2k} \cdot (2 \cdot a^2 - b^2)$. Thus $z/3^{2k}$ is an integer which is congruent modulo 3 to $2 \cdot 1 - 1$. This shows that z is non-zero.

So assume that $k > \ell$. Then z is equal to

$$3^{2\ell} \cdot (2 \cdot 3^m a^2 - b^2),$$

where $m = 2k - 2\ell > 0$. Then $z/3^{2\ell}$ is an integer congruent to -1 modulo 3. This again shows that z is non-zero.

The last sub-case is $k < \ell$. Then z is equal to

$$3^{2k} \cdot (2 \cdot a^2 - 3^m \cdot b^2),$$

where $m = 2\ell - 2k > 0$. Again $z/3^{2k}$ is an integer congruent to 2 modulo 3. This again shows that z is non-zero.

55.5 Open Problems

Is the proof of irrationality of $\sqrt{2}$ without contradiction? Can it be simplified? Are there other gifts that you would like to share this time of year?

55.6 Notes and Links

Original post:

<http://rjlipton.wordpress.com/2010/12/20/some-mathematical-gifts/>

RSA cryptosystem:

[http://en.wikipedia.org/wiki/RSA_\(algorithm\)](http://en.wikipedia.org/wiki/RSA_(algorithm))

Winter solstice:

http://en.wikipedia.org/wiki/Winter_solstice

Daniel Kirsch site:

<http://detexify.kirelabs.org/classify.html>

Chess Informant pictographs:

<http://www.chessinformant.rs/system-of-signs/>

Aanderaa–Karp–Rosenberg conjecture:

http://en.wikipedia.org/wiki/Aanderaa-Karp-Rosenberg_conjecture

Wikipedia on decision trees:

http://en.wikipedia.org/wiki/Decision_tree

Bettina Richmond article:

<http://www.ingentaconnect.com/content/maa/cmj/2010/00000041/00000005/art00011>

MathOverflow discussion on $\sqrt{2}$:

<http://mathoverflow.net/questions/32011/direct-proof-of-irrationality>

Discussion on Tim Gowers' weblog:

<http://gowers.wordpress.com/2010/03/28/when-is-proof-by-contradiction-necessary/>

Picture credits:

1. Dick's drawing of emptyset

2. Feedback from

<http://detexify.kirelabs.org/classify.html>

Frank Ryan is not just a theorist, but is also a mathematician who specialized in complex analysis. He earned his PhD from Rice University on “A Characterization of the Set of Asymptotic Values of a Function Holomorphic in the Unit Disc,” in 1965.

Let’s hear about how we learn, and how we teach.

I have a story to tell about Ryan—I had just shared it with Alan Kay—and he insisted I had to tell it. Right away. So here it is Alan.

Ryan was not only a professor at Case Western Reserve, where I was an undergraduate in the 1960s, but he was at the same time the starting quarterback for the Cleveland Browns. The starting quarterback for an NFL football team, and a professor with a full teaching load. During the football season he taught his class early in the week, and then on Sundays he was behind the center taking the ball. Handing it off, throwing passes, and getting sacked—like any other quarterback in the league. He was one of the best quarterbacks of his day, and had great successes. For example, he [appeared](#) in three straight Pro Bowls.

I still remember listening to him explain a fine point in complex analysis on Tuesday, and then watching him on TV getting tackled, on Sunday. It was hard to believe, even though I knew it was the same person, the player being taken hard to the ground was my professor. The player being tackled knew how to throw a perfect spiral 40 yards, and also could go to the board and prove Picard’s Little Theorem. Amazing.

Today, I believe there is no chance a quarterback—or any player—on an NFL team would want to or be allowed to be a full-time professor during the season. The game has gotten very technical, the pay is too great, and the stakes are too high for any team to allow this to happen. But back when I was taking complex analysis it happened. Really.

I still recall wincing when he got sacked during one tough game. Then, a few days later I saw him in class, with his arm in a sling and his speech slightly slurred. I assumed the slurring came from pain killers he was taking for the shoulder injury. Sling or not we pushed on, deeper into the beautiful structure of complex analysis.

56.1 Ryan's Seminar

As an undergraduate I took a seminar with Ryan on complex analysis. This was one of the strangest classes I ever had in mathematics, and probably one of the best. It was a small group, about eight of us, taking his class on advanced topics in complex analysis. The course was based on a thin monograph, but Ryan did zero lecturing. Instead, at the beginning of each class he ran the following protocol:

- He would shuffle up a deck of playing cards, and we all would gather around a table.
- He then would deal out the cards one at a time face up on the table: we each got the cards landing in front of us.
- There were two **bad** cards: the Queen of Spades and the Queen of Diamonds. Once these two cards were dealt, the class really began. The player who was “lucky” enough to get the Queen of Spades went to the blackboard and was expected to start explaining from exactly where we stopped at the end of the last class. You were allowed to use the book or notes, and you typically explained the proof of some theorem.

After half the class was over, it was the turn of the other “lucky” person—the one who got the Queen of Diamonds—to take over from the first student.

Sounds not too hard, but it was a killer. The main problem was that the thin book's concept of a proof was not a detailed proof, but at best a high-level sketch. Proofs were filled with phrases like: “it is easy to see that $f(z)$ is continuous,” or “it clearly follows that $g(z)$ is never 0 in the unit disk.” The person at the board would say these phrases, but Ryan would usually jump in with a simple “why?” Why indeed was $f(z)$ continuous? Why indeed was $g(z)$ never 0?

Sometimes the student at the board could answer and we moved on to the next point, but often he got stuck. The rest of us could help and make suggestions—of course we were usually lost too. The class might stay on a single question for the entire first half of class. If this happened, then the next student would have to get up and try to convince Ryan why it was true.

The student who was up second had an interesting prediction problem. He had 45 minutes to prepare for his turn, but he had no idea where the first student would get to in his 45 minutes. I remember being in this position—half listening to the class while trying to guess where I would have to start explaining.

The cards, the Queen of Spades and the Queen of Diamonds, were picked as the “bad” cards for a reason. The first is, of course, the worst card to get in the game of [hearts](#): the player who is stuck with this card gets 13 points. The second is based on the original movie “The Manchurian Candidate.” In the movie this card is used to trigger Laurence Harvey to follow orders without question. This is one of the great movies, in my opinion.

56.2 Learning Methods

I sometimes wonder how we learn and how we should teach. I once wrote about the possibility of an “Educational Extinction Event” for universities, based on the rise of online classes and campus costs.

Despite my fascination with the teacher, I hated Ryan’s course itself. One consequence of the way the class was organized was that I learned relatively little of the material. In a more conventional class I think I would have learned more theorems, more proofs, more concrete facts from complex analysis.

However, I loved Ryan’s classes. They taught me to think on my feet—literally. I learned how to read a proof and find the “gaps” I needed to fill in myself. I may have learned relatively little complex analysis, but I learned a great deal about mathematics in general.

By the way, the above musings about phrases in proofs came from the example of Charles Picard’s Little Theorem:

Theorem 56.1 *Suppose $f(z)$ is an entire function. Then the range of $f(z)$ is either the whole complex plane or the plane minus a single point.*

The function $f(z) = \exp(z)$ shows that the theorem statement is the best possible. How would you go about interesting a class in the craft of proving this?

56.3 Open Problems

The main open problem is this: what is the best way to present mathematical material? I am especially interested in hearing what you think about Ryan’s method. Did you have some similar experience? Should we teach more classes in this way? Or is it better to cover lots of material?

56.4 Notes and Links

Original post:

<http://rjlipton.wordpress.com/2010/03/18/the-quarterback-and-the-professor/>

Frank Ryan:

[http://en.wikipedia.org/wiki/Frank_Ryan_\(American_football\)](http://en.wikipedia.org/wiki/Frank_Ryan_(American_football))

Referenced for “educational extinction event”:

<http://rjlipton.wordpress.com/2010/01/29/an-educational-extinction-event/>

Leonard Schulman was the program chair of the [STOC 2010](#) conference. He is of course an expert in many areas of computational complexity. He has made important contributions to areas such as game theory, quantum theory, and *theory* in general.

We thank Len for doing such a great job with that year's program, and also the rest of the committee: Timothy Chan, Ken Clarkson, Constantinos Daskalakis, Irit Dinur, Faith Ellen, Alan Frieze, Parikshit Gopalan, Piotr Indyk, Valentine Kabanets, Yael Kalai, Howard Karloff, Robert Kleinberg, Assaf Naor, Noam Nisan, Chris Peikert, Jaikumar Radhakrishnan, Oded Regev, Alexander Russell, Aravind Srinivasan, Santosh Vempala, and Andrew Yao. I enjoyed seeing the talks, and seeing everyone in Boston.

I also thank the committee for not rejecting one of my papers that year. Since I did not submit one, it was extremely unlikely that I would get rejected. But thanks anyway. I once had a nightmare: what if some people got rejection emails by mistake, when they had not submitted papers?

Len was at Georgia Tech years ago. Actually he left Tech to go to Caltech, just as I arrived. I do not believe these events were linked; I certainly would have loved to still have Len here at Tech—not Caltech.

Len did a very clever piece of work in 1999 with Sridhar Rajagopalan, which I wish to talk about. What I like so much about their work is it shows how surprising upper bounds can be. It also shows how clever Len and Sridhar can be.

57.1 Multiplication Tables

Suppose I present you with a simple n by n table of some “multiplication” type operator, let \circ be the operator. More precisely, there is a set S of n elements and a binary operation

$$\circ : S \times S \rightarrow S.$$

For any two elements x and y in S , their “product” is denoted by $x \circ y$.

Given such a table there are many natural questions you might ask:

- (1) Is it the multiplication table for an associative operator?
- (2) Is it the multiplication table for a group?
- (3) Is it the multiplication table for an Abelian group?
- (4) Is it the multiplication table for a group with some property X?

For example question (3) can be solved in time $O(n^2)$ by brute force: try all the pairs x, y and see if

$$x \circ y = y \circ x.$$

Years ago Zeke Zalcstein and I worked on such problems. In particular we showed, in 1978, that there was a constant-time randomized algorithm for checking if a group was Abelian. Much more recently Bin Fu [proved](#) both upper and lower bounds for this problem. He proved an $\frac{n^2}{3}$ lower bound for deterministic algorithms, and re-proved our old result.

Zeke and I worked hard on the first question, to check whether an operator was associative and we could do no better than the obvious brute force $O(n^3)$ time. That is, we could not beat the algorithm: try all x, y, z and check

$$x \circ (y \circ z) = (x \circ y) \circ z.$$

We never tried to prove a lower bound, but it seemed “clear” that you would have to somehow try all the possible triples. Of course this is no proof, and is completely wrong.

57.2 Testing Associativity

Len and Sridhar’s [solution](#) is quite clever in my opinion. They proved:

Theorem 57.1 *There is a randomized algorithm that runs in time $O(n^2 \log \frac{1}{\delta})$ to check whether or not an n by n table defines an associative operator.*

In the above statement, δ is the error probability of the algorithm. I will sketch their proof—as always please look at their well-written paper for all the details and some additional results.

The key insight is to look at the “algebra” defined by the operation. The elements of this algebra \mathcal{S} are formal sums of the form

$$\sum_{s \in S} a_s s,$$

where a_s lies in the Galois Field of two elements. They point out that if the operation \circ defined a group this would be called the “group algebra.” There are three natural operations defined on these elements:

- (1) Addition: $\sum_s a_s s + \sum_s b_s s = \sum_s (a_s + b_s) s.$

(2) Scalar multiplication: $b \sum_s a_s s = \sum_s b a_s s$.

(3) The operation \circ : $(\sum_r a_r r) \circ (\sum_s b_s s) = \sum_r \sum_s a_r b_s (r \circ s)$.

These operations can be done efficiently. The key is that the last operation can be done in time n^2 by a brute force method. Just form all the products and collect like terms.

It is not hard to show that the table is associative if and only if the operation \circ on \mathcal{S} is also associative. Their algorithm is just this: select three random elements \mathbf{a} , \mathbf{b} , \mathbf{c} from \mathcal{S} and check that

$$\mathbf{a} \circ (\mathbf{b} \circ \mathbf{c}) = (\mathbf{a} \circ \mathbf{b}) \circ \mathbf{c}.$$

The last step is the lemma:

Lemma 57.2 *Suppose the operation \circ is not associative. Then with probability at least $1/8$ the above random test will fail.*

The point is that in blowing up the domain from S to \mathcal{S} , now any failure of associativity over S extends to at least $|\mathcal{S}|^3/8$ triples from \mathcal{S} . In a sense their associative testing algorithm is using another amplification trick. Very neat.

57.3 Open Problems

Can their method be used to get other properties of tables in less than the obvious time? The “group algebra” type approach seems quite powerful. Can it be used to solve other problems?

57.4 Notes and Links

Original post:

<http://rjlipton.wordpress.com/2010/06/03/an-amplification-trick-and-stoc-2010/>

STOC 2010 page:

<http://research.microsoft.com/en-us/um/newengland/events/stoc2010/>

Paper by Bin Fu:

<http://www.spclab.com/publisher/journals/Vol3No1/BinFu.pdf>

Paper by Rajagopalan and Schulman:

<http://www.cs.caltech.edu/~schulman/Papers/verify.pdf>

Paul Seymour is one of the greatest graph theorists in the world, who is perhaps best known for famous work with Neil Robertson on [graph minors](#). In addition, Paul has worked on and solved numerous other famous graph theory problems.

We discuss a recent result on graph minors by Martin Grohe—who builds on Robertson–Seymour’s work. I also want to discuss a bit why the proofs of these theorems are so long.

I almost wrote a paper with Paul Seymour once—somehow we never got past some initial discussions. Oh well.

The problem Paul and I considered was a game played on an undirected graph G by two players: we called them *breaker* and *maker*. Let $P(G)$ be any polynomial-time computable monotone graph property. For example, $P(G)$ could be: the graph G is connected, the graph has two edge disjoint spanning trees, or the graph has two disjoint paths from a special vertex s to another special vertex t .

We assume $P(G)$ is true. The game is played in two rounds:

- In round one, *breaker* removes any l edges from G .
- In round two, *maker* can add any k edges to G —maker is not restricted to just replacing removed edges.

Maker wins if the final graph has the property $P(G)$, and *breaker* wins if it does not. Of course we assumed $k < l$: otherwise, the game would be trivial, since *maker* would just replace all the deleted edges.

As a simple example, let the graph be a cycle and the property be the graph is connected. Clearly, if *breaker* can remove 2 edges, *maker* wins if he can add 1 edge; if *breaker* can remove 3 edges, then he wins if *maker* can only add 1 edge.

Paul and I, really Paul, could show there were polynomial-time algorithms to decide who won this game for a variety of simple properties and for small values of k, l . He is beyond impressive for understanding questions like these—I played a pretty small role.

We were interested in the game, since we thought of “breaker” as destroying part of a telecom network, for example, and “maker” then had to quickly repair edges to restore a property. I still like the general class of these problems, but we never did write down the details. Let me know if this is all known today.

58.1 The Result of Grohe

The central idea Martin's work is based on is the notion of [graph minor](#). A graph H is a **minor** of G provided H is isomorphic to a subgraph of G obtained by

- (1) contracting edges,
- (2) deleting edges, and/or
- (3) deleting isolated vertices.

The concept of minor has some beautiful properties—many have been discovered and then proved by Robertson and Seymour. For example,

Theorem 58.1 *For a fixed graph H , there is an algorithm with running time $O(n^3)$ for testing whether or not H is a minor of a given graph with n vertices.*

The algorithm's running time is cubic, but the constant in the O -notation depends super-exponentially on the size of the graph H . Note, this must be true: if it were only polynomial in the size of H , then one could solve the Hamiltonian Cycle problem in polynomial time.

Martin Grohe is an expert on graph minors, and had an important [paper](#) in the 2010 [LICS](#) conference—Logic in Computer Science. I thank Luca Aceto for pointing out the paper: it was discussed on his interesting [blog](#). Martin's paper is titled, "Fixed-Point Definability and Polynomial Time on Graphs with Excluded Minors."

This is what Aceto wrote about the paper:

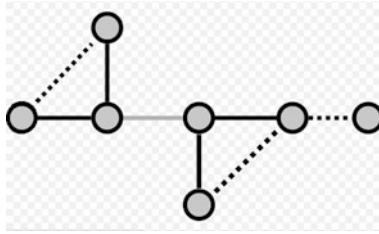
"For an interested, but not very knowledgeable, observer like me, one of the most interesting looking papers that have been selected for the conference seems to be yet another seminal contribution by Martin Grohe."

Actually, it seems like an amazing result to me. The detailed proof of Grohe's theorem will eventually be well over 200 pages—more on the length of the proof later.

One consequence of Grohe's powerful work is:

Theorem 58.2 *Consider the family of all graphs \mathcal{G}_H without H as a minor. Then there is a polynomial-time algorithm for graph isomorphism for this family of graphs.*

This is a vast generalization of the following classic result: graph isomorphism on planar graphs is in polynomial time. It further re-proves Eugene Luks' famous [theorem](#) on graph isomorphism for graphs with bounded degrees. Note that the family of all graphs with degree at most d is covered by Grohe's theorem: the graph H is just a star with $d + 2$ points. Another cool point about Grohe's theorem is it does not rely on the deep structure of finite groups as Luks' did. Here is an example of a graph with a minor of higher degree (four) than the graph itself (three), as can happen upon contracting an edge.



58.2 Fixed-Point Logic with Counting

Grohe actually proved that fixed-point logic with counting (FP+C) captures polynomial time over all classes of graphs with excluded minors. What does this mean? He shows for families of graphs without a given minor(s) there is a nice definition of polynomial algorithms: polynomial time equals what can be defined by a certain logic. As Ken Regan says the key is:

No one had provided a logic over *unordered* vertex labels which captures polynomial-time decidability on graphs.

This is what Grohe does for minor-excluded graphs.

One of the major quests, in an attempt to better understand polynomial time, is to discover more “natural” definitions of polynomial time. The usual definition of polynomial time is clear, but it is not easy to work with—especially for negative results. The hope is by replacing Turing machine definitions by logical ones, perhaps more progress can be made.

This program has been pushed by many top researchers over the years, and I will have to devote a whole discussion to it in the future. There are many beautiful results by Jin-Yi Cai, Martin Fürer, Neil Immerman, Moshe Vardi, and many others. Cai, Fürer, and Immerman showed FP+C does **not** capture all of polynomial time on any graph, for instance. Note, this beautiful theorem is the exact opposite of Grohe’s theorem.

58.3 Why Are Minor Results so Major?

An interesting question is why are proofs of the results on minor theorems so long? I do not claim to understand the reason, but I have a hypothesis. The obvious hypothesis is the proofs require many cases to be considered. This seems correct, but is not completely satisfying.

There should be a deeper reason why the proofs are so long. Perhaps there is some connection to the classification of simple groups? The reason I raise this is Grohe’s theorem, unlike Luks’, does not need the deep structure of finite groups. Instead, Martin relies on the Robertson–Seymour approach and shows there is a structure theorem for families of minor-excluded graphs. The problem, in my opinion, is that these structural theorems are not trivial even to state. One reason for this may be

that graphs are far less constrained than algebraic objects like groups. In any event the theorems are truly impressive.

58.4 Open Problems

One obvious open problem is to use Grohe's ideas to improve our understanding of graph isomorphism in general. Can we use his theorem to prove a better bound than is currently known? Can we get a polynomial-time graph isomorphism algorithm for graphs whose degree is $f(n)$ where $f(n)$ is a very slow growing function? I think this could be possible, but I am not an expert in this area.

58.5 Notes and Links

Original post:

<http://rjlipton.wordpress.com/2010/04/05/graph-isomorphism-and-graph-minors/>

Graph minors:

[http://en.wikipedia.org/wiki/Minor_\(graph_theory\)](http://en.wikipedia.org/wiki/Minor_(graph_theory))

Robertson–Seymour work on graph minors:

http://en.wikipedia.org/wiki/Robertson-Seymour_theorem

Grohe's paper:

<http://www-i7.informatik.rwth-aachen.de/~grohe/pub.de>

LICS conferences:

<http://www2.informatik.hu-berlin.de/lics/>

Blog post by Luca Aceto:

<http://processalgebra.blogspot.com/2010/03/lics-2010-accepted-paper-and-martin.html>

Eugene Luks on graph isomorphism:

http://en.wikipedia.org/wiki/Graph_isomorphism_problem#State_of_the_art

Alfred Tarski was one of the great logicians of the twentieth century. He is famous for the formal definition of truth, the decidability of many important theories, and many other results—one of the strangest is his famous joint [theorem](#) with Stefan Banach. This theorem proves that any solid ball of one pound of gold can be cut into a finite number of pieces and reassembled into two solid balls each of one pound of gold. This is a pretty neat trick.

We shall talk about the complexity of logic theories. There are some classic results that I think should be wider known, and their proofs use techniques that may have application elsewhere.

In the early days of computer science theory there was a great deal of interest in the complexity of decision problems of logical theories. I think these are natural problems, and often they could be answered with strong lower bounds. Further, some of them are close to practical questions. For example, special cases of the theory of the reals occur in many actual problems—notably in computational geometry. The theory restricted to existential sentences is quite natural and arises in many applications.

59.1 Upper Bounds on the Theory of the Reals

Tarski worked on various theories with the main goal of proving their decidability. Since Kurt Gödel's famous result on the incompleteness of Peano Arithmetic, a goal has been to find weaker theories that are complete. Such a theory has two neat properties:

- (1) the theory can either prove or disprove all statements in the theory;
- (2) the theory is usually decidable.

The latter only needs the theory to have a reasonable axiom system—then it will be decidable. Given a sentence ϕ in such a theory the decision procedure is simple:

Start enumerating proofs from the theory. If a proof of ϕ appears, then say the sentence is valid; if a proof of $\neg\phi$ appears, then say the sentence is invalid.

Note, one of ϕ or $\neg\phi$ must appear, since the theory is complete; and both cannot appear, since the theory is consistent. A complete theory is defined to always be consistent.

Tarski turned away from natural numbers to real numbers. The theory he studied is the theory of real numbers. It has a language that allows any first-order formula made up from $\{+, -, \times, =\}$, and its axioms are those for a field, with additional axioms for a real closed field. The last are axioms that state that every *odd* degree polynomial must have a root.

One of the main results of Tarski is:

Theorem 59.1 *The first-order theory of the real numbers is complete.*

He used a method that is called *quantifier elimination*. A theory has quantifier elimination if the theory has the property that every formula is equivalent to an open formula: a formula with no quantifiers. Tarski's theorem immediately proves that the theory is decidable, but left open the actual computational complexity of its decision problem. This called for computer science theorists to the rescue.

59.2 Lower Bounds on the Theory of the Reals

We give a table listing some of the basic results concerning decidability of various theories. The problem is: Given a sentence ϕ in the theory, is the sentence provable or not? Clearly for propositional logic the problem is NP-complete, and has unknown computational complexity.

Complexity of Logics		
Logic	Lower Bound	Who Proved
Propositional Logic	Open	Who?
Predicate Calculus	Undecidable	Church and Turing
Theory of Reals	2^{cn}	Fischer and Rabin

Predicate calculus concerns whether or not an arbitrary formula is logically valid; as long as there is at least one predicate letter $A(x, y, \dots)$ of arity at least two, the theory is undecidable. The trick is it is possible to encode the behavior of a Turing machine into a sentence using A , and show that the sentence is valid if and only if the Turing machine [halts](#).

Michael Fischer and Michael Rabin in 1974 proved several beautiful exponential lower bounds on the complexity of any decision procedure for various theories. I will discuss here their work on the theory of real numbers.

Of course no theory that includes propositional formulas can have an easy decision procedure, unless $P = NP$. My opinions aside, it is reasonable to expect the theory of the reals to be much more powerful than NP. This expectation turns out to be true: the interplay between the ability to add and multiply real numbers with predicate calculus's abilities allows powerful encoding tricks that cannot be done

in pure propositional logic. Together these two features, operations on reals and predicate calculus, allow strong lower bounds to be proved.

Fischer and Rabin proved:

Theorem 59.2 *In the theory of reals, there is a constant $c > 0$ so that there are true sentences of length $n > n_0$ whose shortest proof in the theory has length 2^{cn} .*

This is a very strong theorem. It is unconditional, and shows that the proofs themselves are huge. This is stronger than just proving they are hard to find, since if the proofs are large there can be no method that always proves valid theorems quickly—the proofs are just too big.

59.3 Lower Bounds on the Complexity of Theories

I thought that it might be useful to recall how such proofs are obtained. They are quite clever, in my opinion, and rely on several neat tricks. Partly I hope that these tricks could be used elsewhere in complexity theory.

How do you prove lower bounds on a theory? If the theory of the reals could define the set of integers, then the theory would be undecidable. Since the theory is decidable by the famous result of Tarski, it must be the case that the integers are undefinable. What Fischer and Rabin showed is how to define a very large set of integers, with a relatively small formula. This allowed them to use the same techniques that proved the predicate calculus was undecidable to prove their own result.

For example, imagine we could define $x \in \mathbb{N} \wedge x < 2^{2^n}$ by $B_n(x)$ where $B_n(x)$ is a “small” formula. Then, the plan is to use this to simulate the halting problem for Turing machines that do not use too much tape. An easy way to see that this will work is to look at a simpler application. Suppose that $P(x)$ stands for

$$\forall r \forall s \quad B_n(x) \wedge B_n(r) \wedge B_n(s) \wedge rs = x \rightarrow r = 1 \vee s = 1.$$

Then, clearly $P(x)$ defines the primes below 2^{2^n} . The sentence,

$$\forall x \exists y \quad B_n(x) \wedge B_{n+1}(y) \wedge y > x \wedge P(y) \wedge P(y + 2)$$

says that for any $x < 2^{2^n}$ there is a twin prime above it. This would be very exciting if it had a short proof—actually even weaker statements would be exciting. What Mike and Michael prove is in general such sentences are hard to prove—they do not, unfortunately, show this particular sentence is hard. Oh well.

59.4 A Recursion Trick

Fischer and Rabin do construct a formula $B_n(x)$ that defines

$$x \in \mathbb{N} \wedge x < 2^{2^n}.$$

Their construction uses recursion: B_n is defined in terms of B_m for some $m < n$. This should come as no shock, since recursion is so useful in all of computer science. However, in order to make their theorem work, they need to avoid making too many recursive calls. They use a “trick” that was discovered by Fischer with Albert Meyer and independently by Volker Strassen.

Let $F(x, y)$ be any formula and consider the conjunction:

$$G = F(x_1, y_1) \wedge F(x_2, y_2).$$

It follows that $G \leftrightarrow H$ where

$$H = \forall x \forall y [(x = x_1 \wedge y = y_1) \vee (x = x_2 \wedge y = y_2)] \rightarrow F(x, y).$$

The reason this construction is useful is simple: it is the size of the resulting formulas. The size of G is roughly twice that of F , but the size of H is only the size of F plus a constant. This is one of the key tricks used in their proof: it keeps the size of the formula B_n small.

59.5 A Name Trick

In their construction of B_n they also use many variable names. The naive method would use an exponential number of variables, and this would cause their construction to explode in size. They cleverly reuse names—in this way they avoid having to use very long names. It is not too far from a trick that was used, many years later, by Adi Shamir in his famous proof that $IP = PSPACE$.

Fischer and Rabin used this trick. Consider this formula,

$$\forall x \dots \forall x \quad A(x) \dots \tag{59.1}$$

There is a name collision between the first x and the second x : we would usually write this with different variables as

$$\forall x \dots \forall x' \quad A(x') \dots$$

to make it clearer. However, the name collision is allowed in first-order logic formulas, since it is easy to disambiguate the meaning of the formula (59.1). The importance of this trick is it allows the re-use of variable names; this re-use saves the number of variables needed and this reduces the size of the formulas B_n .

59.6 Lower Bounds on the Theory of the Complex Numbers?

I believe their results hold for this case too, but they do not seem to explicitly state this. Perhaps this is a well-known result, but I cannot track down the reference.

59.7 Open Problems

Can we use any of the tricks here in other parts of complexity theory? The recursion trick seems to me to be quite powerful. Perhaps we can use this trick in another part of theory.

59.8 Notes and Links

Original post:

<http://rjlipton.wordpress.com/2010/02/23/proving-that-proving-is-hard/>

Banach–Tarski theorem:

http://en.wikipedia.org/wiki/Banach-Tarski_paradox

Halting problem:

http://en.wikipedia.org/wiki/Halting_problem

Ken Thompson is one of the co-inventors of UNIX, perhaps one of the greatest programmers who ever lived, and won the 1983 Turing Award for his important work.

We—mainly Ken Regan as guest author—discuss Thompson’s work on chess playing programs. It is Ken on Ken, or rather Ken on *ken*: When Ken (Regan) played at the Westfield (NJ) chess club as a teen in matches against distant clubs, he knew Ken (Thompson) not by his famous programming handle but as the “club techie” who relayed his moves by telex. (Ken R. went on to become an International Master with a rating over 2400, before making academics his career.)

I once visited Bell Labs and spent some time talking to Ken T. on his chess program. I asked why he was doing so well—his program was winning a lot. He said his program probably had fewer bugs than the competition. In those early days chess programs were often buggy, and a program with a bug is likely to be a program with a chess weakness. The reason, Ken T. explained, the programs were so hard to get right was simple:

All chess programs looked at a huge set of positions and then generated **one** move. As long as the move was legal the programmer had no idea the program had made an error—unless the move led to an unexpected loss.

60.1 Chess Programs and the Big Match

Unlike us, Thompson has done pioneering research in computer chess programming. He and Joe Condon created the chess computer “Belle,” which won the 1980 World Computer Chess Championship and numerous other tournaments. In 1983 Belle became the first machine to achieve a recognized Master rating. It took only 14 more years for an IBM computer, “Deep Blue,” to topple the world champion, Garry Kasparov.

Today you can buy a \$50 program for your laptop computer that likely can beat Kasparov, at least according to published ratings. Weird to me, this meant that people following the 2010 World Championship Match online could often instantly

know more about the state of play than the human champions. Often but not always, as shown by the climactic game in which Viswanathan Anand of India won with Black to defeat challenger Veselin Topalov $6\frac{1}{2} - 5\frac{1}{2}$.

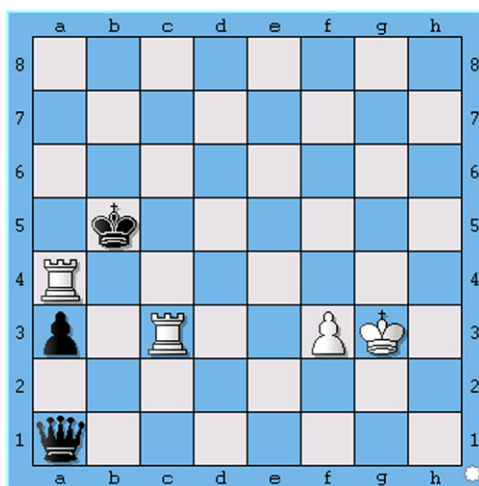
At Move 40, Anand calculated 11 moves ahead to realize that a position after Move 50 with only a King and three Pawns left for each side would be winning for him. Many computer programs seeing only yea-far for minutes thought Anand's move was a blunder allowing a draw, causing their owners to express consternation on numerous chat channels and blogs. Thus, sometimes the programs are wrong.

Yet Thompson's other great chess creation removes even that chance of error, and ties chess to several problems in computer science we've been thinking about.

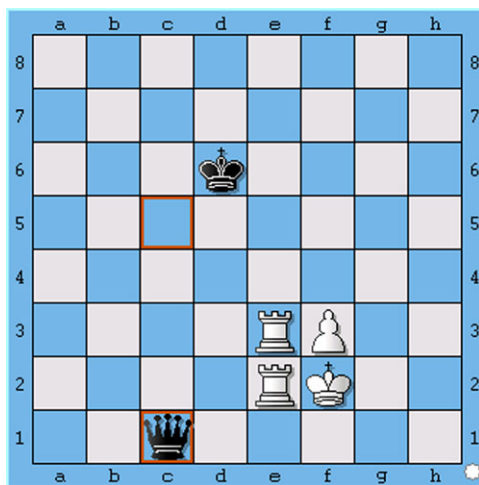
60.2 The Book of Chess

Thompson greatly extended algorithms and data structures for carrying out *perfect* analysis of chess endgame positions with at most five pieces on the board—three besides the kings. You could be better than him and better than Belle overall, but if you got down to 5 pieces, you might as well be “playing chess against God” as he called it. It was a chess version of Paul Erdős' proofs from the “Book,” in giving you not only the answer but the shortest route to the win or draw. His database overturned centuries of belief that certain endgames were drawn. You can win them—if you have the patience to find over 100 perfect moves . . .

In the past decade people have completed a perfect analysis of positions with *six* pieces, and are now working on *seven*. This discovered a position where it takes **517** moves to force a winning capture, and then a few more moves to checkmate. [That position](#) hasn't come up in any real game, but others do all the time. In fact, one could have arisen in the nerve-wracking 9th game where Anand had Topalov on the ropes, but missed knockouts that machines saw instantly. Consider this position:

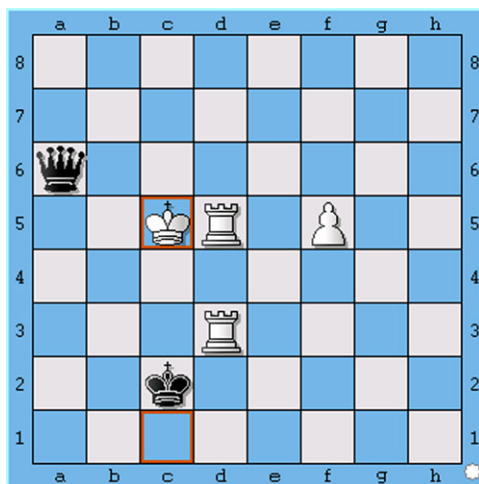


Anand, as White to move, lost his last chance by using his Rook on c3 to take Black's pawn. Most players would take with the other Rook to form a line along the 3rd rank. Leaving both Rooks on the rim gave Topalov an easy draw by perpetual check. Topalov could still have drawn, but it would take only a little inattention to allow White's Rooks to block checks and reach a position like this:



Here White can give checkmate—in **99** moves. I don't think a human would even find the first move, let alone all 99. It's not a Rook check; it's not pushing the pawn which White needs to promote to win; it's a move neither of us understands. You can see it by setting up this position at [Eiko Bleicher's online server](#). The only move to win is **Kf2-g2**, shunting aside the King.

If you keep clicking the best moves for both sides, you reach this position when it's mate-in-56:



Now “The Book of Chess” says to play **56.Rd3-d2+ Kc2-c1 55.Rd2-d1+ Kc1-c2 54.Rd1-d3!**, reaching the same position but with Black to move! This idea called *Zugzwang* is known to chess players in many immediate situations, but we have no idea why “The Book” uses it here.

White has 4 other winning moves, but the first two allow Black quickly to force White back to this position, so they are not *independent* wins. The other two *might* be independent, but they increase the “mating distance” from 56 to 72 and 74, respectively. Now we’re starting to use terms from graph theory, with positions as nodes and moves as edges, but these are not your familiar garden-variety graphs. These graphs typify *alternating computation*, which is a fundamental part of complexity theory. And these graphs are **big**.

60.3 Building Big Files

How big? Under Eugene Nalimov’s modification of Thompson’s original file format, the database for up to 5 pieces takes 7.05 gigabytes, compressed over 75 % from the originals. The complete compressed 6-piece set fills **1.2 terabytes**. That’s on the order of large data set examples given by Pankaj Agarwal in a University at Buffalo CSE Distinguished Speaker lecture in 2010, for NASA, Akamai, the LSS Telescope, and others. The complete 7-piece set has been estimated to be around 70 terabytes, and its computation may finish by 2015.

Nalimov’s distance-to-mate (“DTM”) files are built iteratively working backwards from checkmate positions. The newer ones (and Thompson’s originals) use distance-to-conversion (“DTC”), which means including all winning positions with fewer pieces in the ground set. Following chess endgame practice, let’s call the winning side White. The first iteration marks all positions where White can give checkmate—and/or make a winning capture—in one move. The next iteration finds all positions where Black can’t avoid the previously-marked positions (or in DTC, must make a losing capture). Note that this is only a subset of the usual graph neighborhood of the marked positions—it’s the subset of v for which *all* out-neighbors are marked. But the next iteration (with White again to move) marks all the positions that have edges to the previously-marked Black-to-move positions. The 517-move win was the only unmarked position left (up to symmetry) in the 1,034th iteration for those particular pieces, after months of runtime.

60.4 The Search Problem

You’d like to take a particular position p and decide whether White can win, which means analyzing *forward*. However, it is not known how to save appreciably over the brute-force backwards process, just to decide the search problem for one position. This is related to asking how “fast-mixing” the chess endgames are. In that single 99-move winning line, both Kings go all over the place, Black’s as well as White’s, so a pretty wide swath of the whole KRRP vs. KQ (with White’s pawn on the f-file) search space is involved.

The tablebases of course can help in analyzing forward from a given position, because when the usual alpha-beta search reaches a node with 6 or fewer pieces, it will return a perfect answer. But herein lies a paradox: the hard-disk fetches slow the search down so much that competitive chess programs often play *worse*. The world champion “Rybka 3” chess program works best with its tablebase-probe policy set to “Rarely” (the default) or “Never,” not “Normally” or “Frequently.”

The degree to which the chess search does not stay localized is reflected in poor cache performance by the tablebase files in practice. One remedy is putting selected endgames on a smaller but faster flash drive. Another is using so-called *bitbases*, which give only the win/draw information and use far less memory. Omitting the distance information, however, raises the problem of playing programs going around in cycles. Can we address the caching issue more directly, by changing the strategy of how they are encoded, without losing their full information?

Asymptotically, for chess on $n \times n$ boards, the search problem is complete for EXP or PSPACE according to whether one imposes a “generalized fifty-move rule” to limit the length of individual games. Whether a player has 2 or more independent winning strategies is equally hard. Of course, for chess on the 8×8 board we are talking about the concrete complexity of the problem.

60.5 Shorter Tablebase Descriptions?

Here’s one natural encoding shortcut that turns out to be *wrong*, exemplifying pitfalls of carrying intuition from garden-variety graphs to alternating graphs. Most chess programs today can solve mate-in-5 problems in microseconds, looking ahead 9–10 *ply* where 2 *ply* means a move for each player. Suppose we store only the checkmate positions, then those with a DTM of 5 moves = 10 *ply*, then those with DTM 20 *ply*, 30, 40, etc. The idea is that if a chess program probes a position with a DTM of 29, then the program can do 9 *ply* of ordinary search, enough to hit the DTM-20 stored positions. We might expect to save 90 % of the storage, on top of the 75 % compression already achieved.

This idea works for an ordinary leveled graph, but fails in an alternating graph. This is because Black can make inferior moves that hop over the 10-*ply*, 20-*ply*, 30-*ply* boundaries. This creates game paths in which the minimax search never sees a marked node, until its own depth limit aborts it without result.

What we would like to do is to find short descriptions for whole hosts of positions, say all-drawn or all-winning for White, that we can express in lookup rules. The idea is to *tile* the whole position space by these rules, in a way that needs only a lower order of exceptional positions to require individual table entries. This raises the **concept** of *succinctness*, namely descriptions of circuits that take an index i as (auxiliary) input, and need to output (only) the i -th bit of the function value. Is it feasible to produce that kind of circuit? The need for DTM or DTC info to find shortest paths and avoid cycling may make this idea difficult to realize, but it can certainly apply to bitbases.

60.6 The Tablebase Graphs

Can we organize the endgame files to make it feasible, given an initial position p in the database, to retrieve and cache only the parts needed in the search problem from p ? This relates to the expansion of the graph. Although the Anand–Topalov game example shows how progress from p can roam all over the board, perhaps the expansion is only through relatively few lines of play. This raises the idea of graphs that are expanders locally on small scales, but have irregular expansion on larger scales. If one can identify lines that transit between phases with play localized to part of the board, then the remaining positions would break into isolated components that could be read piecemeal into main memory.

This reminds me of papers on distance and clustering in large social-network graphs. The tablebase graph is huge and natural to study. What general properties does it have? Which classic graph properties and theorems can be carried over? Note the following trick: One can have position p with three winning moves a, b, c . There can be three later positions X, Y, Z such that the defender after move a can go to X or go to Y , after b can go to Y or Z , and after c can go to X or Z . In such cases position p remains winning even if any one of X, Y, Z were to change from “win” to “draw.” By analogy with *Menger’s Theorem* one might expect this to imply p has two independent winning strategies, but here that’s false. Notions of “flows” and “cuts” are hence also tricky to carry over.

60.7 The Book as “Deep” Oracle?

Ever since IBM’s “Deep Blue” beat Kasparov, multi-processing chess programs have prefixed “Deep” to their names. But here I mean a different technical use of the word: *computational depth*. A string x is computationally deep if it has short descriptions, but every short description requires a long time to produce x .

The Book of Chess has *almost no information* in classical terms—because it is described entirely by the rules of chess. These rules fit on half a page, and asking for the portion needed in the search from a given position p adds just one more line to describe p . If this portion really has no better computation than building basically the entire table, it is computationally deep.

Although the database entries for the positions you see when playing from the Anand–Topalov position p are classically the opposite of random, they sure *look* random, as far as which ones are wins or draws. Can they be used in applications where one desires pseudorandomness, but also wishes to leverage the digested information of an NP-complete or PSPACE-complete problem? There have been several papers in the past few years on the problem-solving power of deep strings, in general or specifically those that are witnesses for SAT. Certainly intense computational power has gone into building the files, so one feels they should provide power for doing something else.

60.8 Open Problems

The ultimate goal is to solve chess—does White win, does Black win, or is it a forced draw? Even as computers become vastly more powerful than humans, we still have no idea who wins this great game, even from some well-analyzed famous positions. Can we even prove that *Black* does *not* have a forced win? This may sound silly, but actually Bobby Fischer opined that in a particular position reached after five symmetrical opening moves, he would rather be Black!

A realistic goal is to try and extend the perfect tables of play to eight, nine, ten, and more pieces. The underlying algorithms are simple, but there seem to be challenging computer science questions. How to handle simple graph search on such huge graphs? How to apply methods that work for graph connectivity, but seem not to work on the game graphs? How to distribute large files, and optimize the caching tradeoffs?

60.9 Notes and Links

Original post:

<http://rjlipton.wordpress.com/2010/05/12/can-we-solve-chess-one-day/>

517-move win in chess:

http://en.wikipedia.org/wiki/Chess_endgame#Longest_forced_win

Chess endgame server:

<http://www.k4it.de/index.php?topic=egt&lang=en>

Reference on succinctness:

<http://rjlipton.wordpress.com/2010/01/05/an-approach-to-pnp-via-descriptive-complexity>

This post generated some useful comments from chess programmers. The thought of solving chess was pooh-poohed, but a notable hoax perpetrated by the news site ChessBase.com in April 2012 fooled a lot of people and revealed anxiety over this prospect. The fake [story](#) claimed that exhaustive computation with a cluster of over 100 CPUs maintained by Vasik Rajlich, the designer of the Rybka chess program, had analyzed the venerable King's Gambit opening to a draw. This had me (Ken) fooled for only a minute, mostly because the story bore an April 2 dateline, but ChessBase in their [followup](#) explained that the article had been posted when it was still April 1 somewhere on earth, a policy sometimes followed as well with conference deadlines.

ChessBase.com King's Gambit hoax:

<http://chessbase.com/newsdetail.asp?newsid=8047>

Fallout:

<http://www.chessbase.com/newsdetail.asp?newsid=8051>

Picture credits: Chess diagrams made by KWR with Arena GUI, then screen capture.

Virginia Vassilevska was a post-doc at Berkeley, before and after marrying Ryan Williams, and she works in several aspects of computational complexity. Her main area of expertise is in algorithms for various problems on graphs; for example, she has worked on new algorithms for finding cliques in graphs.

We will discuss some of Virginia's recent work on "cheating." I personally love the topic, and I think she has done some quite neat work on this problem.

Her work is on the complexity of "cheating" or "fixing" a single-elimination tournament. At the time of writing the original post this seemed especially relevant, since the Wimbledon tournament had just started—it is the oldest and perhaps most important of all the Grand-Slam tennis tournaments. More importantly it is a single-elimination tournament: lose one match and you go home.

Virginia's [paper](#) was presented at the [2010 AAI](#) conference. This is a conference of the Association for the Advancement of Artificial Intelligence (AAAI), and that year it was in Atlanta. Many of the papers are on similar topics to what we have at STOC/FOCS. There are papers on various algorithmic problems ranging from games, to economic problems, to routing problems, to approaches to solving SAT, and more.

I have never attended an AAAI conference, but I think I may start. It was easy for me in 2010, since I live in Atlanta. The technical program looked very interesting, and I found that many of the papers were potentially relevant to complexity theory. I am curious to see how the authors of papers at AAAI think about problems—perhaps they will have some new insights that will help us in our research. Perhaps.

AAAI's conference does some things differently from how we do them at our main conferences. For example, I noticed they have a *senior track*. At first I thought great, finally something special for some of us who have been around a while. But the truth is the track is for "survey" type talks. Oh well.

The Senior Member Presentation Track provides an opportunity for established researchers to give a broad talk on a well-developed body of research, an important new research area, or a thoughtful critique of trends in the field.

I wonder if we should have this at STOC/FOCS? Would you like to hear one of the “senior” members talk about new ideas or trends? I was interested to see how these talks go.

Let’s turn to Virginia’s results on cheating.

61.1 Ratings

The names of the key concepts in this area are a bit confusing—this has nothing to do with Virginia, since the names are well established. To make this discussion clearer, I feel I need to change the names a bit. I hope this is okay. I hope it does make it easier to follow.

As you would expect there are n players—what other variable could be used for the number of players? A *rating* R is a complete directed graph on the n players: for every two distinct players x and y either

$$x \rightarrow y \quad \text{or} \quad y \rightarrow x.$$

Think of $a \rightarrow b$ as meaning that player a always beats player b —always. Thus a rating R is the $\binom{n}{2}$ bits of information of who beats whom. Note, a rating is **not** the method used to decide anything. It is just the name for all the information about who beats whom.

A rating R is, of course, a “tournament” in the sense of graph theory. Seems confusing—no? That is why I decided to use slightly different names for this discussion.

Right away there is an idealization here. The assumption that a given player will always beat or lose to another player is clearly not true in real-life. If it were true there would be much less interest in watching Wimbledon. However, there are two reasons for making this assumption:

- For one, if probabilities are added, then the computational complexity of cheating becomes much harder. Even if the probabilities are restricted to be in the set

$$\{0, 1/2, 1\}$$

this and many problems become NP-hard. These results are [due](#) to Thuc Vu, Alon Altman, and Yoav Shoham.

- For another, even in the deterministic case the questions of how cheating works is already quite involved and interesting.

So we will continue to assume that the ratings R are perfect. If $a \rightarrow b$, then a always beats b .

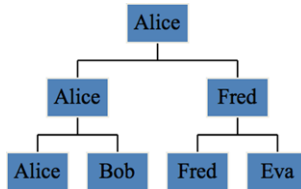
61.2 Tournaments

A rating R answers the question of whether or not a single player wins or loses to another player in head-to-head play. The central question is to use this information to select the “best” player. This is not easy, since as with voting problems there is

no simple method for defining the best player. Clearly, if one player can beat all the others, then there is no problem. But if the rating has no such super-winner, then we need a mechanism to decide who is the best.

One popular method is the single-elimination tournament, which is used at Wimbledon. Such a method is just a binary tree: at the leaves are the n players, and as they play the winner moves up to the ancestor node. This continues until one player is at the root of the tree, and they are declared the “best.” They get the first-place trophy, they get millions in prize money, and they are declared the best.

See Virginia’s [paper](#) for the formal definitions, but I hope it is clear what such a tree is. For example if there were just four players, Alice, Bob, Fred, and Eva, then a possible tournament would be:



Note that Alice is the winner, but it could be that Eva beats Alice in head-to-head play. Since they never play in this tree, Alice still wins. Let’s agree to call such trees *tournaments*.

61.3 Fixing Tournaments

Virginia proves a variety of interesting theorems—as usual see her pretty paper for the details. Here I will give a high-level view of her results.

Suppose that R is a rating of n players. The main question is when can we make a particular player p the winner? This means: when is there a tournament T so that when we play out the games the player p is the winner. Let’s call this fixing the tournament in p ’s favor.

There are two kinds of results that Virginia proves. First, there are necessary conditions on p and R for this to happen. Clearly, there cannot be any player q that beats everyone including p . If this were true, then there is no way to fix a tournament in p ’s favor. Moreover, if the tree is balanced, then it is also easy to prove that p must beat at least $\log_2 n$ other players.

Second, Virginia proves some nice results about very strong players. She shows that if p is a “super-king,” then there always is a way to fix a tournament so that p wins. A player p is a *king* provided for all other players q either

$$p \rightarrow q,$$

or for some other player r ,

$$p \rightarrow r \rightarrow q.$$

Then p is a *super-king* if p is a king, but when $p \not\rightarrow q$ there are at least $\log_2 n$ players r such that $p \rightarrow r \rightarrow q$.

Theorem 61.1 *Let R rate n players. If p is a king and more over the number of q so that*

$$p \rightarrow q$$

is at least $n/2$, then there is a tournament where p wins. Further this tournament can be constructed in polynomial time.

61.4 Open Problems

The main open question is: given a rating R , determine if there is a way to fix a tournament so player p wins. She leaves open the question of whether or not this is NP-hard, or has a polynomial-time algorithm. I have no good intuition on this problem—I do not have a good guess whether the problem is easy or hard.

In the early days right after the discovery of the concept of NP-completeness, there was a “theory” advanced by Ian Munro. He said that given a new problem, whether it was easy or hard was determined by the first person who thought about it. If you looked at it first and tried to prove it was NP-complete, then it was hard; if you first tried to find a polynomial algorithm, then it was easy. According to this theory, it is too bad the first attempts at SAT were to prove that it was NP-complete.

I know Ian was kidding—he has a great sense of humor—but one wonders if there is something to this theory.

Finally, Ken Regan has a question: if the probabilities $a \rightarrow b$ are “consistent” in the following technical sense, does this make deciding if a tournament can be fixed computationally easy? Assign to each player p a skill rating $S(p) \geq 0$, such as the [Elo](#) rating in Chess, which was created by Arpad Elo. Let the probability of $p \rightarrow q$ depend only on $S(p) - S(q)$. One can stipulate it to be 50–50 when $S(p) = S(q)$, and to be monotone-increasing in the difference.

61.5 Notes and Links

Original Post:

<http://rjlipton.wordpress.com/2010/06/23/cheating-tournaments/>

2010 AAI conference page:

<http://www.aaai.org/Conferences/AAAI/aaai10.php>

Paper on tournaments:

<http://www.cs.berkeley.edu/~virgi/tournament3.pdf>

Vu–Altman–Shoham paper:

<http://robotics.stanford.edu/~shoham/>

Elo chess rating system:

http://en.wikipedia.org/wiki/Elo_rating_system

Picture credit: TournamentTree.png is an original figure.

Arkadev Chattopadhyay is one of the experts on low-level complexity, but is especially an expert on computations modulo composite numbers. While he was a member of Princeton's Institute for Advanced Study, he worked with Avi Wigderson on various aspects of this important area.

We will explore representing Boolean functions as polynomials. The goal is to talk about representations modulo primes. The original post was cowritten with Ken Regan, who also knows Arkadev and his work well.

62.1 Representations of Boolean Functions

A Boolean function $f : \{0, 1\}^n \rightarrow \{0, 1\}$ can always be *represented* by a polynomial g . The rules of representation, modulo any prime p , are simple:

(1) If the Boolean function $f(x) = 0$, then $g(x) \equiv 0 \pmod{p}$.

(2) If the Boolean function $f(x) = 1$, then $g(x) \equiv 1 \pmod{p}$.

We may also require that the polynomial g be *multilinear*; recall this means that all the monomials of the polynomial have no x_i raised to a power higher than 1. Thus xyz is fine, but xy^2z is not. The rationale for this restriction is that the values of the variables will always be 0, 1 so $xyz = xy^2z$.

62.2 A Question

I, Dick, spoke to Arkadev before he went to Barcelona for part of summer 2010, and he explained a nice way to look at some basic questions about polynomial representations modulo primes.

The **degree modulo p** of a Boolean function $f(x)$ is the smallest degree of any $g(x)$ that represents $f(x)$ modulo p . We use

$$\deg_p(f)$$

to denote this. I asked him:

Suppose that $f(x)$ is a symmetric Boolean function with $\deg_p(f) = d$. Then must the polynomial that represents $f(x)$ of degree d be symmetric?

The reason I was interested in this question is I was thinking about weaker notions of representations, but for the kind we have defined he could immediately answer: “Yes.” Let’s see why he could answer so quickly.

62.3 Uniqueness

Let us define symmetric as used here. A Boolean function $f(x)$ is *symmetric*, if for all Boolean x and all permutations π ,

$$f(x_1, \dots, x_n) = f(x_{\pi(1)}, \dots, x_{\pi(n)}).$$

In the same way a polynomial g is *symmetric* if for all x and all permutations π ,

$$g(x_1, \dots, x_n) = g(x_{\pi(1)}, \dots, x_{\pi(n)}).$$

The reason he could answer so quickly, besides knowing this area so well, is that he relied on the following *Uniqueness Lemma*:

Lemma 62.1 *Suppose p is a prime. Then each Boolean function $f : \{0, 1\}^n \rightarrow \{0, 1\}$ has exactly one polynomial that represents f modulo p .*

This solves my question. Suppose that $f(x)$ is a symmetric Boolean function. Then it is easy to prove it has a symmetric polynomial representation. But by the Uniqueness Lemma, the polynomial of minimal degree must have the degree of this symmetric one. Very neat—both Ken and I agree.

The proof of this lemma is simple in the case of multilinear representations. Suppose that some Boolean function $f(x)$ had two such representations modulo p . Let $g(x)$ and $h(x)$ be the distinct polynomials. Then their difference

$$D(x) = g(x) - h(x)$$

must be a polynomial that is zero modulo p for all Boolean x , yet is not the zero polynomial. Let $x_{i_1}x_{i_2}\dots x_{i_d}$ be a monomial of $D(x)$ of minimum degree: if there are several choose one. Then set all the x_j ’s in this monomial to 1 and the rest to 0. At this point $D(x)$ has one non-zero term, and thus has the value 1 modulo p . This is a contradiction. Note that this argument relies on the polynomials being multilinear—else it would be possible for $D(x)$ to have a higher-degree term involving only variables in the chosen monomial, which would not be zeroed out by x .

62.4 Another Question

I, Dick, asked Arkadev another question: Suppose that $f(x, y)$ is a Boolean function of the form

$$a(x) \oplus b(y).$$

That is $f(x, y)$ is the exclusive-or sum of two Boolean functions on disjoint variables—such functions arise naturally in some learning problems. I asked what is

$$\deg_p(f),$$

where p is an odd prime? Again by a simple argument based on the Uniqueness Lemma, he could prove the degree is

$$\deg_p(a) + \deg_p(b).$$

Note that the restriction that p be odd is essential, since for $p = 2$ the answer is

$$\max(\deg_p(a), \deg_p(b)).$$

62.5 A Peek Ahead

The notion we used here for representation can be weakened, and often is when discussing computations modulo composites.

A Boolean function $f : \{0, 1\}^n \rightarrow \{0, 1\}$ can be *weakly represented* by a polynomial g . The rules of representation, modulo any number m , are now:

- (1) If the Boolean function $f(x) = 0$, then $g(x) \equiv 0 \pmod{m}$.
- (2) If the Boolean function $f(x) = 1$, then $g(x) \not\equiv 0 \pmod{m}$.

The key change is now we allow any non-zero value when the Boolean function is 1. Note that over a prime m this yields a representation as before if we replace $g(x)$ by

$$g(x)^{m-1}.$$

This is not possible to do if m is composite. We think of the mapping

$$x \rightarrow x^{p-1} \pmod{p}$$

as a “clean-up” function, since it sends 0 to 0 and all else to 1. The fact there is no such function modulo a composite is the main reason that composites start to become much more difficult to understand. Especially when coupled with the notion of weak representation.

62.6 Open Problems

What other properties of Boolean functions can be proved using the Uniqueness Lemma?

62.7 Notes and Links

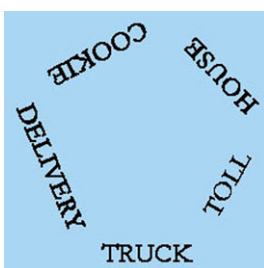
Original post:

<http://rjlipton.wordpress.com/2010/06/15/the-power-of-uniqueness/>

Referenced post on work with Wigderson:

<http://rjlipton.wordpress.com/2009/07/07/linear-equations-over-composite-moduli/>

Charles Bennett is famous for his seminal work on reversible computation, quantum computation and protocols, and many other wonderful results. He also enjoys wordplay, here is one example:



The point is you can read the cycle starting anywhere and still get a meaningful phrase. Check his [website](#) (referenced in the end notes) for an even more elaborate one.

I wish to finish by talking about impossibility proofs. Recently, quantum protocols have been attacked, after they had been proved secure. Are impossibility results really possible?

I have stated [before](#) that one of my all-time favorite “Far Side” cartoons is the one with a “burning arrow,” and the question:



Can they do that?

I actually tried, by the way, to see if I could legally post that cartoon on the blog—I was willing to pay a reasonable amount for the privilege—instead of the above. I went to the “Far Side” official site, filled out a request form, and got back an email. No way. Gary Larson will never legally approve anyone to post his material on the Web. Period. So no cartoon. Oh well.

It is possible to prove that something is impossible, but it sure is hard. My suggestion when anyone tries to tell you that X is impossible, is to translate it into the statement:

It is not possible to do X with the following allowed methods . . .

Those methods may include all possible ones or they may not. That is the key issue: do the methods cover all possible ones, including burning arrows?

A classic example of an impossibility result is based on a famous problem first raised by the Greeks: angle trisection with only a ruler and a compass. Pierre Wantzel proved in 1837:

Theorem 63.1 *There is no general method for angle trisection that uses only*
 (1) *an unmarked ruler;*
 (2) *a compass.*

Note, the proof of this famous theorem is perfect, but it can be “attacked.” With the allowed tools, certain angles can be trisected; any angle can be trisected up to an arbitrarily small error. Using slightly different tools, for example, a ruler with two marks, any angle **can** be trisected. So is angle trisection really impossible? Yes it is in a narrow sense, but beware of the burning arrows.

63.1 Quantum Protocols

Bennett and Gilles Brassard first suggested quantum protocols for secret key exchange that might be impossible to attack, in a beautiful paper. The title is “Quantum Cryptography: Public Key Distribution and Coin Tossing,” but the paper and its protocol are usually called just **BB84**.

Modern cryptography is really based on one of the most important notions of complexity theory: the notion of a **reduction**. A usual theorem in cryptography is of the form:

Theorem 63.2 *If there is a method to break the protocol X in polynomial time, then there is an algorithm that solves Y in polynomial time.*

The brilliant idea of BB84 was to change this to theorems of the form:

Theorem 63.3 *If there is a method to break the quantum protocol X , then quantum mechanics is wrong.*

Note, two differences. There is no restriction on the resources used by the adversary to break the quantum protocol, and second the consequence of an attack on such a protocol would be terrible: the consequence would be that *quantum mechanics is wrong*. Any theory, including quantum mechanics, may eventually need to be modified or extended. But for quantum theory to be wrong would be an immense

result. The theory is “tested” every day when we use modern devices, the theory continues to pass all the strange predictions that it makes, and it would be very surprising to find it wrong.

This is the beauty, the promise, and the excitement of quantum protocols. They have changed how protocols are proved secure. Classic security protocols are usually based on the hardness of a computational problem such as:

- the factorization of numbers that are the product of two large random primes;
- the solving of discrete logarithms over large finite fields;
- the solving of other number theoretic problems;
- and more recently, certain lattice-based problems.

What makes—made—quantum protocols so different was that the hardness was based on the correctness of quantum mechanics. The hope was that anyone who broke a quantum protocol would have to show that quantum mechanics is wrong. We believe in quantum mechanics, since its predictions have been spot on, so this seems like a pretty neat idea. This is why BB84 was such a major result, and why it generated so much interesting subsequent research.

The trouble is there are quantum attacks that use quantum burning arrows.

63.2 Attacks

There is now a long history of quantum protocols, with a corresponding large literature. The promise is still there, but quantum protocols have been just as fragile to the exact assumptions as classic protocols. Quantum proofs of impossibility of attacks have often failed—just as has happened over and over in classic hardness-based cryptography.

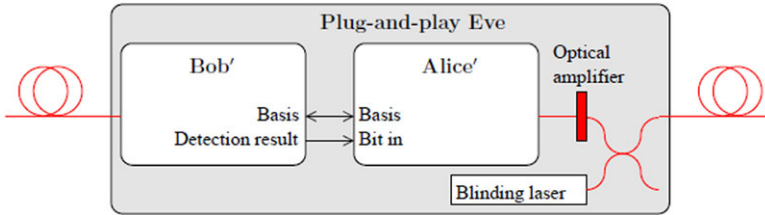
Here is a quote from a [paper](#) by Dominic Mayers in 1997:

In 1993, a protocol was proposed to realize bit commitment in the framework of quantum mechanics, and the unconditional security (see sections b and c) of this protocol has been generally accepted for quite some time. However, this result turned out to be wrong.

Here is another quote from a 2010 [paper](#) by Hoi-Kwan Lau and Hoi-Kwong Lo:

Recently, position-based quantum cryptography has been claimed to be unconditionally secure. In contrary, here we show that the existing proposals for position-based quantum cryptography are, in fact, insecure . . .

A recent paper got my attention, since it showed how to break quantum-based protocols by a general “man-in-the-middle” attack. The result is due to Lars Lydersen, Carlos Wiechers, Christoffer Wittmann, Dominique Elser, Johannes Skaar and Vadim Makarov. They point out in their paper that their attack is against implementations, not against the fundamental idea of quantum protocols. In the figure outlining the attack, the left end connects to Alice and the right end to Bob.



Roughly, Eve, the eavesdropper, does the following:

- (1) She receives photons from the real Alice, pretending to be Bob.
- (2) Eve processes them as Bob would, and then sends them off to the real Bob.

When the protocol is done, the secret is shared by all three—Alice, Bob, and Eve.

I will not try to give an even partial survey of all the results and counter-results in the last few decades. The key point to take away is simple:

Quantum protocols have many interesting advantages over classic ones, but they can be attacked.

One must be very careful to check what the attacks are and whether those attacks cover all feasible ones.

63.3 A Theorem

What is interesting about breaks in classic systems and breaks in quantum systems is that they usually, if not always, are fixable. The attack exploits some gap in the impossibility proof and the gap often can be fixed by modifying the protocol—sometimes very slightly. This is summarized in what I will call the **Crypto Designer's Theorem**:

Theorem 63.4 *For every cryptosystem S that is broken by some attack, there is another system S' with the following properties:*

- *The attack fails for S' .*
- *The system S' is a small modification of S .*

I leave the proof of this theorem to the reader.

63.4 Open Problems

Can we prove that quantum protocols are really secure? Or are we always going to have to worry—like classic systems—that someone will discover a burning arrow? What do you think?

63.5 Notes and Links

Original post:

<http://rjlipton.wordpress.com/2010/09/21/are-quantum-impossibility-proofs-possible/>

Bennett's website:

<http://researcher.watson.ibm.com/researcher/view.php?person=us-bennetc>

Referenced post with the "burning arrow":

<http://rjlipton.wordpress.com/2009/09/13/are-impossibility-proofs-possible/>

BB84 protocol:

<http://en.wikipedia.org/wiki/BB84>

Dominic Mayers' paper:

<http://arxiv.org/abs/quant-ph/9605044v2>

Lau–Lo paper:

<http://arxiv.org/abs/1009.2256>

Lydersen et al. paper:

<http://arxiv.org/pdf/1008.4593v1.pdf>

A popular exposition of the "man-in-the-middle" attack in the Lydersen et al. paper is the article by William Jackson, "Quantum crypto products cracked by researchers," in the Sept. 10, 2010 online issue of *GCN*,

<http://gcn.com/articles/2010/09/10/quantum-cryptography-security-risk.aspx>

This reiterated one of the iconic themes from 2009: the "burning arrow" representing attack methods or adverse possibilities that were thought to have been ruled out but weren't. In the next two years we took greater interest in quantum computing, including a long debate on whether quantum computing technology can possibly *scale* to produce realizable speedups over classical computers.

Picture credits:

1. Bennett pentagon is no longer on his website, as linked from <http://www.research.ibm.com/people/b/bennetc/>
2. Burning arrow: original source has vanished, Google Images now gives only our copy from the original post: <http://rjlipton.wordpress.com/2010/09/21/are-quantum-impossibility-proofs-possible/>
3. Bob–Eve attack cropped from screen capture of original Lydersen et al. paper <http://arxiv.org/pdf/1008.4593v1.pdf>