

MACHINE LEARNING AND THEOREM PROVING

Josef Urban

Czech Technical University in Prague

Summer School *Logic for the AI Spring*
September 13-14, 2022, Como



Outline

Motivation, Learning vs. Reasoning

Computer Understandable (Formal) Math

Bird's-Eye View of ATP and ML

Learning of Theorem Proving - Overview

Demos

High-level Reasoning Guidance: Premise Selection

Low Level Guidance of Theorem Provers

Mid-level Reasoning Guidance

Synthesis and Autoformalization

Motivation: Learning vs. Reasoning

“C’est par la logique qu’on démontre, c’est par l’intuition qu’on invente.”

(It is by logic that we prove, but by intuition that we discover.)

Henri Poincaré, *Mathematical Definitions and Education*.

“Hypothesen sind Netze; nur der fängt, wer auswirft.”

(Hypotheses are nets: only he who casts will catch.)

Novalis, quoted by Popper – *The Logic of Scientific Discovery*

Leibniz's/Hilbert's/Russell's Dream: Let Us Calculate!

Solve all (math, physics, law, economics, society, ...) problems by
reduction to logic/computation



[Adapted from: *Logicomix: An Epic Search for Truth* by A. Doxiadis]

How Do We Automate Math and Science?

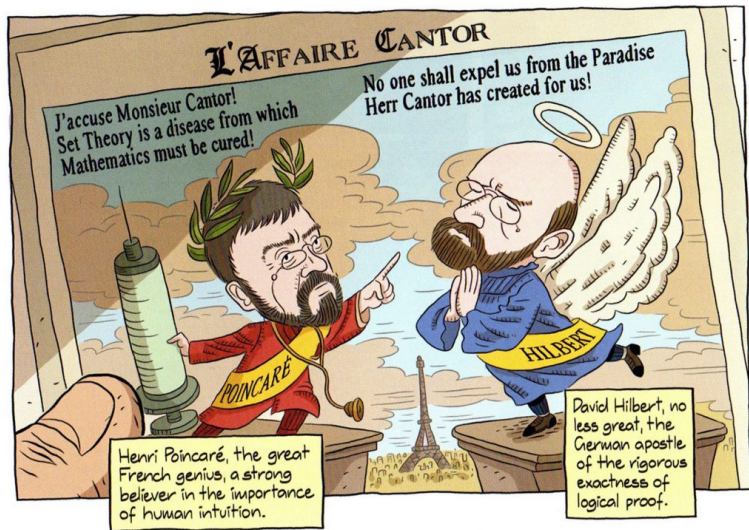
- What is mathematical and scientific thinking?
- Pattern-matching, analogy, induction from examples
- Deductive reasoning
- Complicated feedback loops between induction and deduction
- Using a lot of previous knowledge - both for induction and deduction

- We need to develop such methods on computers
- Are there any large corpora suitable for nontrivial deduction?
- Yes! Large libraries of formal proofs and theories
- So let's develop strong AI on them!

History, Motivation, AI/TP/ML

- Intuition vs Formal Reasoning – Poincaré vs Hilbert, Science & Method
- Turing's 1950 paper: **Learning Machines**, learn Chess?, undecidability??
- 50s-60s: Beginnings of ATP and ITP – Davis, Simon, Robinson, de Bruijn
- Lenat, Langley: **AM**, manually-written heuristics, learn Kepler laws,...
- Denzinger, Schulz, Goller, Fuchs – late 90's, ATP-focused:
Learning from Previous Proof Experience
- My MSc (1998): Try ILP to learn rules and heuristics from IMPS/Mizar
- Since: Use large formal math (Big Proof) corpora: Mizar, Isabelle, HOL ... to combine/develop symbolic/statistical deductive/inductive ML/TP/AI ... hammer-style methods, internal guidance, **feedback loops**, ...
- AI vs ML vs DL?: Ben Goertzel's 2018 Prague talk:
<https://youtu.be/Zt2HSTuGBn8>

Intuition vs Formal Reasoning – Poincaré vs Hilbert



[Adapted from: *Logicomix: An Epic Search for Truth* by A. Doxiadis]

Induction/Learning vs Reasoning – Henri Poincaré



- Science and Method: Ideas about the interplay between correct deduction and induction/intuition
- “*And in demonstration itself logic is not all. The true **mathematical reasoning is a real induction** [...]*”
- I believe he was right: strong general reasoning engines have to **combine deduction and induction** (learning patterns from data, making conjectures, etc.)

Learning vs Reasoning – Alan Turing 1950 – AI



- 1950: *Computing machinery and intelligence* – AI, Turing test
- “We may hope that machines will eventually compete with men in *all purely intellectual fields*.” (regardless of his 1936 undecidability result!)
- last section on **Learning Machines**:
- “But which are the best ones [fields] to start [learning on] with?”
- “... Even this is a difficult decision. Many people think that a very abstract activity, like the *playing of chess*, would be best.”
- Why not try with **math**? It is much more (universally?) expressive ...

Induction/Learning vs Reasoning – Turing 1950 – AI



- 1950: *Computing machinery and intelligence* – AI, Turing test
- On pure deduction: *“For at each stage when one is using a logical system, there is a very large number of alternative steps, any of which one is permitted to apply, so far as obedience to the rules of the logical system is concerned. These choices make the difference between a brilliant and a footling reasoner, not the difference between a sound and a fallacious one.”*

Why Combine Learning and Reasoning Today?

1 It practically helps!

- Automated theorem proving for large formal verification is **useful**:
 - Formal Proof of the Kepler Conjecture (2014 – Hales – 20k lemmas)
 - Formal Proof of the Feit-Thompson Theorem (2012 – Gonthier)
 - Verification of compilers (CompCert) and microkernels (seL4)
 - ...
- **But** good learning/AI methods needed to cope with large theories!

2 Blue Sky AI Visions:

- Get **strong AI** by learning/reasoning over large KBs of **human thought**?
- Big formal theories: good **semantic** approximation of such thinking KBs?
- Deep non-contradictory semantics – better than scanning books?
- Gradually try **learning math/science**:
 - What are the components (inductive/deductive thinking)?
 - How to combine them together?

The AITP Plan for World Domination

- 1 Make **large formal thought** (Mizar/MML, Isabelle/HOL/AFP, HOL/Flyspeck ...) accessible to strong reasoning and learning AI tools – **DONE** (or well under way)
- 2 Test/Use/Evolve existing AI and ATP tools on such large corpora
- 3 Build custom/combined inductive/deductive tools/metasystems
- 4 Continuously test performance, define harder AI tasks as the performance grows

Hilbert's update for 21st century (AGI'18):

NO ONE SHALL DRIVE US FROM THE SEMANTIC AI PARADISE
OF COMPUTER-UNDERSTANDABLE MATH AND SCIENCE!

aitp-conference.org - since 2016

Outline

Motivation, Learning vs. Reasoning

Computer Understandable (Formal) Math

Bird's-Eye View of ATP and ML

Learning of Theorem Proving - Overview

Demos

High-level Reasoning Guidance: Premise Selection

Low Level Guidance of Theorem Provers

Mid-level Reasoning Guidance

Synthesis and Autoformalization

What is Formal Mathematics?

- Developed thanks to the Leibniz/Russell/Frege/Hilbert/... program
- Mathematics put on **formal logic foundations** (*symbolic computation*)
- ... which btw. led also to the rise of computers (Turing/Church, 1930s)
- Formal math (1950/60s): **combine formal foundations and computers**
- **Proof assistants/Interactive theorem provers** and their large libraries:
- De Bruijn, Milner, Trybulec, Boyer and Moore, Gordon, Huet, Paulson, ...
- Automath (1967), LCF, Mizar, NQTHM, HOL, Coq, Isabelle, ACL2, Lean
- **Conceptually very simple:**
- Write all your axioms and theorems so that computer understands them
- Write all your inference rules so that computer understands them
- Use the computer to check that your proofs follow the rules
- **But in practice, it turns out not to be so simple**
- Many approaches, still not mainstream, but big breakthroughs recently

The QED Manifesto – 1994

- *QED is the very tentative title of a project to build a computer system that effectively represents all important mathematical knowledge and techniques.*
- *The QED system will conform to the highest standards of mathematical rigor, including the use of strict formality in the internal representation of knowledge and the use of mechanical methods to check proofs of the correctness of all entries in the system.*
- *The QED project will be a major scientific undertaking requiring the cooperation and effort of hundreds of deep mathematical minds, considerable ingenuity by many computer scientists, and broad support and leadership from research agencies.*
-
- *Never happened, but inspired a lot of development – “QED Singularity”*

Intros to ITP Systems and Formal Math

- Hales's talk at Bourbaki seminar:
<https://www.youtube.com/watch?v=wgfbt-X28XQ>
- Harrison's article on formalization:
<http://www.cl.cam.ac.uk/~jrh13/papers/cacm.pdf>
- Harrison, Urban, Wiedijk: History of Interactive Theorem Proving:
<http://www.cl.cam.ac.uk/~jrh13/papers/joerg.html>
- More recent: Just look up Kevin Buzzard on YouTube
- ITP, CPP, IJCAR, CADE conferences

Bird's Eye View of ITP Systems by T. Hales



HOL Light

HOL Light has an exquisite minimal design. It has the smallest kernel of any system. John Harrison is the sole



Mizar

Once the clear front-runner, it now shows signs of age. Do not expect to understand the inner workings of this system unless you have been



Coq

Coq is built of modular components on a foundation of dependent type theory. This system has grown one PhD thesis at a time.



Isabelle

Designed for use with multiple foundational architectures, Isabelle's early development featured classical constructions in set theory. However,



Metamath

Does this really work? Defying expectations, Metamath seems to function shockingly well for those who are happy to live without plumbing.



Lean

Lean is ambitious, and it will be massive. Do not be fooled by the name. "Construction area keep out" signs are prominently posted on the perimeter fencing.

F. Wiedijk: Irrationality of $\sqrt{2}$ (informal text)

tiny proof from Hardy & Wright, texts collected by F. Wiedijk:

Theorem 43 (Pythagoras' theorem). $\sqrt{2}$ is irrational.

The traditional proof ascribed to Pythagoras runs as follows. If $\sqrt{2}$ is rational, then the equation

$$a^2 = 2b^2 \tag{4.3.1}$$

is soluble in integers a, b with $(a, b) = 1$. Hence a^2 is even, and therefore a is even. If $a = 2c$, then $4c^2 = 2b^2$, $2c^2 = b^2$, and b is also even, contrary to the hypothesis that $(a, b) = 1$. \square

Irrationality of $\sqrt{2}$ (Formal Proof Sketch)

exactly the same text in Mizar syntax:

```
theorem Th43: :: Pythagoras' theorem
  sqrt 2 is irrational
proof
  assume sqrt 2 is rational;
  consider a,b such that
4_3_1: a^2 = 2*b^2 and
  a,b are relative prime;
  a^2 is even;
  a is even;
  consider c such that a = 2*c;
  4*c^2 = 2*b^2;
  2*c^2 = b^2;
  b is even;
  thus contradiction;
end;
```

Irrationality of $\sqrt{2}$ in HOL Light

```
let Sqrt_2_Irrational = prove
  (~rational(sqrt(&2)))`,
  SIMP_TAC[rational; real_abs; Sqrt_Pos_Le; REAL_POS] THEN
  REWRITE_TAC[NOT_EXISTS_THM] THEN REPEAT GEN_TAC THEN
  DISCH_THEN(CONJUNCTS_THEN2 ASSUME_TAC MP_TAC) THEN
  SUBGOAL_THEN (~((&p / &q) pow 2 = sqrt(&2) pow 2))`
    (fun th -> MESON_TAC[th]) THEN
  SIMP_TAC[Sqrt_Pow_2; REAL_POS; REAL_POW_DIV] THEN
  ASM_SIMP_TAC[REAL_EQ_LDIV_EQ; REAL_OF_NUM_LT; REAL_POW_LT;
    ARITH_RULE `0 < q <=> ~(q = 0)`] THEN
  ASM_MESON_TAC[NSqrt_2; REAL_OF_NUM_POW; REAL_OF_NUM_MUL; REAL_OF_NUM_EQ]];
```

Irrationality of $\sqrt{2}$ in Isabelle/HOL

```
theorem sqrt2_not_rational:
  "sqrt (real 2)  $\notin$   $\mathbb{Q}$ "
proof
  assume "sqrt (real 2)  $\in$   $\mathbb{Q}$ "
  then obtain m n :: nat where
    n_nonzero: "n  $\neq$  0" and sqrt_rat: "|sqrt (real 2)| = real m / real n"
    and lowest_terms: "gcd m n = 1" ..
  from n_nonzero and sqrt_rat have "real m = |sqrt (real 2)| * real n" by simp
  then have "real (m2) = (sqrt (real 2))2 * real (n2)"
    by (auto simp add: power2_eq_square)
  also have "(sqrt (real 2))2 = real 2" by simp
  also have "... * real (m2) = real (2 * n2)" by simp
  finally have eq: "m2 = 2 * n2" ..
  hence "2 dvd m2" ..
  with two_is_prime have dvd_m: "2 dvd m" by (rule prime_dvd_power_two)
  then obtain k where "m = 2 * k" ..
  with eq have "2 * n2 = 22 * k2" by (auto simp add: power2_eq_square mult_ac)
  hence "n2 = 2 * k2" by simp
  hence "2 dvd n2" ..
  with two_is_prime have "2 dvd n" by (rule prime_dvd_power_two)
  with dvd_m have "2 dvd gcd m n" by (rule gcd_greatest)
  with lowest_terms have "2 dvd 1" by simp
  thus False by arith
qed
```

Irrationality of $\sqrt{2}$ in Coq

```
Theorem irrational_sqrt_2: irrational (sqrt 2%nat).
intros p q H H0; case H.
apply (main_thm (Zabs_nat p)).
replace (Div2.double (q * q)) with (2 * (q * q));
  [idtac | unfold Div2.double; ring].
case (eq_nat_dec (Zabs_nat p * Zabs_nat p) (2 * (q * q))); auto; intros H1.
case (not_nm_INR _ _ H1); (repeat rewrite mult_INR).
rewrite <- (sqrt_def (INR 2)); auto with real.
rewrite H0; auto with real.
assert (q <> 0%R :=> R); auto with real.
field; auto with real; case p; simpl; intros; ring.
Qed.
```

Irrationality of $\sqrt{2}$ in Metamath

```
{
  $d x y $.
  $( The square root of 2 is irrational. $)
  sqr2irr $p |- ( sqr ` 2 ) e/ QQ $=
    ( vx vy c2 csqr cfv cq wnel wcel wn cv cddiv co wceq cn wrex cz cexp
    cmulc sqr2irrlem3 sqr2irrlem5 bi2rexa mtbir cc0 clt wbr wa wi wb nngt0t
    adantr cr ax0re ltmuldivt mp3an1 nnet zret syl2an mpd ancoms 2re 2pos
    sqrgt0i breq2 mpbii syl5bir cc ncnt mulzer2t syl breql d adant1 sylib d
    exp r19.23adv anc2li elnnc syl6ibr impac r19.22i2 mto elq df-nel mpbir )
  CDEZFGWDFHZIWEWDAJZBJZKLZMZBNOZAPQZWKWJANOZWLWFCQLCWGCQLRLMZBNOANOABSWIWM
  ABNNWFWGTUAUBWJWJAPNWFPHZWJWFNHZWNWJWNUCWFUDUEZUFWOWNWJWPWNWIWPBNWNWGNHZW
  IWPUGNWQUFZWIUCWGRLZWFUDUEZWPWRWTUCWHUDUEZWIWQWNWTXAUHZWQWNUFUCWGUDUEZXB
  WQXCWNWGUIUJWGUKHZWFUKHZXCXBUGZWFQWNUCCKHXDXEXFULUCWGWGFUMUNWGUOWFUPUQURUSW
  IUCWDUDUEXACUTVAVBWDWHUCUDVCVDVVEWQWTWPUHWNWQWSUCWFUDWQWGVFHWVSUCMVGWGVGVHV
  IVJVKVLVMNVVOWFVVPVQVRVSVTABWDWAUBWDFWBWC $.
  $( [8-Jan-02] $)
}
```

Irrationality of $\sqrt{2}$ in Metamath Proof Explorer

sqr2irr - Metamath Proof Explorer - Chromium

us.metamath.org/mpegif/sqr2irr.html

Proof of Theorem sqr2irr

Step	Hyp	Ref	Expression
1		sqr2irrlem3 10838	$\vdash \neg \exists x \in \mathbb{N} \exists y \in \mathbb{N} (x^2 = (2 \cdot (y^2)))$
2		sqr2irrlem5 10840	$\vdash ((x \in \mathbb{N} \wedge y \in \mathbb{N}) \rightarrow ((\sqrt{2} = (x/y) \leftrightarrow (x^2 = (2 \cdot (y^2)))))$
3	2	2rexbia 2329	$\vdash (\exists x \in \mathbb{N} \exists y \in \mathbb{N} (\sqrt{2} = (x/y) \leftrightarrow \exists x \in \mathbb{N} \exists y \in \mathbb{N} (x^2 = (2 \cdot (y^2))))$
4	1, 3	mtbir 288	$\vdash \neg \exists x \in \mathbb{N} \exists y \in \mathbb{N} (\sqrt{2} = (x/y))$
5		2re 8838	$\vdash 2 \in \mathbb{R}$
6		2pos 8849	$\vdash 0 < 2$
7	5, 6	sqrgt0i 10213	$\vdash 0 < (\sqrt{2})$
8		breq2 3595	$\vdash ((\sqrt{2} = (x/y) \rightarrow (0 < (\sqrt{2}) \leftrightarrow 0 < (x/y)))$
9	7, 8	mpbi 200	$\vdash ((\sqrt{2} = (x/y) \rightarrow 0 < (x/y))$
10		zre 9029	$\vdash (x \in \mathbb{Z} \rightarrow x \in \mathbb{R})$
11	10	adantr 444	$\vdash ((x \in \mathbb{Z} \wedge y \in \mathbb{N}) \rightarrow x \in \mathbb{R})$
12		nncr 8788	$\vdash (y \in \mathbb{N} \rightarrow y \in \mathbb{R})$
13	12	adantl 445	$\vdash ((x \in \mathbb{Z} \wedge y \in \mathbb{N}) \rightarrow y \in \mathbb{R})$
14		nngt0 8807	$\vdash (y \in \mathbb{N} \rightarrow 0 < y)$
15	14	adantl 445	$\vdash ((x \in \mathbb{Z} \wedge y \in \mathbb{N}) \rightarrow 0 < y)$
16		gt0div 8083	$\vdash ((x \in \mathbb{R} \wedge y \in \mathbb{R} \wedge 0 < y) \rightarrow (0 < x \leftrightarrow 0 < (x/y)))$
17	11, 13, 15, 16	sy3anc 1145	$\vdash ((x \in \mathbb{Z} \wedge y \in \mathbb{N}) \rightarrow (0 < x \leftrightarrow 0 < (x/y)))$
18	9, 17	sy5ibr 210	$\vdash ((x \in \mathbb{Z} \wedge y \in \mathbb{N}) \rightarrow ((\sqrt{2} = (x/y) \rightarrow 0 < x))$
19		simpl 436	$\vdash ((x \in \mathbb{Z} \wedge y \in \mathbb{N}) \rightarrow x \in \mathbb{Z})$
20	18, 19	ctild 522	$\vdash ((x \in \mathbb{Z} \wedge y \in \mathbb{N}) \rightarrow ((\sqrt{2} = (x/y) \rightarrow (x \in \mathbb{Z} \wedge 0 < x)))$
21		elnz 9035	$\vdash (x \in \mathbb{N} \leftrightarrow (x \in \mathbb{Z} \wedge 0 < x))$
22	20, 21	sy6ibr 210	$\vdash ((x \in \mathbb{Z} \wedge y \in \mathbb{N}) \rightarrow ((\sqrt{2} = (x/y) \rightarrow x \in \mathbb{N}))$
23	22	rexlimdva 2414	$\vdash (x \in \mathbb{Z} \rightarrow (\exists y \in \mathbb{N} (\sqrt{2} = (x/y) \rightarrow x \in \mathbb{N}))$
24	23	impac 598	$\vdash ((x \in \mathbb{Z} \wedge \exists y \in \mathbb{N} (\sqrt{2} = (x/y)) \rightarrow (x \in \mathbb{N} \wedge \exists y \in \mathbb{N} (\sqrt{2} = (x/y))))$
25	24	reximi2 2396	$\vdash (\exists x \in \mathbb{Z} \exists y \in \mathbb{N} (\sqrt{2} = (x/y) \rightarrow \exists x \in \mathbb{N} \exists y \in \mathbb{N} (\sqrt{2} = (x/y)))$
26	4, 25	mt0 165	$\vdash \neg \exists x \in \mathbb{Z} \exists y \in \mathbb{N} (\sqrt{2} = (x/y))$
27		elq 9308	$\vdash ((\sqrt{2}) \in \mathbb{Q} \leftrightarrow \exists x \in \mathbb{Z} \exists y \in \mathbb{N} (\sqrt{2} = (x/y)))$
28	26, 27	mtbir 288	$\vdash \neg (\sqrt{2}) \in \mathbb{Q}$
29		df-nel 3210	$\vdash ((\sqrt{2}) \notin \mathbb{Q} \leftrightarrow \neg (\sqrt{2}) \in \mathbb{Q})$
30	28, 29	mpbir 196	$\vdash (\sqrt{2}) \notin \mathbb{Q}$

Colors of variables: wff set class

Context menu: [new](#) [del](#) [copy](#) [paste](#) [undo](#) [redo](#) [find](#) [replace](#) [print](#) [help](#) [about](#) [quit](#)

Irrationality of $\sqrt{2}$ in Otter

Problem

```
set(auto).
set(ur_res).
assign(max_distinct_vars, 1).
list(usable).
x = x.
m(1,x) = x. %identity
m(x,1) = x.
m(x,m(y,z)) = m(m(x,y),z). %assoc
m(x,y) = m(y,x). %comm
m(x,y) != m(x,z) | y = z. %cancel
-d(x,y) | m(x,f(x,y)) = y. %divides
m(x,z) != y | d(x,y).
-d(2,m(x,y)) | d(2,x) | d(2,y). %2 prime
m(a,a) = m(2,m(b,b)). % a/b=sqrt(2)
-d(x,a) | -d(x,b) | x = 1. % a/b lowest
2 != 1.
end_of_list.
```

Proof

```
1 [] m(x,y)!=m(x,z)|y=z.
2 [] -d(x,y)|m(x,f(x,y))=y.
3 [] m(x,y)!=z|d(x,z).
4 [] -d(2,m(x,y))|d(2,x)|d(2,y).
5 [] -d(x,a)|-d(x,b)|x=1.
6 [] 2!=1.
7 [factor,4.2.3] -d(2,m(x,x))|d(2,x).
13 [] m(x,m(y,z))=m(m(x,y),z).
14 [copy,13,flip.1] m(m(x,y),z)=m(x,m(y,
16 [] m(x,y)=m(y,x).
17 [] m(a,a)=m(2,m(b,b)).
18 [copy,17,flip.1] m(2,m(b,b))=m(a,a).
30 [hyper,18,3] d(2,m(a,a)).
39 [para_from,18.1.1,1.1.1] m(a,a)!=m(2,
42 [hyper,30,7] d(2,a).
46 [hyper,42,2] m(2,f(2,a))=a.
48 [ur,42,5,6] -d(2,b).
50 [ur,48,7] -d(2,m(b,b)).
59 [ur,50,3] m(2,x)!=m(b,b).
60 [copy,59,flip.1] m(b,b)!=m(2,x).
145 [para_from,46.1.1,14.1.1.1,flip.1] m
189 [ur,60,39] m(a,a)!=m(2,m(2,x)).
190 [copy,189,flip.1] m(2,m(2,x))!=m(a,a
1261 [para_into,145.1.1.2,16.1.1] m(2,m(
1272 [para_from,145.1.1,190.1.1.2] m(2,m
1273 [binary,1272.1,1261.1] $F.
```

Today: Computers Checking Large Math Proofs



Scientists Deliver Formal Proof of Famous Kepler Conjecture

Jun 16, 2017 by News Staff / Source

◀ Previous | Next ▶

Published in
Mathematics

Tagged as
Johannes Kepler
Kepler conjecture

**Follow
You Might Like**



Researchers Develop First-Ever 3D Numerical Model of Melting Snowflake



Researchers Develop Mathematical Model for How Innovations

An international team of mathematicians led by University of Pittsburgh Professor **Thomas Hales** has delivered a formal proof of the **Kepler conjecture**, a famous problem in discrete geometry. The team's **paper** is published in the journal *Forum of Mathematics, Pi*.



LATEST NEWS



SPHERE Captures Young Exoplanet Beta Pictoris b Orbiting around Its Star

Nov 13, 2018 | Astronomy



Mirace eatoni: Newly-Discovered Cretaceous Bird Lived Among Dinosaurs, Was Strong Flier

Nov 13, 2018 | Paleontology



Juno Takes Closer Look at Jupiter's Magnificent, Swirling Clouds

Nov 13, 2018 | Space Exploration



Physicists Solve Structure of Unusually Complex Form of Nitrogen

Nov 13, 2018 | Physical Chemistry



Natural Compound Protects Hypertensive Rats against Heart Disease

Nov 13, 2018 | Medicine



Inventive Orangutans Make Hook Tools to Retrieve Food

Nov 12, 2018 | Biology



Researchers Find 40,000-Year-Old Figurative Paintings in Bornean Cave

Nov 12, 2018 | Archaeology

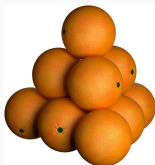


Hubble Sees Lensing Galaxy Cluster,

cdn.sci-news.com/images/enlarge3/image_4960e-Kepler-Conjecture.jpg

Big Example: The Flyspeck project

- Kepler conjecture (1611): The most compact way of stacking balls of the same size in space is a pyramid.



$$V = \frac{\pi}{\sqrt{18}} \approx 74\%$$

- Formal proof finished in 2014
- 20000 lemmas in geometry, analysis, graph theory
- All of it at <https://code.google.com/p/flyspeck/>
- All of it **computer-understandable and verified** in HOL Light:
- `polyhedron s /\ c face_of s ==> polyhedron c`
- However, this took **20 – 30 person-years!**

Kepler conjecture formally in HOL Light

```
|- packing V <=>
  (!u v. u IN V /\ v IN V /\ dist(u,v) < &2 ==> u = v)

|- the_kepler_conjecture <=>
  (!V. packing V
    ==> (?c. !r. &1 <= r
      ==> &(CARD(V INTER ball(vec 0,r))) <=
        pi * r pow 3 / sqrt(&18) + c * r pow 2))
```

Kepler conjecture informally

In words, we define the Kepler conjecture to be the following claim: for every packing V , there exists a real number c such that for every real number $r \geq 1$, the number of elements of V contained in an open spherical container of radius r centered at the origin is at most

$$\frac{\pi r^3}{\sqrt{18}} + c r^2.$$

An analysis of the proof shows that there exists a small computable constant c that works uniformly for all packings V , but we only formalize the weaker statement that allows c to depend on V . The restriction $r \geq 1$, which bounds r away from 0, is needed because there can be arbitrarily small containers whose intersection with V is nonempty.

Parts of Flyspeck

- combination of traditional mathematical argument and three separate bodies of computer calculations.
- nearly a thousand nonlinear inequalities.
- The combinatorial structure of each possible counterexample to the Kepler conjecture is encoded as a plane graph satisfying a number of restrictive conditions. Any graph satisfying these conditions is said to be *tame*.
- A list of all tame plane graphs up to isomorphism has been generated by an exhaustive computer search. The formal statement that every tame plane graph is isomorphic to one of these cases. This was part was done in Isabelle and imported into HOL Light.
- a large collection of linear programs.

Independent verification of Flyspeck

- Mark Adams: HOL Zero system
- more secure than HOL Light, independently implemented
- an fast exporter of the HOL Light verifications based on kernel modifications
- verification of every HOL Light kernel step inside HOL Zero
- so far only for the text part (the other parts are much slower)

What Has Been Formalized? (2014)

top 100 of interesting theorems/proofs (Paul & Jack Abad, 1999),
tracked by Freek Wiedijk - <https://www.cs.ru.nl/~freek/100/>

1. $\sqrt{2} \notin \mathbb{Q}$	<i>all together</i> 88%
2. fundamental theorem of algebra	HOL Light 86%
3. $ \mathbb{Q} = \aleph_0$	Mizar 57%
4. $a^2 + b^2 = c^2 \Rightarrow \triangle$	Isabelle 52%
5. $\pi(x) \sim \frac{x}{\ln x}$	Coq 49%
6. Gödel's incompleteness theorem	ProofPower 42%
7. $\left(\frac{p}{q}\right)\left(\frac{q}{p}\right) = (-1)^{\frac{p-1}{2}\frac{q-1}{2}}$	Metamath 24%
8. impossibility of trisecting the angle and doubling the cube	ACL2 18%
⋮	PVS 16%
32. four color theorem	
33. Fermat's last theorem	
⋮	
99. Buffon needle problem	
100. Descartes rule of signs	

What Has Been Formalized? (2022)

top 100 of interesting theorems/proofs (Paul & Jack Abad, 1999),
tracked by Freek Wiedijk - <https://www.cs.ru.nl/~freek/100/>


1. $\sqrt{2} \notin \mathbb{Q}$	<i>all together</i>	98%
2. fundamental theorem of algebra	HOL Light	86%
3. $ \mathbb{Q} = \aleph_0$	Mizar	69%
4. $a^2 + b^2 = c^2 \Rightarrow \triangle$	Isabelle	86%
5. $\pi(x) \sim \frac{x}{\ln x}$	Coq	78%
6. Gödel's incompleteness theorem	ProofPower	43%
7. $\left(\frac{p}{q}\right)\left(\frac{q}{p}\right) = (-1)^{\frac{p-1}{2}\frac{q-1}{2}}$	Metamath	74%
8. impossibility of trisecting the angle and doubling the cube	Lean	70%
⋮	ACL2	18%
	PVS	22%
32. four color theorem		
33. Fermat's last theorem		
⋮		
99. Buffon needle problem		
100. Descartes rule of signs		

Named Theorems in the Mizar Library

FM - Chromium

fm.uwb.edu.pl/mmiquery/fillin.php?filedfilename=mml-facts.mqt&argument=number+102

Mizar home, download files: [abstr.](#), [articles](#), [bin.](#), [doc.](#), [emacs](#) [gabs](#), [fmbibs](#), [gabs](#) (more) [semantic MML](#)



MML Query (beta)

Template maker
Environment explanation

Mizar TWiki
MML Query server
Megrez services
Journals:
[FM: MetaPRESS](#), [server](#), [proof-read](#), [regeneration](#)
[MMA](#)
(preparation)

Syntax: [xml](#), [html](#)
[Downloads](#)

[Mizar syntax](#), [xml](#), [txt](#)

[MML 5.25.1220](#)
- [most important facts](#)
(other collection)

- Birkhoff

The most important facts in MML ([decode](#))

[add description](#)

See also [Name carrying facts/theorems/definitions in MML](#)

1	"Alexander's Lemma"	=> WAYBEL_7:31	VOTE
2	"All Primes (1 mod 4) Equal the Sum of Two Squares"	=> NAT_5:23	VOTE
3	"Axiom of Choice"	=> WELLORD2:18	VOTE
4	"Baire Category Theorem (Banach spaces)"	=> LOPBAN_5:3	VOTE
5	"Baire Category Theorem (Hausdorff spaces)"	=> NORMSP_2:10	VOTE
6	"Baire Category Theorem for Continuous Lattices"	=> WAYBEL12:39	VOTE
7	"Banach Fix Point Theorem for Compact Spaces"	=> AL12:1	VOTE
8	"Banach-Steinhaus theorem (uniform boundedness)"	=> LOPBAN_5:7	VOTE
9	"Bertrand's Ballot Theorem"	=> BALLOT_1:28	VOTE
10	"Bertrand's postulate"	=> NAT_4:56	VOTE
11	"Bezout's Theorem"	=> NEWTON:67	VOTE
12	"Bing Theorem"	=> NAGATA_2:22	VOTE
13	"Binomial Theorem"	=> BINOM:25	VOTE
14	"Birkhoff Variety Theorem"	=> BIRKHOFF:sch 12	VOTE
15	"Bolzano theorem (intermediate value)"	=> TOPREAL5:8	VOTE
16	"Bolzano-Weierstrass Theorem (1 dimension)"	=> SEO_4:40	VOTE
17	"Borsuk Theorem on Decomposition of Strong Deformation Retracts"	=> BORSUK_1:42	VOTE
18	"Borsuk-Ulam Theorem"	=> BORSUK_7:condreg 3	VOTE
19	"Boundary Points of Locally Euclidean Spaces"	=> MFOLD_0:2	VOTE
20	"Brouwer Fixed Point Theorem"	=> BROUWER:14	VOTE
21	"Brouwer Fixed Point Theorem for Disks on the Plane"	=> BROUWER:15	VOTE
22	"Brouwer Fixed Point Theorem for Intervals"	=> TREAL_1:24	VOTE
23	"Brown Theorem"	=> GCD_1:40	VOTE
24	"Cantor Theorem"	=> CARD_1:14	VOTE
25	"Cantor-Bernstein Theorem"	=> CARD_1:10	VOTE

Big Formalizations

- Kepler Conjecture (Hales et al, 2014, HOL Light, Isabelle)
- Feit-Thompson (odd-order) theorem
 - Two graduate books
 - Gonthier et al, 2012, Coq
- Compendium of Continuous Lattices (CCL)
 - 60% of the book formalized in Mizar
 - Bancerek, Trybulec et al, 2003
- The Four Color Theorem (Gonthier and Werner, 2005, Coq)

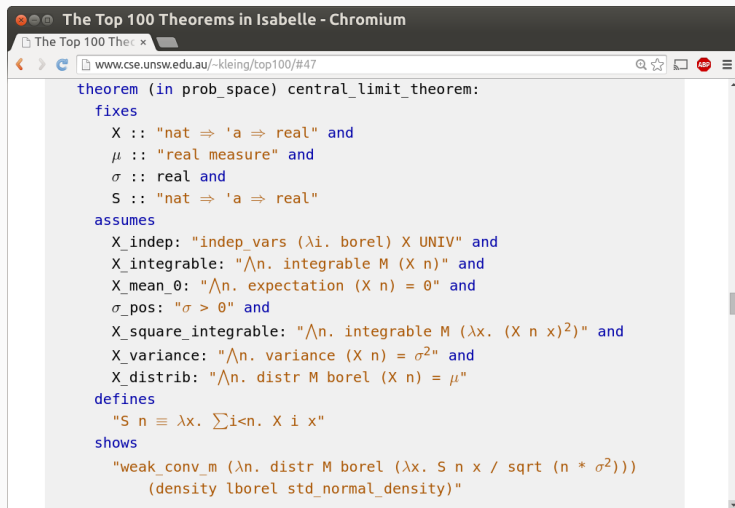
Mid-size Formalizations

- Gödel's First Incompleteness – N. Shankar (NQTHM), R. O'Connor (Coq)
- Brouwer Fixed Point Theorem – K. Pak (Mizar), J. Harrison (HOL Light)
- Jordan Curve Th. – T. Hales (HOL Light), A. Kornilowicz et al. (Mizar)
- Prime Number Th. – J. Avigad et al (Isab/HOL), J. Harrison (HOL Light)
- Gödel's Second Incompleteness Theorem – L. Paulson (Isabelle/HOL)
- Central Limit Theorem – J. Avigad (Isabelle/HOL)
- Consistency of the Negation of CH – J. Han and F. van Doorn (Lean)
- ... and many more

Large Software Verifications

- seL4 – operating system microkernel
 - Gerwin Klein and his group at NICTA, Isabelle/HOL
- CompCert – a formally verified C compiler
 - Xavier Leroy and his group at INRIA, Coq
- EURO-MILS – verified virtualization platform
 - ongoing 6M EUR FP7 project, Isabelle
- CakeML – verified implementation of ML
 - Magnus Myreen, HOL4

Central Limit Theorem in Isabelle/HOL



The screenshot shows a Chromium browser window titled "The Top 100 Theorems in Isabelle - Chromium". The address bar displays the URL "www.cse.unsw.edu.au/~kleing/top100/#47". The main content area shows the Isabelle/HOL code for the Central Limit Theorem. The code is as follows:

```
theorem (in prob_space) central_limit_theorem:
  fixes
    X :: "nat  $\Rightarrow$  'a  $\Rightarrow$  real" and
     $\mu$  :: "real measure" and
     $\sigma$  :: real and
    S :: "nat  $\Rightarrow$  'a  $\Rightarrow$  real"
  assumes
    X_indep: "indep_vars ( $\lambda$ i. borel) X UNIV" and
    X_integrable: " $\bigwedge$ n. integrable M (X n)" and
    X_mean_0: " $\bigwedge$ n. expectation (X n) = 0" and
     $\sigma$ _pos: " $\sigma > 0$ " and
    X_square_integrable: " $\bigwedge$ n. integrable M ( $\lambda$ x. (X n x)2)" and
    X_variance: " $\bigwedge$ n. variance (X n) =  $\sigma^2$ " and
    X_distrib: " $\bigwedge$ n. distr M borel (X n) =  $\mu$ "
  defines
    "S n  $\equiv$   $\lambda$ x.  $\sum$ i<n. X i x"
  shows
    "weak_conv_m ( $\lambda$ n. distr M borel ( $\lambda$ x. S n x / sqrt (n *  $\sigma^2$ )))
      (density lborel std_normal_density)"
```

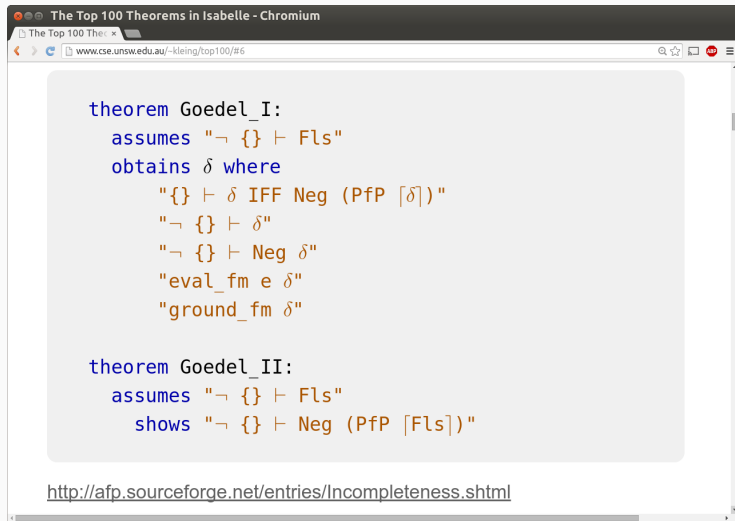
Sylow's Theorems in Mizar

```
theorem :: GROUP_10:12
  for G being finite Group, p being prime (natural number)
  holds ex P being Subgroup of G st P is_Sylow_p-subgroup_of_prime p;
```

```
theorem :: GROUP_10:14
  for G being finite Group, p being prime (natural number) holds
  (for H being Subgroup of G st H is_p-group_of_prime p holds
    ex P being Subgroup of G st
      P is_Sylow_p-subgroup_of_prime p & H is Subgroup of P) &
  (for P1,P2 being Subgroup of G
    st P1 is_Sylow_p-subgroup_of_prime p & P2 is_Sylow_p-subgroup_of_prime p
    holds P1,P2 are_conjugated);
```

```
theorem :: GROUP_10:15
  for G being finite Group, p being prime (natural number) holds
  card the_sylow_p-subgroups_of_prime(p,G) mod p = 1 &
  card the_sylow_p-subgroups_of_prime(p,G) divides ord G;
```

Gödel Theorems in Isabelle

A screenshot of a web browser window titled "The Top 100 Theorems in Isabelle - Chromium". The address bar shows the URL "www.cse.unsw.edu.au/~kleing/top100/#6". The main content area displays Isabelle/HOL code for two theorems, Gödel_I and Gödel_II. The code is color-coded: keywords like "theorem", "assumes", "obtains", and "shows" are in blue, while logical expressions and variables are in orange. The code defines Gödel's first incompleteness theorem (Gödel_I) and its proof (Gödel_II).

```
theorem Goedel_I:  
  assumes "¬ {} ⊢ Fls"  
  obtains δ where  
    "{} ⊢ δ IFF Neg (PfP [δ])"  
    "¬ {} ⊢ δ"  
    "¬ {} ⊢ Neg δ"  
    "eval_fm e δ"  
    "ground_fm δ"  
  
theorem Goedel_II:  
  assumes "¬ {} ⊢ Fls"  
  shows "¬ {} ⊢ Neg (PfP [Fls])"
```

<http://afp.sourceforge.net/entries/Incompleteness.shtml>

Prime Number Theorem in HOL Light

```
|- ((\n. &(CARD {p | prime p /\ p <= n}) / (&n / log(&n)))  
    ---> &1) sequentially
```

Foundational Wars - Set Theory

- Mizar, MetaMath, Isabelle/ZF, Naproche/SAD
- ZFC
- Tarski-Grothendieck (added inaccessible cardinals)
- strong choice
- issues:
 - how to add a type system (**soft types** - predicates with automation)
 - how to handle higher-order reasoning
 - how to compute

Foundational Wars - Higher-order logic (HOL)

- HOL4, HOL Light, Isabelle/HOL, ProofPower, HOL Zero
- based on polymorphic simply-typed lambda calculus
- but quickly added extensionality and choice (classical)
- weaker than set theory - canonical model is $V_{\omega+\omega} \setminus \{0\}$
- *HOL universe*: U is a set of non-empty sets, such that
 - U is closed under non-empty subsets, finite products and powersets
 - an infinite set $I \in U$ exists
 - a choice function ch over U exists (i.e., $\forall X \in U : ch(X) \in X$)
 - guarantees also function spaces ($I \rightarrow I$)
- Isabelle adds typeclasses, ad-hoc overloading
- issues:
 - can be too weak
 - not so well known foundations as ZFC
 - the type system does not have dependent types (e.g. matrix over a ring)
 - how to compute

Foundational Wars - Type theory

- Coq, Lean, Agda, NuPrl, HoTT
- constructive type theory (classical in Lean)
- Curry-Howard isomorphism:
 - formulas as types
 - proofs as terms
- proofs are in your universe of discourse!
- two proofs of the same formula might not be equal!
- what does it mean?
- excluded middle avoided, classical math not supported so much
- computation is a big topic
- very rich type system
- lots of research issues for constructivists
- non-experts typically don't have a good idea about the semantics of it all
- *'they have been calling it baroque, but it's almost rococo'* (A. Trybulec)

Foundational Wars - Logical Frameworks

- LF, Twelf, MMT, Isabelle?, Metamath?
- Try to cater for everybody
- Let users encode their logic and inference rules (deep embedding)
- issues:
 - None of them really used
 - maintenance – the embedded systems evolve fast
 - efficiency: Isabelle/Pure ended up enriching its kernel to fit HOL
 - efficiency: things like computation
 - probably needs a lot of investment to benefit multiple foundations
 - more ad-hoc translations between systems are often cheaper to develop

Implementation

- Most systems written in ML (OCAML or SML)
- Sometimes Lisp, Pascal, C++
- LCF approach (Milner): small inference kernel
- isolated by an abstract ML datatype "theorem"
- this means that only a small number of allowed inferences can result in a "theorem"
- Every more complicated procedure has to produce the kernel inferences, to get a "theorem"
- HOL Light - about 400 lines for the whole kernel
- Coq - about 20000 lines

HOL Light kernel - terms and types

```
module Hol : Hol_kernel = struct

  type hol_type = Tyvar of string
                | Tyapp of string * hol_type list

  type term = Var of string * hol_type
            | Const of string * hol_type
            | Comb of term * term
            | Abs of term * term

  type thm = Sequent of (term list * term)
```

THE DAILY NEWSLETTER

Sign up to our daily email newsletter

NewScientist

SUBSCRIBE AND SAVE 64%

News Technology Space Physics Health Environment Mind Video | Travel Live Jobs

Sign In Search

Home | News | Technology

TECHNOLOGY NEWS 16 September 2015

Unhackable kernel could keep all computers safe from cyberattack

From helicopters to medical devices and power stations, [mathematical proof](#) that software at the heart of an operating system is secure could keep hackers out



POPULAR

We thought the Incas couldn't write. These knots change everything

End of days: Is Western civilisation on the brink of collapse?

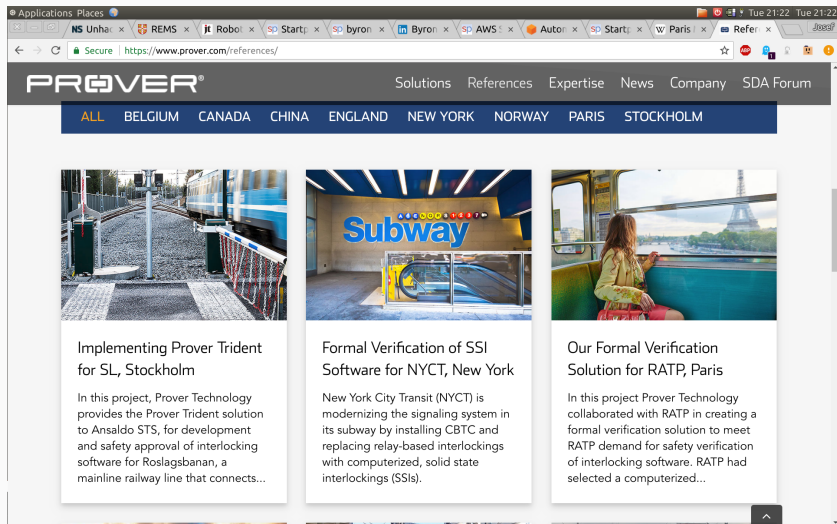
The origins of sexism: How men came to rule 12,000 years ago

The brain's 7D sandcastles could be

Unhackable kernel could keep all computers safe from cyberattack

Is quantum physics behind your brain's ability to think?

Today's Applications



The screenshot shows a web browser displaying the Prover website. The browser's address bar shows the URL <https://www.prover.com/references/>. The website's navigation bar includes the Prover logo and a menu with the following items: Solutions, References, Expertise, News, Company, and SDA Forum. Below the navigation bar is a dark blue horizontal bar with white text listing various locations: ALL, BELGIUM, CANADA, CHINA, ENGLAND, NEW YORK, NORWAY, PARIS, and STOCKHOLM. The main content area features three columns, each with an image and a text block. The first column shows a railway crossing with a blue train and a barrier, with the text 'Implementing Prover Trident for SL, Stockholm'. The second column shows a subway station with a 'Subway' sign and a train, with the text 'Formal Verification of SSI Software for NYCT, New York'. The third column shows a person sitting on a train with the Eiffel Tower in the background, with the text 'Our Formal Verification Solution for RATP, Paris'.

Applications Places

NS Unhar x REMS x It Robot x SP Start x SP byron x In Byron x SP AWS x Autor x SP Start x W Paris x Refer x

Secure | <https://www.prover.com/references/>

PROVER Solutions References Expertise News Company SDA Forum

ALL BELGIUM CANADA CHINA ENGLAND NEW YORK NORWAY PARIS STOCKHOLM

Implementing Prover Trident for SL, Stockholm

In this project, Prover Technology provides the Prover Trident solution to Ansaldo STS, for development and safety approval of interlocking software for Roslagsbanan, a mainline railway line that connects...

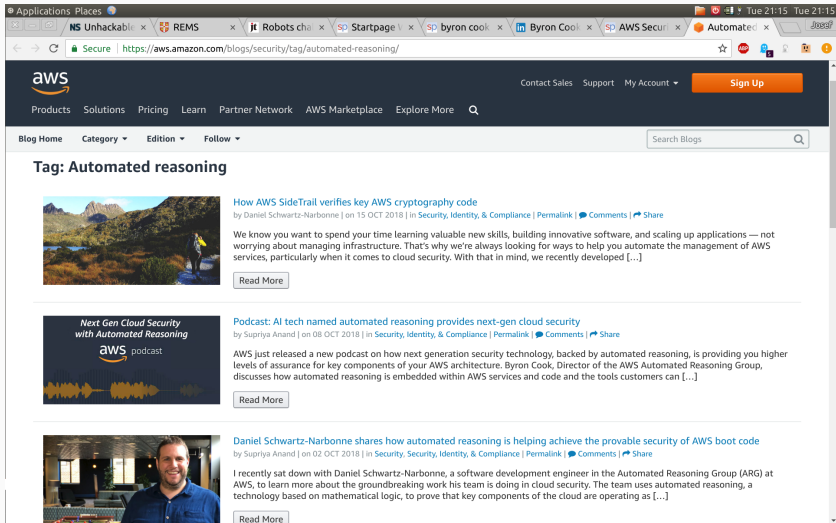
Formal Verification of SSI Software for NYCT, New York

New York City Transit (NYCT) is modernizing the signaling system in its subway by installing CBTC and replacing relay-based interlockings with computerized, solid state interlockings (SSIs).

Our Formal Verification Solution for RATP, Paris

In this project Prover Technology collaborated with RATP in creating a formal verification solution to meet RATP demand for safety verification of interlocking software. RATP had selected a computerized...

Today's Applications



The screenshot shows a web browser window with multiple tabs open, including 'NS Unhackable', 'REMS', 'Robots cha', 'Startpage', 'byron cook', 'Byron Cook', 'AWS Securi', 'Automated', and 'Jostif'. The address bar shows the URL 'https://aws.amazon.com/blogs/security/tag/automated-reasoning/'. The page header features the AWS logo and navigation links: 'Products', 'Solutions', 'Pricing', 'Learn', 'Partner Network', 'AWS Marketplace', and 'Explore More'. A search bar is located in the top right corner. The main content area is titled 'Tag: Automated reasoning' and contains three article entries:

- How AWS SideTrail verifies key AWS cryptography code**
by Daniel Schwartz-Narbonne | on 15 OCT 2018 | in Security, Identity, & Compliance | Permalink | Comments | Share
We know you want to spend your time learning valuable new skills, building innovative software, and scaling up applications — not worrying about managing infrastructure. That's why we're always looking for ways to help you automate the management of AWS services, particularly when it comes to cloud security. With that in mind, we recently developed [...]
[Read More](#)
- Podcast: AI tech named automated reasoning provides next-gen cloud security**
by Supriya Anand | on 08 OCT 2018 | in Security, Identity, & Compliance | Permalink | Comments | Share
AWS just released a new podcast on how next generation security technology, backed by automated reasoning, is providing you higher levels of assurance for key components of your AWS architecture. Byron Cook, Director of the AWS Automated Reasoning Group, discusses how automated reasoning is embedded within AWS services and code and the tools customers can [...]
[Read More](#)
- Daniel Schwartz-Narbonne shares how automated reasoning is helping achieve the provable security of AWS boot code**
by Supriya Anand | on 02 OCT 2018 | in Security, Security, Identity, & Compliance | Permalink | Comments | Share
I recently sat down with Daniel Schwartz-Narbonne, a software development engineer in the Automated Reasoning Group (ARG) at AWS, to learn more about the groundbreaking work his team is doing in cloud security. The team uses automated reasoning, a technology based on mathematical logic, to prove that key components of the cloud are operating as [...]
[Read More](#)

Today's Applications

Applications Places

Secure | <https://www.absint.com/compcert/>

Absint Products Support News About us Contact Search

CompCert How it works New in 18.10 Try now

Formally verified compilation

CompCert is a formally verified optimizing C compiler. Its intended use is compiling safety-critical and mission-critical software written in C and meeting high levels of assurance. It accepts most of the ISO C 99 language, with some exceptions and a few extensions. It produces machine code for ARM, PowerPC, x86, and RISC-V architectures.

What sets CompCert apart?

CompCert is the only production compiler that is formally verified, using machine-assisted mathematical proofs, to be exempt from miscompilation issues. The code it produces is proved to behave exactly as specified by the semantics of the source C program.

This level of confidence in the correctness of the compilation process is unprecedented and contributes to meeting the highest levels of software assurance.

The formal proof covers [all transformations](#) from the abstract syntax tree to the generated assembly code. To preprocess and

serveimage.jpeg

Show all

Today's Applications



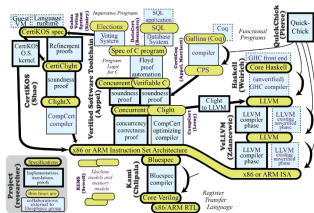
the science of deep specification

DeepSpec is an Expedition in Computing funded by the National Science Foundation.

We focus on the specification and verification of full functional correctness of software and hardware.

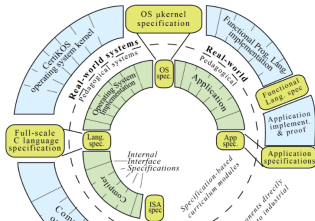
Research

We have several major research projects, and our ambitious goal is to connect them at specification interfaces to prove end-to-end correctness of whole systems.



Education

To deliver secure and reliable products, the software industry of the future needs engineers trained in specification and verification. We'll produce that curriculum.



Today's Applications

PHYS.ORG Nanotechnology ▾ Physics ▾ Earth ▾ Astronomy & Space ▾ Technology ▾ Chemistry ▾ Biology ▾ Other Sciences ▾


f t r e m

search 🔍 👤

Home > Other Sciences > Mathematics > October 12, 2012

Six-year journey leads to proof of Feit-Thompson Theorem

October 12, 2012 by Rob Kries, Microsoft




Georges Gonthier.


At 5:46 p.m. on Sept. 20, Georges Gonthier, principal researcher at Microsoft Research Cambridge, sent a brief email to his colleagues at the Microsoft Research-Inria Joint Centre in Paris. It read, in full: "This is really the End."


Those five innocuous words heralded the culmination of a project that had consumed more than six years and resulted in the formal proof of the Feit-Thompson Theorem, the first major step of the classification of finite simple groups.


The theorem, first proved by Walter Feit and John Griggs Thompson in 1963 and also known as the Odd-Order Theorem, states that in mathematical group theory, every finite group of odd order is solvable.


Featured Last comments Popular

 Gaia spots a 'ghost' galaxy next door 19 hours ago 81

 How plants evolved to make ants their servants Nov 12, 2018 21

 Physicists build fractal shape out of electrons Nov 12, 2018 0

 Dark matter 'hurricane' offers chance to detect axions 18 hours ago 36

 How to drive a robot on Mars Nov 12, 2018 2

Outline

Motivation, Learning vs. Reasoning

Computer Understandable (Formal) Math

Bird's-Eye View of ATP and ML

Learning of Theorem Proving - Overview

Demos

High-level Reasoning Guidance: Premise Selection

Low Level Guidance of Theorem Provers

Mid-level Reasoning Guidance

Synthesis and Autoformalization

What Are Automated Theorem Provers?

- Computer programs that (try to) automatically determine if
 - A conjecture C is a logical consequence of a set of axioms Ax
 - The derivation of conclusions that follow inevitably from facts.
- Systems: Vampire, E, SPASS, Prover9, Z3, CVC4, Satallax, iProver, ...
- Brute-force search calculi (resolution, superposition, tableaux, inst-gen)
- **more limited logics**: SAT, QBF, SMT, UEQ, ... (DPLL, CDCL, ...)
- **TP-motivated PLs**: Prolog (*logic programming* - Hayes, Kowalski)
- Human-designed heuristics for pruning of the search space
- Combinatorial explosion on large KBs like Flyspeck and Mizar
- Need to be equipped with good domain-specific inference guidance ...
- ... and that is what I try to do ...
- ... typically by **learning** in various ways from large TP corpora ...

Propositional – SAT-satisfiability solving

- DPLL- Davis–Putnam–Logemann–Loveland algorithm
- choosing a literal
- assigning a truth value to it
- simplifying the formula
- recursively check if the simplified formula is satisfiable
- Unit propagation
- Pure literal elimination
- **Clause learning** - the CDCL revolution (J. Marques-Silva 1996)
- John Harrison in 2005: *"People now say that a problem is **NP-easy**"*
- Basis of many more-involved algorithms, hardware checking, model checking, etc.

Satisfiability Modulo Theories – SMT

- adds **theories** like arithmetics, bit-arrays, etc.
- works like SAT, but simplifies the theory literals whenever possible
- very useful for software and hardware verification
- a lot of development in the recent decade
- today also limited treatment of quantifiers (first-order logic):
- often incomplete for first-order logic
- related to complete *instantiation-based methods*:
- instantiate first-order terms by guessing their instances
- **cross-fertilization and competition with ATPs**

First Order – Automated Theorem Proving (ATP)

- try to infer conjecture C from axioms Ax : $Ax \vdash C$
- most classical methods proceed by **refutation**: $Ax \wedge \neg C \vdash \perp$
- $Ax \wedge \neg C$ are turned into *clauses*: universally quantified disjunctions of atomic formulas and their negations
- *skolemization* is used to remove existential quantifiers
- strongest methods: resolution (generalized modus ponens) on clauses:
- $\neg man(X) \vee mortal(X), man(socrates) \vdash mortal(socrates)$
- resolution/superposition (equational) provers generate inferences, looking for the contradiction (empty clause)
- tableaux, connection calculus
- instantiation-based – systematically add ground instances and use SAT solvers to check satisfiability
- combined approaches - SAT run often inside the ATP (generalized splitting)

The CADE ATP System Competition (CASC)

Higher-order Theorems	Zipperpi 2.8	Satallax 3.4	Satallax 3.5	Vampire 4.5	Leo-III 1.5	CVC4 1.8	LEO-II 1.7.0						
Solved ₅₀₀	424 ₅₀₀	323 ₅₀₀	319 ₅₀₀	299 ₅₀₀	287 ₅₀₀	194 ₅₀₀	112 ₅₀₀						
Solutions	424 84%	323 64%	319 63%	299 59%	287 57%	194 38%	111 22%						
Typed First-order Theorems +*_/	Vampire 4.5	Vampire 4.4	CVC4 1.8										
Solved ₂₅₀	191 ₂₅₀	190 ₂₅₀	187 ₂₅₀										
Solutions	191 76%	190 76%	187 74%										
First-order Theorems	Vampire 4.5	Vampire 4.4	Enigma 9.5.1	E 2.5	CSE_E 1.2	iProver 3.3	GKC 0.5.1	CVC4 1.8	Zipperpi 2.0	Etableau 0.2	Prover9 1109a	CSE 1.3	leanCo 2.2
Solved ₅₀₀	429 ₅₀₀	416 ₅₀₀	401 ₅₀₀	351 ₅₀₀	316 ₅₀₀	312 ₅₀₀	289 ₅₀₀	275 ₅₀₀	237 ₅₀₀	162 ₅₀₀	146 ₅₀₀	124 ₅₀₀	111 ₅₀₀
Solutions	429 85%	416 83%	401 80%	351 70%	316 63%	312 62%	289 57%	275 55%	237 47%	162 32%	146 29%	124 24%	111 22%
First-order Non-theorems	Vampire SAT-4.5	Vampire SAT-4.4	iProver SAT-3.3	CVC4 SAT-1.8	E FNT-2.5	PyRes 1.3							
Solved ₂₅₀	238 ₂₅₀	226 ₂₅₀	182 ₂₅₀	98 ₂₅₀	63 ₂₅₀	13 ₂₅₀							
Solutions	238 95%	226 90%	182 72%	98 39%	63 25%	13 5%							
Unit Equality CNF	E 2.5	Type 2.2.1	E 2.4	Vampire 4.5	Etableau 0.2	GKC 0.5.1	iProver 3.3	lazyCoP 0.1					
Solved ₂₅₀	202 ₂₅₀	197 ₂₅₀	185 ₂₅₀	162 ₂₅₀	148 ₂₅₀	128 ₂₅₀	124 ₂₅₀	20 ₂₅₀					
Solutions	202 80%	197 78%	185 74%	162 64%	148 59%	128 51%	124 49%	0 0%					
Large Theory Batch Problems	MaLARE 0.5	E LTB-2.5	iProver LTB-3.3	Zipperpi LTB-2.0	Leo-III LTB-1.5	ATPBoost 1.0	GKC LTB-0.5.1	Leo-III LTB-1.4					
Solved ₁₀₀₀₀	7054 ₁₀₀₀₀	3393 ₁₀₀₀₀	3164 ₁₀₀₀₀	1699 ₁₀₀₀₀	1413 ₁₀₀₀₀	1237 ₁₀₀₀₀	493 ₁₀₀₀₀	134 ₁₀₀₀₀					
Solutions	7054 70%	3393 33%	3163 31%	1699 16%	1413 14%	1237 12%	493 4%	134 1%					

Using First/Higher Order Automated Theorem Proving

- 1996: Bill McCune proof of Robbins conjecture (Robbins algebras are Boolean algebras)
- Robbins conjecture unsolved for 50 years by mathematicians like Tarski
- 2021: M. Kinyon, R. Veroff, Prover9: Weak AIM conjecture
- If Q is an Abelian Inner Mapping loop, then Q is nilpotent of class ≤ 3 .
- ATP has currently **only limited use for proving new conjectures**
- mainly in very specialized algebraic domains
- however ATP has become very useful in **Interactive Theorem Proving**
- a recent (2020) **performance jump in higher-order ATP**:
- Zipperposition, HO-Vampire, E-HO (J. Blanchette, A Bentkamp, P. Vukmirovic)

Learning Approaches - Data vs Theory Driven

- John Shawe-Taylor and Nello Cristianini – Kernel Methods for Pattern Analysis (2004):
- *"Many of the most interesting problems in AI and computer science in general are extremely complex often making it **difficult or even impossible to specify an explicitly programmed solution.**"*
- *"As an example consider the problem of recognising genes in a DNA sequence. We do not know how to specify a program to pick out the subsequences of, say, human DNA that represent genes."*
- *"Similarly we are not able directly to program a computer to recognise a face in a photo."*

Learning Approaches - Data vs Theory Driven

- *"Learning systems offer an alternative methodology for tackling these problems."*
- *"By exploiting the knowledge extracted from a sample of data, they are often capable of adapting themselves to infer a solution to such tasks."*
- *"We will call this alternative approach to software design the **learning methodology**."*
- *"It is also referred to as the **data driven** or **data based** approach, in contrast to the **theory driven** approach that gives rise to precise specifications of the required algorithms."*

For Fun: My Depressive Slide From 2011 AMS

- My personal puzzle:
- The year is 2011.
- The recent AI successes are data-driven, not theory-driven.
- Ten years after the success of Google.
- Fifteen years after the success of Deep Blue with Kasparov.
- Five year after a car drove autonomously across the Mojave desert.
- Four years after the Netflix prize was announced.
- *Why am I still the only person training AI systems on large repositories of human proofs like the Mizar library???*
- (This finally started to change in 2011)

Sample of Learning Approaches

- **neural networks** (**statistical ML**) – backpropagation, deep learning, convolutional, recurrent, attention/transformers, tree NNs, graph NNs, etc.
- **decision trees, random forests, gradient boosted trees** – find good classifying attributes (and/or their values); more **explainable**
- **support vector machines** – find a good classifying hyperplane, possibly after non-linear transformation of the data (*kernel methods*)
- **k-nearest neighbor** – find the k nearest neighbors to the query, combine their solutions
- **naive Bayes** – compute probabilities of outcomes assuming complete (naive) independence of characterizing features (just multiplying probabilities)
- **inductive logic programming** (**symbolic ML**) – generate logical explanation (program) from a set of ground clauses by generalization
- **genetic algorithms** – evolve large population by crossover and mutation
- various **combinations** of statistical and symbolic approaches
- **supervised, unsupervised, reinforcement** learning (actions, explore/exploit, cumulative reward)

Learning – Features and Data Preprocessing

- **Extremely important** - if irrelevant, there is no way to learn the function from input to output (“garbage in garbage out”)
- **Feature discovery/engineering** – a big field
- **Deep Learning** – design neural architectures that **automatically find important high-level features** for a task
- **Data Augmentation and Selection** – how do we generate/select more/better data to learn on?
- **Latent Semantics, dimensionality reduction**: use linear algebra (eigenvector decomposition) to discover the most similar features, make approximate equivalence classes from them
- **word2vec and related methods**: represent words/sentences by *embeddings* (in a high-dimensional real vector space) learned by predicting the next word on a large corpus like Wikipedia
- **math and theorem proving**: syntactic/semantic/computational patterns/abstractions/programs
- how do we represent math objects (formulas, proofs, ideas) in our mind?

Future: AITP Challenges/Bets from 2014

- 3 AITP bets from my 2014 talk at Institut Henri Poincare
 - In 20 years, 80% of Mizar and Flyspeck toplevel theorems will be provable automatically (same hardware, same libraries as in 2014 - about 40% then)
 - In 10 years: 60% (**DONE** already in 2021 - 3 years ahead of schedule)
 - In 25 years, 50% of the toplevel statements in LaTeX-written Msc-level math curriculum textbooks will be **parsed automatically** and with correct formal semantics (this may be **faster** than I expected)
- My (conservative?) estimate when we will do **Fermat**:
 - Human-assisted formalization: by 2050
 - Fully automated proof (hard to define precisely): by 2070
 - See the Foundation of Math thread: <https://bit.ly/300k9Pm>
 - and the AITP'22 panel: <https://bit.ly/3dcY5HW>
- Big challenge: Learn complicated **symbolic algorithms** (not black box - motivates also our OEIS research)

Outline

Motivation, Learning vs. Reasoning

Computer Understandable (Formal) Math

Bird's-Eye View of ATP and ML

Learning of Theorem Proving - Overview

Demos

High-level Reasoning Guidance: Premise Selection

Low Level Guidance of Theorem Provers

Mid-level Reasoning Guidance

Synthesis and Autoformalization

Using Learning to Guide Theorem Proving

- **high-level**: pre-select lemmas from a large library, give them to ATPs
- **high-level**: pre-select a good ATP strategy/portfolio for a problem
- **high-level**: pre-select good *hints* for a problem, use them to guide ATPs
- **low-level**: guide every inference step of ATPs (tableau, superposition)
- **low-level**: guide every kernel step of LCF-style ITPs
- **mid-level**: guide application of tactics in ITPs, learn new tactics
- **mid-level**: invent suitable strategies/procedures for classes of problems
- **mid-level**: invent suitable conjectures for a problem
- **mid-level**: invent suitable concepts/models for problems/theories
- **proof sketches**: explore stronger/related theories to get proof ideas
- **theory exploration**: develop interesting theories by conjecturing/proving
- **feedback loops**: (dis)prove, learn from it, (dis)prove more, learn more, ...
- **autoformalization**: (semi-)automate translation from \LaTeX to formal
- ...

Large Datasets

- Mizar / MML / MPTP – since 2003
- MPTP Challenge (2006), MPTP2078 (2011), Mizar40 (2013)
- Isabelle (and AFP) – since 2005
- Flyspeck (including core HOL Light and Multivariate) – since 2012
- HOL4 – since 2014, CakeML – 2017, GRUNGE – 2019
- Coq – since 2013/2016
- ACL2 – 2014?
- Lean?, Stacks?, Arxiv?, ProofWiki?, ...

Outline

Motivation, Learning vs. Reasoning

Computer Understandable (Formal) Math

Bird's-Eye View of ATP and ML

Learning of Theorem Proving - Overview

Demos

High-level Reasoning Guidance: Premise Selection

Low Level Guidance of Theorem Provers

Mid-level Reasoning Guidance

Synthesis and Autoformalization

AI/TP Examples and Demos

- **ENIGMA/hammer proofs of Pythagoras** : <https://bit.ly/2MVPAn7>
(more at <http://grid01.ciirc.cvut.cz/~mptp/enigma-ex.pdf>) and
simplified Carmichael <https://bit.ly/3oGBdRz>,
- **3-phase ENIGMA**: <https://bit.ly/3C0Lwa8>,
<https://bit.ly/3BWqR6K>
- **Long trig proof from 1k axioms**: <https://bit.ly/2YZ0OgX>
- **Extreme Deepire/AVATAR proof of $\epsilon_0 = \omega^{\omega^{\dots}}$** <https://bit.ly/3Ne4WNX>
- **Hammering demo**: <http://grid01.ciirc.cvut.cz/~mptp/out4.ogv>
- **TacticToe on HOL4**:
http://grid01.ciirc.cvut.cz/~mptp/tactictoe_demo.ogv
- **TacticToe longer**: <https://www.youtube.com/watch?v=BO4Y8ynwT6Y>
- **Tactician for Coq**:
<https://blaaubroek.eu/papers/cicm2020/demo.mp4>,
<https://coq-tactician.github.io/demo.html>
- **Inf2formal over HOL Light**:
<http://grid01.ciirc.cvut.cz/~mptp/demo.ogv>
- **QSynt: AI rediscovers the Fermat primality test**:
<https://www.youtube.com/watch?v=24oejR9wsXs>

Outline

Motivation, Learning vs. Reasoning

Computer Understandable (Formal) Math

Bird's-Eye View of ATP and ML

Learning of Theorem Proving - Overview

Demos

High-level Reasoning Guidance: Premise Selection

Low Level Guidance of Theorem Provers

Mid-level Reasoning Guidance

Synthesis and Autoformalization

High-level ATP guidance: Premise Selection

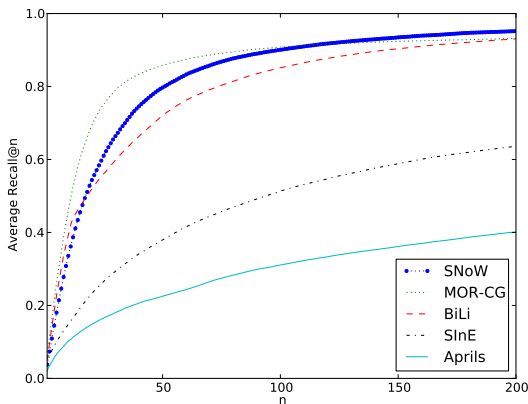
- 2003: Can existing ATPs be used over the large Mizar library?
- About 80000 nontrivial math facts at that time – impossible to use them all
- Is good premise selection for proving a new conjecture possible at all?
- Or is it a mysterious power of mathematicians? (Penrose)
- Today: Premise selection is not a mysterious property of mathematicians!
- Reasonably good algorithms started to appear (more below).

First system: Mizar Proof Advisor (2003)

- train naive-Bayes fact selection on all previous Mizar/MML proofs (50k)
- **input features**: conjecture symbols; **output labels**: names of facts
- recommend relevant facts when proving new conjectures
- give them to unmodified FOL ATPs
- possibly reconstruct inside the ITP afterwards (lots of work)
- First results over the whole Mizar library in 2003:
 - about **70% coverage** in the first 100 recommended premises
 - chain the recommendations with strong ATPs to get full proofs
 - about **14%** of the Mizar theorems were then automatically provable (SPASS)
- Compare with 2016 methods: about **40-45%** automatically provable

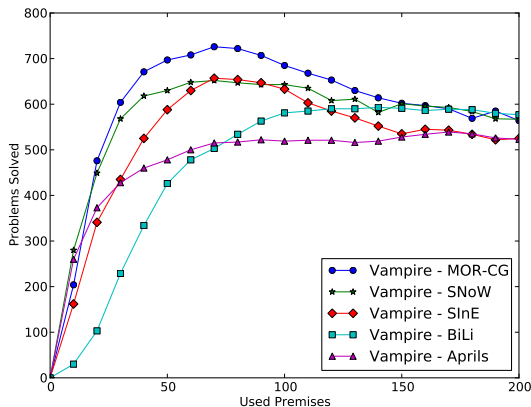
ML Evaluation of methods on MPTP2078 – recall

- Coverage (recall) of facts needed for the Mizar proof in first n predictions
- MOR-CG – kernel-based, SNoW - naive Bayes, BiLi - bilinear ranker
- SInE, Aprils - heuristic (non-learning) fact selectors

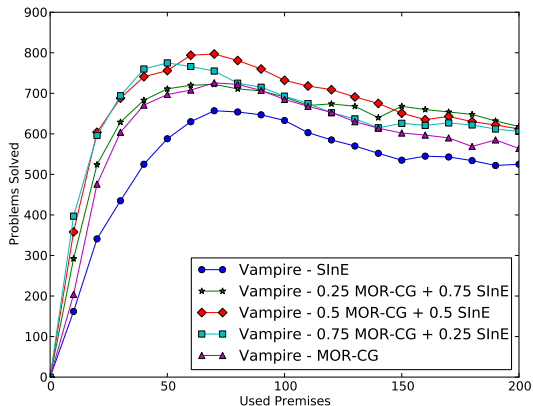


ATP Evaluation of methods on MPTP2078

- Number of the problems proved by ATP when given n best-ranked facts
- Good machine learning on previous proofs really matters for ATP!



Combined (ensemble) methods on MPTP2078

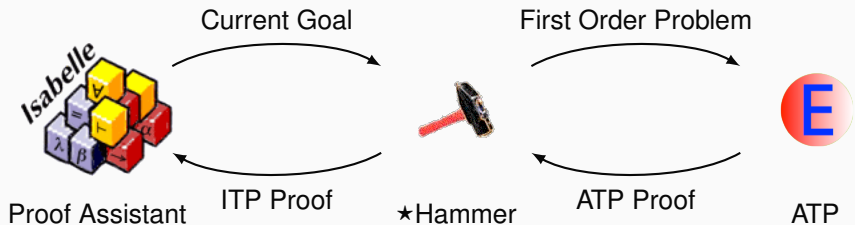


Large 2014 Evaluation on MML – 60k theorems

14 most covering (40.6%) ML/ATP methods ordered by greedy coverage

Method	Parameters	Premis.	ATP	°-SOTAC	Theorem (%)	Greedy (%)
comb	min_2k_20_20	128	Epar	1728.34	15789 (27.3)	15789 (27.2)
lsi	3200ti_8_80	128	Epar	1753.56	15561 (26.9)	17985 (31.0)
comb	qua_2k_k200_33_33	512	Epar	1520.73	13907 (24.0)	19323 (33.4)
knn	is_40	96	Z3	1634.50	11650 (20.1)	20388 (35.2)
nb	idf010	128	Epar	1630.77	14004 (24.2)	21057 (36.4)
knn	is_80	1024	V	1324.39	12277 (21.2)	21561 (37.2)
geo	r_99	64	V	1357.58	11578 (20.0)	22006 (38.0)
comb	geo_2k_50_50	64	Epar	1724.43	14335 (24.8)	22359 (38.6)
comb	geo_2k_60_20	1024	V	1361.81	12382 (21.4)	22652 (39.1)
comb	har_2k_k200_33_33	256	Epar	1714.06	15410 (26.6)	22910 (39.6)
geo	r_90	256	V	1445.18	13850 (23.9)	23107 (39.9)
lsi	3200ti_8_80	128	V	1621.11	14783 (25.5)	23259 (40.2)
comb	geo_2k_50_00	96	V	1697.10	15139 (26.1)	23393 (40.4)
geo	r_90	256	Epar	1415.48	14093 (24.3)	23478 (40.6)

Today's AI-ATP systems (★-Hammers)



How much can it do?

- Mizar / MML – MizAR
- Isabelle (Auth, Jinja) – Sledgehammer
- Flyspeck (including core HOL Light and Multivariate) – HOL(y)Hammer
- HOL4 (Gauthier and Kaliszyk)
- CoqHammer (Czajka and Kaliszyk) - about 40% on Coq standard library
≈ 40-45% success by 2016, 60% on Mizar as of 2021

Various Improvements and Additions

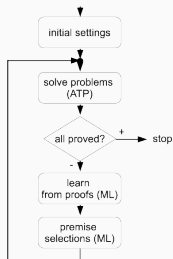
- Model-based features for **semantic similarity** [IJCAR'08]
- Features encoding **term matching/unification** [IJCAI'15]
- Stronger learners: SVMs, weighted k-NN, boosted trees (XGBoost)
- **Matching and transferring concepts** and theorems between libraries (Gauthier & Kaliszyk) – allows “superhammers”, conjecturing, and more
- **Lemmatization** – extracting and considering millions of low-level lemmas
- LSI, word2vec, first neural models, definitional embeddings (with Google)
- Combined with tactical search and MCTS: **TacticToe** (Gauthier, 2017)
- Learning in binary setting from many alternative proofs
- Negative/positive mining (ATPBoost - Piotrowski & JU, 2018)
- Stateful neural methods: RNNs and Transformers (Piotrowski & JU, 2020)
- **Currently strongest**: Property-invariant graph neural networks (Olsak, 2020)

Summary of Features Used

- From **syntactic** to more **semantic**:
- Constant and function symbols
- Walks in the term graph
- Walks in clauses with polarity and variables/skolems unified
- Subterms, de Bruijn normalized
- Subterms, all variables unified
- Matching terms, no generalizations
- terms and (some of) their generalizations
- Substitution tree nodes
- All unifying terms
- LSI/PCA, word2vec, fasttext, etc.
- Neural embeddings: CNN, RNN, Tree NN, Graph CNN, ...
- Evaluation in a large set of (finite) models
- Vectors of proof similarities (proof search hidden states)
- Vectors of problems solved (for ATP strategies)

High-level feedback loops – MALARea, ATPBoost

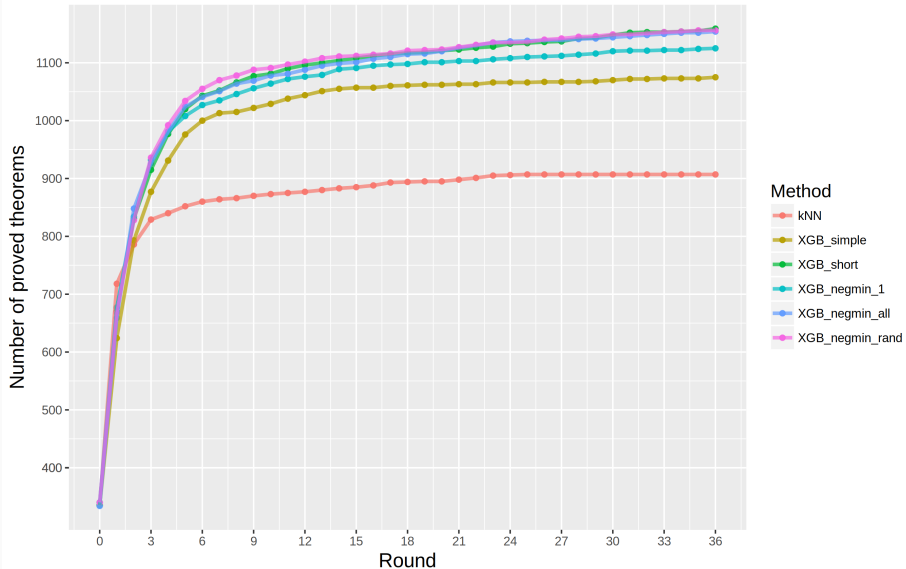
- Machine Learner for Autom. Reasoning (2006) – infinite hammering
- feedback loop interleaving ATP with learning premise selection
- both syntactic and **semantic** features for characterizing formulas:
- evolving set of finite (counter)models in which formulas evaluated
- winning AI/ATP benchmarks (MPTPChallenge, CASC 08/12/13/18/20)
- ATPBoost (Piotrowski) - recent incarnation focusing on multiple proofs



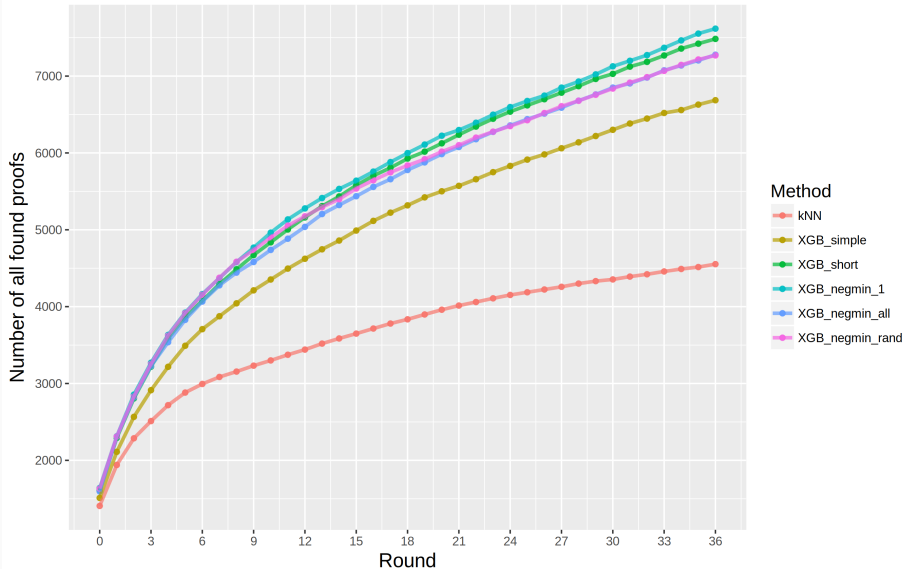
The screenshot shows a browser window with the URL tptp.org/CASC/110/WWWFiles/DivisionSummary1.html. The table displays performance metrics for various ATP solvers on the Large Theory Batch Problems.

Large Theory Batch Problems	MaLARE	E	Prover	Zipperpi	Leo-III	ATPBoost	GKC	Leo-III
	8.9	LTB-2.5	LTB-1.1	LTB-2.0	LTB-1.5	1.0	LTB-0.5.1	LTB-1.4
Solved ₁₀₀₀₀	7054 ₁₀₀₀₀	3393 ₁₀₀₀₀	3164 ₁₀₀₀₀	1699 ₁₀₀₀₀	1413 ₁₀₀₀₀	1237 ₁₀₀₀₀	493 ₁₀₀₀₀	134 ₁₀₀₀₀
Solutions	7054 70%	3393 33%	3163 31%	1699 16%	1413 14%	1237 12%	493 4%	134 1%

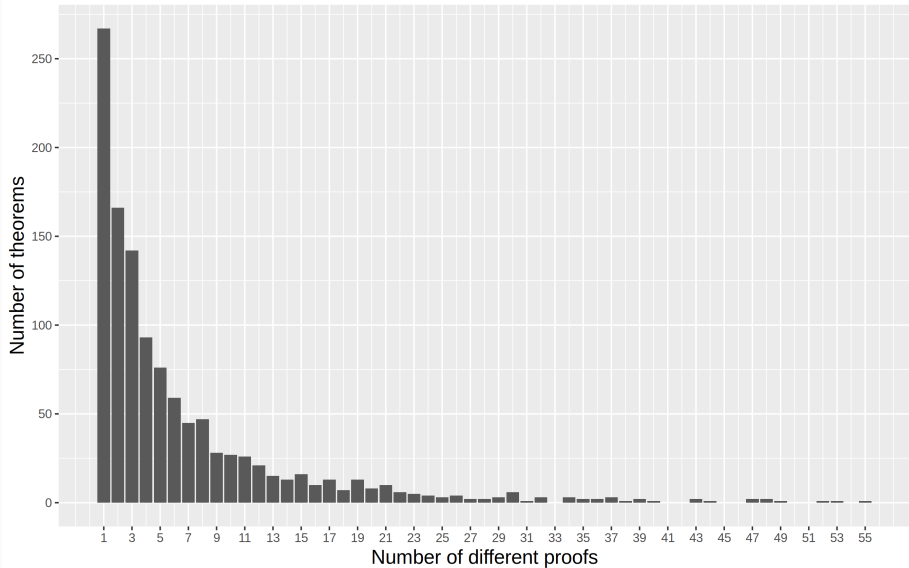
Prove-and-learn loop on MPTP2078 data set



Prove-and-learn loop on MPTP2078 data set



Number of found proofs per theorem at the end of the loop



ATPBoost – Binary settings

There are two possible settings in which we can approach premise selection with machine learning:

- 1 *multilabel setting*: here we treat premises used in the proofs as opaque labels on theorems and we train a model capable of labeling conjectures based on their features,
- 2 *binary setting*: here the aim of the learning model is to recognize pairwise-relevance of the *conjecture-premise* pairs, i.e. to decide what is the chance of the premise being relevant for proving the conjecture based on the features of both the conjecture and the premise.

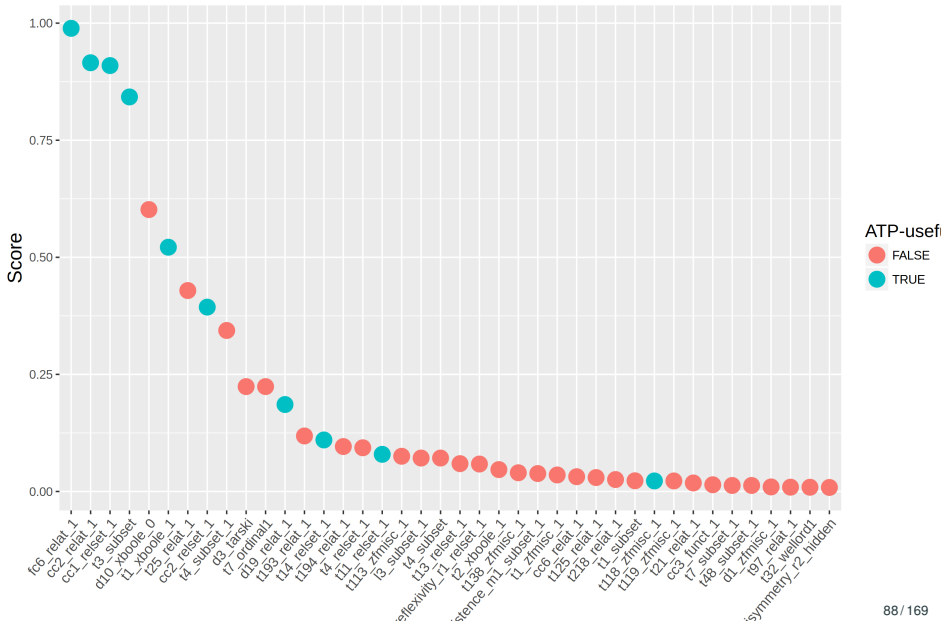
The first approach is more accessible and was more used so far. The second setting, though, is more general and better for modern ML algorithms.

- Positive and negative examples for training set were initially generated from the theorems with proofs in the following way:
 - as positives we take pairs (*theorem-premise*) if premise appears in at least one proof of the *theorem*,
 - negatives are randomly taken from the set of pairs (*theorem-premise*) where the *premise* is available for the *theorem* but there is no ATP-proof of the *theorem* with this *premise*.

Every such pair is presented to the ML algorithm as concatenation of its feature representation labeled by 0 or 1.

- After a predictor is trained, we use it to create ranking of premises available for theorem from the outside of training set.

Ranking for theorem t17_relat_1



MaLAREa-SG1 (2008): ML plus Semantics

- Run **finite countermodel finder** (Paradox, Mace, ...) before ATPs when it makes sense
- Paradox used when there are less than 64 axioms (and low time limit)
- Detects countersatisfiability much more often and much faster than ATPs on small problems
- Thousands of (typically different) models usually found in MaLAREa runs
- Creating a database of interesting models relevant for the large theory
- The **model database usable for further purposes**

Combining ML with Semantic Selection in Malarea

- Use the semantic information for **updating axiom relevance**
- All formulas from the large theory are evaluated in the models found by model finders
- Heuristically, axiom A is more useful for a negated conjecture $\neg C$ if it excludes more models of $\neg C$
- Also, the more rare the exclusion of a certain model of $\neg C$, the more valuable is the axiom
- **Invalidity in each model is another feature characterizing formulas ...**
- ... So use it for machine learning as other features!
- This works in the Bayesian framework exactly the same as e.g. symbol-based similarity:
- **Axiom sharing a rare countermodel with a conjecture is promoted** in the same way as an axiom sharing a rare symbol with the conjecture

Semantic features in MaLAREa-SG1 - 2008 Evaluation

- Chainy division of the 2006 MPTP Challenge: 252 related problems
- Average size 400 formulas, 1500 formulas in total
- 21 hours overall timelimit
- The base ATPs (E,SPASS): 80 - 90 problems each with 300s
- 104 together (if each 64s for each problem)
- We add **term-structure features** (TS) and **semantic guidance** (SG)

Table: Full run of three versions of MaLAREa on the MPTP Challenge

description	total solved	solved in 21 hours	passes in 21 hours	total passes	last pass with success
old	149	144	114	234	208
with TS	154	149	148	263	237
with TS & SG	165	161	121	195	170

FACE_OF_POLYHEDRON_POLYHEDRON

```
let FACE_OF_POLYHEDRON_POLYHEDRON = prove
('!s:real^N->bool c. polyhedron s /\ c face_of s ==> polyhedron c',
 REPEAT STRIP_TAC THEN FIRST_ASSUM
 (MP_TAC o GEN_REWRITE_RULE I [POLYHEDRON_INTER_AFFINE_MINIMAL]) THEN
 REWRITE_TAC[RIGHT_IMP_EXISTS_THM; SKOLEM_THM] THEN
 SIMP_TAC[LEFT_IMP_EXISTS_THM; RIGHT_AND_EXISTS_THM; LEFT_AND_EXISTS_THM] THEN
 MAP_EVERY X_GEN_TAC
 ['f:(real^N->bool)->bool'; 'a:(real^N->bool)->real^N';
 'b:(real^N->bool)->real'] THEN
 STRIP_TAC THEN
 MP_TAC(ISPECL ['s:real^N->bool'; 'f:(real^N->bool)->bool';
 'a:(real^N->bool)->real^N'; 'b:(real^N->bool)->real']
 FACE_OF_POLYHEDRON_EXPLICIT) THEN
 ANTS_TAC THENL [ASM_REWRITE_TAC[] THEN ASM_MESON_TAC[]; ALL_TAC] THEN
 DISCH_THEN(MP_TAC o SPEC 'c:real^N->bool') THEN ASM_REWRITE_TAC[] THEN
 ASM_CASES_TAC 'c:real^N->bool = {}' THEN
 ASM_REWRITE_TAC[POLYHEDRON_EMPTY] THEN
 ASM_CASES_TAC 'c:real^N->bool = s' THEN ASM_REWRITE_TAC[] THEN
 DISCH_THEN SUBST1_TAC THEN MATCH_MP_TAC POLYHEDRON_INTERS THEN
 REWRITE_TAC[FORALL_IN_GSPEC] THEN
 ONCE_REWRITE_TAC[SIMPLE_IMAGE_GEN] THEN
 ASM_SIMP_TAC[FINITE_IMAGE; FINITE_RESTRICT] THEN
 REPEAT STRIP_TAC THEN REWRITE_TAC[IMAGE_ID] THEN
 MATCH_MP_TAC POLYHEDRON_INTER THEN
 ASM_REWRITE_TAC[POLYHEDRON_HYPERPLANE]);;
```

FACE_OF_POLYHEDRON_POLYHEDRON

```
polyhedron s /\ c face_of s ==> polyhedron c
```

HOL Light proof: could not be re-played by ATPs.

Alternative proof found by a hammer based on `FACE_OF_STILLCONVEX`:
Face t of a convex set s is equal to the intersection of s with the affine hull of t .

```
FACE_OF_STILLCONVEX:
```

```
!s t:real^N->bool. convex s ==>
```

```
(t face_of s <=>
```

```
t SUBSET s /\ convex(s DIFF t) /\ t = (affine hull t) INTER s)
```

```
POLYHEDRON_IMP_CONVEX:
```

```
!s:real^N->bool. polyhedron s ==> convex s
```

```
POLYHEDRON_INTER:
```

```
!s t:real^N->bool. polyhedron s /\ polyhedron t
```

```
==> polyhedron (s INTER t)
```

```
POLYHEDRON_AFFINE_HULL:
```

```
!s. polyhedron(affine hull s)
```

Outline

Motivation, Learning vs. Reasoning

Computer Understandable (Formal) Math

Bird's-Eye View of ATP and ML

Learning of Theorem Proving - Overview

Demos

High-level Reasoning Guidance: Premise Selection

Low Level Guidance of Theorem Provers

Mid-level Reasoning Guidance

Synthesis and Autoformalization

Low-level: Statistical Guidance of Connection Tableau

- learn guidance of every clausal inference in connection tableau (leanCoP)
- set of first-order clauses, *extension* and *reduction* steps
- proof finished when all branches are closed
- a lot of nondeterminism, requires backtracking
- *Iterative deepening* used in leanCoP to ensure completeness
- good for learning – the tableau compactly represents the proof state

Clauses:

$$c_1 : P(x)$$

$$c_2 : R(x, y) \vee \neg P(x) \vee Q(y)$$

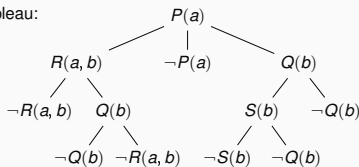
$$c_3 : S(x) \vee \neg Q(b)$$

$$c_4 : \neg S(x) \vee \neg Q(x)$$

$$c_5 : \neg Q(x) \vee \neg R(a, x)$$

$$c_6 : \neg R(a, x) \vee Q(x)$$

Closed Connection Tableau:



leanCoP: Minimal FOL Theorem Prover

```
% prove (Cla , Path , PathLim , Lem , Set)
prove ([ Lit | Cla ] , Path , PathLim , Lem , Set) :-
  ( - NegLit = Lit ; - Lit = NegLit ) ->
  (
    member (NegL , Path) ,
    unify_with_occurs_check (NegL , NegLit)
  ;
    % main nondeterminism
    lit (NegLit , NegL , Cla1 , Grnd1) ,
    unify_with_occurs_check (NegL , NegLit) ,
    prove (Cla1 , [ Lit | Path ] , PathLim , Lem , Set)
  ) ,
  prove (Cla , Path , PathLim , Lem , Set) .
prove ([ ] , _ , _ , _ , _) .
```


Statistical Guidance of Connection Tableau

- **MaLeCoP** (2011): first prototype Machine Learning Connection Prover
- extension rules chosen by naive Bayes trained on good decisions
- training examples: tableau features plus the name of the chosen clause
- initially slow: off-the-shelf learner 1000 times slower than raw leanCoP
- 20-time search shortening on the MPTP Challenge
- second version: 2015, with C. Kaliszyk
- both prover and naive Bayes in OCAML, fast indexing
- Fairly Efficient MaLeCoP = **FEMaLeCoP**
- 15% improvement over untrained leanCoP on the MPTP2078 problems
- using iterative deepening - enumerate shorter proofs before longer ones

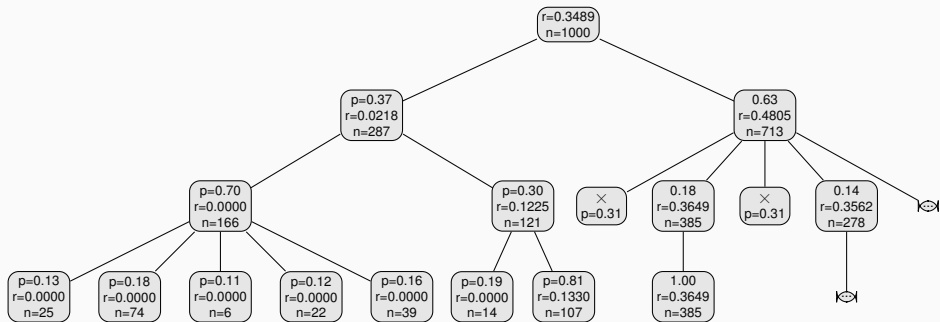
Statistical Guidance of Connection Tableau – rICoP

- 2018: stronger learners via C interface to OCAML (boosted trees)
- **remove iterative deepening**, the prover can go arbitrarily deep
- added Monte-Carlo Tree Search (MCTS)
- MCTS search nodes are sequences of clause application
- a good heuristic to **explore new vs exploit** good nodes:

$$\frac{w_i}{n_i} + c \cdot p_i \cdot \sqrt{\frac{\ln N}{n_i}} \quad (\text{UCT - Kocsis, Szepesvari 2006})$$

- learning both *policy* (clause selection) and *value* (state evaluation)
- clauses represented not by names but also by features (generalize!)
- **binary** learning setting used: | proof state | clause features |
- mostly term walks of length 3 (trigrams), hashed into small integers
- many iterations of proving and learning

Tree Example



Statistical Guidance of Connection Tableau – rICoP

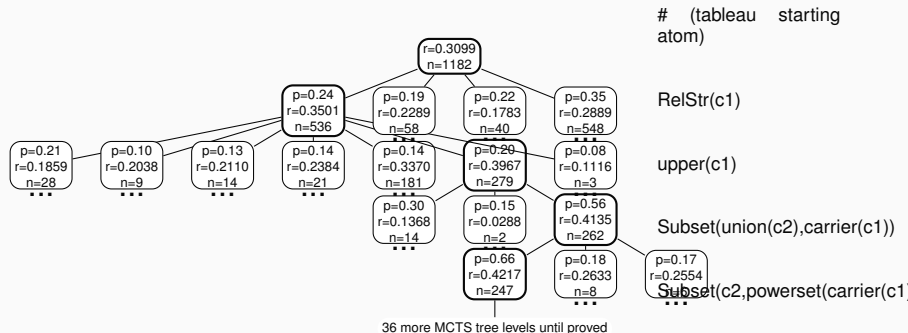
- On 32k Mizar40 problems using 200k inference limit
- nonlearning CoPs:

System	leanCoP	bare prover	rICoP no policy/value (UCT only)
Training problems proved	10438	4184	7348
Testing problems proved	1143	431	804
Total problems proved	11581	4615	8152

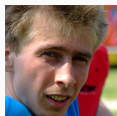
- rICoP with policy/value after 5 proving/learning iters on the training data
- $1624/1143 = 42.1\%$ improvement over leanCoP on the testing problems

Iteration	1	2	3	4	5	6	7	8
Training proved	12325	13749	14155	14363	14403	14431	14342	14498
Testing proved	1354	1519	1566	1595	1624	1586	1582	1591

More trees

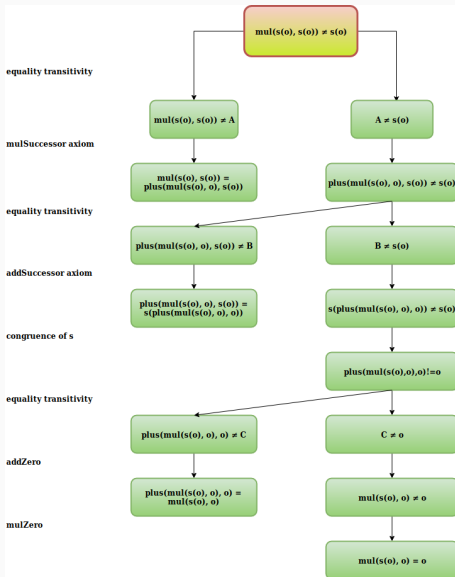


Recent CoP Mutants: FLoP, GNN, RNN, lazyCoP



- FLoP – Finding Longer Proofs (Zombori et al, 2019)
- Curriculum Learning used for connection tableau over Robinson Arithmetic
- addition and multiplication learned perfectly from $1 * 1 = 1$
- headed towards learning algorithms/decision procedures from math data
- currently black-box, combinations with symbolic methods (ILP) our next target
- Using RNNs for better tableau encoding, prediction of actions ...
- ... even guessing (decoding) next tableau literals (Piotrowski 2020)
- plCoP (Zombori 20), GNN-CoP (Olsak 20), lazyCoP (Rawson) ...
- Zombori: learning new explainable Prolog actions (tactics) from proofs

FLoP Training Proof



Guiding Saturation-Based Theorem Proving

Basic Saturation Loop – Given Clause Loop

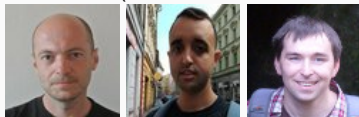
```
 $P := \emptyset$  (processed)  
 $U := \{\text{classified axioms and a negated conjecture}\}$  (unprocessed)  
while ( $U \neq \emptyset$ ) do  
  if ( $\perp \in U \cup P$ ) then return Unsatisfiable  
   $g := \text{select}(U)$  (choose a given clause)  
   $P := P \cup \{g\}$  (add to processed)  
   $U := U \setminus \{g\}$  (remove from unprocessed)  
   $U := U \cup \{\text{all clauses inferred from } g \text{ and } P\}$  (add inferences)  
done  
return Satisfiable
```

Typically, U grows quadratically wrt. P

1M clauses in U in 10s common – choosing good g gets harder

ENIGMA (2017): Guiding the Best ATPs like E Prover

- ENIGMA (Jan Jakubuv, Zar Goertzel, Karel Chvalovsky, others)



- The proof state are two large heaps of clauses *processed/unprocessed*
- learn on E's proof search traces, put classifier in E
- positive examples: clauses (lemmas) used in the proof
- negative examples: clauses (lemmas) not used in the proof
- 2021 **multi-phase architecture** (combination of different methods):
 - fast gradient-boosted decision trees (GBDTs) used in 2 ways
 - fast logic-aware graph neural network (GNN - Olsak) run on a GPU server
 - logic-based subsumption using fast indexing (discrimination trees - Schulz)
- The GNN scores many clauses (context/query) together in a large graph
- Sparse - vastly more efficient than transformers (~100k symbols)
- 2021: leapfrogging and Split&Merge:
- aiming at learning **reasoning/algo components**

Feedback prove/learn loop for ENIGMA on Mizar data

- Done on 57880 Mizar problems recently
- Serious ML-guidance breakthrough applied to the best ATPs
- Ultimately a **70% improvement** over the original strategy in 2019
- From 14933 proofs to 25397 proofs (all 10s CPU - no cheating)
- Went up to 40k in more iterations and 60s time in 2020
- 75% of the Mizar corpus reached in July 2021 - higher times and many runs: https://github.com/ai4reason/ATP_Proofs

	S	$S \odot M_9^0$	$S \oplus M_9^0$	$S \odot M_9^1$	$S \oplus M_9^1$	$S \odot M_9^2$	$S \oplus M_9^2$	$S \odot M_9^3$	$S \oplus M_9^3$
solved	14933	16574	20366	21564	22839	22413	23467	22910	23753
$S\%$	+0%	+10.5%	+35.8%	+43.8%	+52.3%	+49.4%	+56.5%	+52.8%	+58.4
$S+$	+0	+4364	+6215	+7774	+8414	+8407	+8964	+8822	+9274
$S-$	-0	-2723	-782	-1143	-508	-927	-430	-845	-454

	$S \odot M_{12}^3$	$S \oplus M_{12}^3$	$S \odot M_{16}^3$	$S \oplus M_{16}^3$
solved	24159	24701	25100	25397
$S\%$	+61.1%	+64.8%	+68.0%	+70.0%
$S+$	+9761	+10063	+10476	+10647
$S-$	-535	-295	-309	-183

ENIGMA Anonymous: Learning from patterns only

- The GNN and GBDTs only learn from formula structure, not symbols
- Not from symbols like $+$ and $*$ as Transformer & Co.
- E.g., learning on additive groups thus transfers to multiplicative groups
- Evaluation of old-Mizar ENIGMA on 242 new Mizar articles
- 13370 **new theorems**, $> 50\%$ of them with new terminology:
- The 3-phase ENIGMA is **58%** better on them than unguided E
- While **53.5%** on the old Mizar (where this ENIGMA was trained)
- Generalizing, analogizing and transfer abilities **unusual in the large transformer models**
- Recently also trained on 300k Isabelle/AFP problems (Sledgehammer)

3-phase Anonymous ENIGMA

The 3-phase ENIGMA (single strategy) solves in 30s 56.4% of Mizar (bushy)

Given Clause Loop in E + ML Guidance

Contribution 4

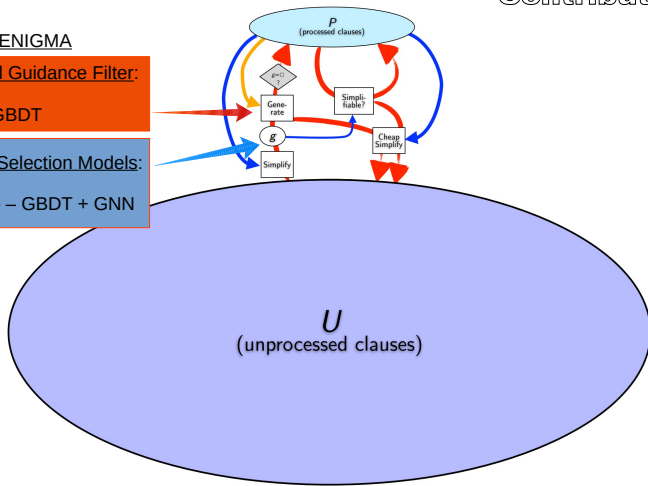
3-phase ENIGMA

Parental Guidance Filter:

Fast – GBDT

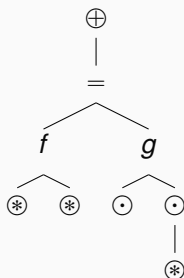
Clause Selection Models:

2-phase – GBDT + GNN



Clauses as Feature Vectors for ENIGMA

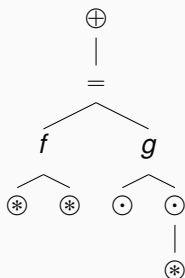
Collect and enumerate all the features. Count the clause features.



#	feature	count
1	$(\oplus, =, a)$	0
\vdots	\vdots	\vdots
11	$(\oplus, =, f)$	1
12	$(\oplus, =, g)$	1
13	$(=, f, *)$	2
14	$(=, g, \odot)$	2
15	$(g, \odot, *)$	1
\vdots	\vdots	\vdots

Clauses as Feature Vectors for ENIGMA

Take the counts as a **feature vector**.

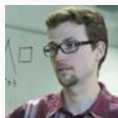


#	feature	count
1	($\oplus, =, a$)	0
\vdots	\vdots	\vdots
11	($\oplus, =, f$)	1
12	($\oplus, =, g$)	1
13	($=, f, *$)	2
14	($=, g, \odot$)	2
15	($g, \odot, *$)	1
\vdots	\vdots	\vdots

ENIGMA Proof Example – Knaster

```
theorem Th21:
  ex a st a is_a_fixpoint_of f
proof
  set H = {h where h is Element of L: h [= f.h];
  set fH = {f.h where h is Element of L: h [= f.h];
  set uH = "\/"(H, L);
  set fuH = "\/"(fH, L);
  take uH;
  now
    let fh be Element of L;
    assume fh in fH;
    then consider h being Element of L such that
A1: fh = f.h and
A2: h [= f.h;
    h in H by A2;
    then h [= uH by LATTICE3:38;
    hence fh [= f.uH by A1,QUANTAL1:def 12;
  end;
  then fH is_less_than f.uH by LATTICE3:def 17;
  then
A3: fuH [= f.uH by LATTICE3:def 21;
  now
    let a be Element of L;
    assume a in H;
    then consider h being Element of L such that
A4: a = h & h [= f.h;
    reconsider fh = f.h as Element of L;
    take fh;
    thus a [= fh & fh in fH by A4;
  end;
  then uH [= fuH by LATTICE3:47;
  then
A5: uH [= f.uH by A3,LATTICES:7;
  then f.uH [= f.(f.uH) by QUANTAL1:def 12;
  then f.uH in H;
  then f.uH [= uH by LATTICE3:38;
  hence uH = f.uH by A5,LATTICES:8;
end;
```

Neural Clause Selection in Vampire (M. Suda)



Deepire: Similar to ENIGMA:

- build a *classifier* for recognizing *good* clauses
- *good* are those that appeared in past proofs

Deepire's contributions:

- Learn from clause *derivation trees only*
Not looking at what it says, just who its ancestors were.
- Integrate using *layered clause queues*
A smooth improvement of the base clause selection strategy.
- Tree Neural Networks: constant work per derived clause
- A signature agnostic approach
- Delayed evaluation trick (not all derived need to be evaluated)

Preliminary Evaluation on Mizar “57880”

- Learn from 63595 proofs of 23071 problems (three 30s runs)
- Deepire solves 26217 (i.e. +4054) problems in a *single 10s run*

Prover9 - Research-Level Open Conjectures

- Michal Kinyon, Bob Veroff and Prover9: quasigroup and loop theory
- the **Abelian Inner Mappings** (AIM) Conjecture (>10 year program)
- Strong AIM: Q is AIM implies $Q/\text{Nuc}(Q)$ is abelian and $Q/Z(Q)$ is a group
- The Weak AIM Conjecture **positively resolved in August 2021**
- **Q is AIM implies Q is nilpotent of class at most 3.**
- 20-200k long proofs by Prover9 assisting the humans
- Prover9 hints strategy (Bob Veroff): extract hints from easier proofs to guide more difficult proofs
- Human-guided exploration to get good hints (not really automated yet)
- Millions of hints collected, various algorithms for their selection for a particular conjecture
- **Symbolic machine learning?**

ProofWatch: Symbolic/Statistical Guidance of E

- Bob Veroff's *hints method* used for Prover9
- solve many easier/related problems, produce millions of lemmas
- load the useful lemmas on the *watchlist* (kind of conjecturing)
- *boost inferences on clauses that subsume a watchlist* clause
- watchlist parts are *fast thinking*, bridged by *standard (slow) search*
- *symbolic guidance*, initial attempts to choose good hints by statistical ML
- Very *long proofs of open conjectures* in quasigroup/loop theory (AIM)
- ProofWatch (Goertzel et al. 2018): load many proofs separately
- *dynamically* boost those that have been covered more
- needed for *heterogeneous* ITP libraries
- *statistical*: watchlists chosen using similarity and usefulness
- *semantic/deductive*: dynamic guidance based on exact proof matching
- results in *better vectorial characterization* of saturation proof searches

ProofWatch: Statistical/Symbolic Guidance of E

```
theorem Th36: :: YELLOW_5:36
```

```
for L being non empty Boolean RelStr for a, b being Element of L  
holds ( 'not' (a "\/" b) = ('not' a) "\/" ('not' b)  
      & 'not' (a "\/" b) = ('not' a) "\/" ('not' b) )
```

- De Morgan's laws for Boolean lattices
- guided by 32 related proofs resulting in 2220 watchlist clauses
- 5218 given clause loops, resulting ATP proof is 436 clauses
- 194 given clauses match the watchlist and 120 (61.8%) used in the proof
- most helped by the proof of WAYBEL_1:85 – done for lower-bounded Heyting

```
theorem :: WAYBEL_1:85
```

```
for H being non empty lower-bounded RelStr st H is Heyting holds  
for a, b being Element of H holds  
'not' (a "\/" b) >= ('not' a) "\/" ('not' b)
```

ProofWatch: Vectorial Proof State

Final state of the proof progress for the 32 proofs guiding YELLOW_5 : 36

0	0.438	42/96	1	0.727	56/77	2	0.865	45/52	3	0.360	9/25
4	0.750	51/68	5	0.259	7/27	6	0.805	62/77	7	0.302	73/242
8	0.652	15/23	9	0.286	8/28	10	0.259	7/27	11	0.338	24/71
12	0.680	17/25	13	0.509	27/53	14	0.357	10/28	15	0.568	25/44
16	0.703	52/74	17	0.029	8/272	18	0.379	33/87	19	0.424	14/33
20	0.471	16/34	21	0.323	20/62	22	0.333	7/21	23	0.520	26/50
24	0.524	22/42	25	0.523	45/86	26	0.462	6/13	27	0.370	20/54
28	0.411	30/73	29	0.364	20/55	30	0.571	16/28	31	0.357	10/28

EnigmaWatch: ProofWatch used with ENIGMA

- Use the **proof completion ratios as features** for **characterizing proof state**
- Instead of just static conjecture features - **the proof vectors evolve**
- Feed them to ML systems along with other features
- Good improvement, extendable in various ways
- Alternative to backtracking-based RL-style systems? (**Doesn't forget!**)

EnigmaWatch: ProofWatch used with ENIGMA

Baseline	Mean	Var	Corr	Rand	Baseline \cup Mean	Total
1140	1357	1345	1337	1352	1416	1483

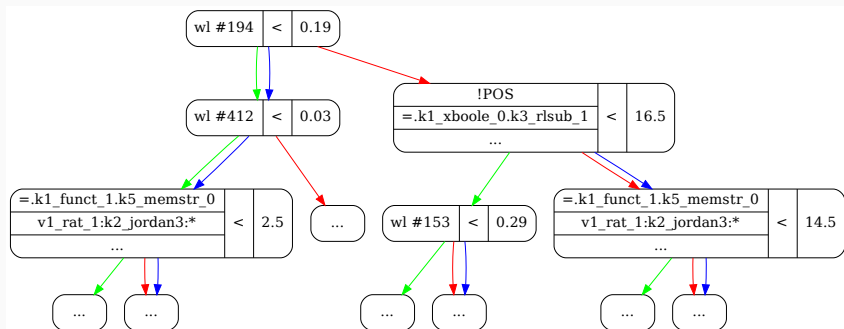
Table: ProofWatch evaluation: Problems solved by different versions.

loop	ENIGMA	Mean	Var	Corr	Rand	ENIGMA \cup Mean	Total
0	1557	1694	1674	1665	1690	1830	1974
1	1776	1815	1812	1812	1847	1983	2131
2	1871	1902	1912	1882	1915	2058	2200
3	1931	1954	1946	1920	1926	2110	2227

Table: ENIGMAWatch evaluation: Problems solved and the effect of looping.

- ENIGMAWatch initially much better, ENIGMA eventually catches up
- Still, EW produces simpler XGBoost models (easier learning)

Example of an XGBoost decision tree



RL of Neural Rewriting



- J. Piepenbrock (IJCAR'22): greatly improved RL for
- Gauthier's normalization in Robinson arithmetic
- Achieved good performance also on the polynomial normalization tasks
- Hold-out performance **better than** a top equational prover (Waldmeister) on AIM after 100 epochs of training on 2500 problems
- Combination with Prover9 **outperforms also unguided Prover9**
- Exciting: again, this is all in the verifiable/explainable proof format

METHOD	SUCCESS RATE ON AIM HOLD-OUT SET
WALDMEISTER (60s)	0.655
NEURALLY GUIDED REWRITING(60s)	0.702 \pm 0.015
PROVER9 (2s)	0.746
E (60s)	0.802
PROVER9 (60s)	0.833
NEURALLY GUIDED REWRITING (1s) + PROVER9 (59s)	0.902 \pm 0.016

Outline

Motivation, Learning vs. Reasoning

Computer Understandable (Formal) Math

Bird's-Eye View of ATP and ML

Learning of Theorem Proving - Overview

Demos

High-level Reasoning Guidance: Premise Selection

Low Level Guidance of Theorem Provers

Mid-level Reasoning Guidance

Synthesis and Autoformalization

TacticToe: mid-level ITP Guidance (Gauthier'17,18)



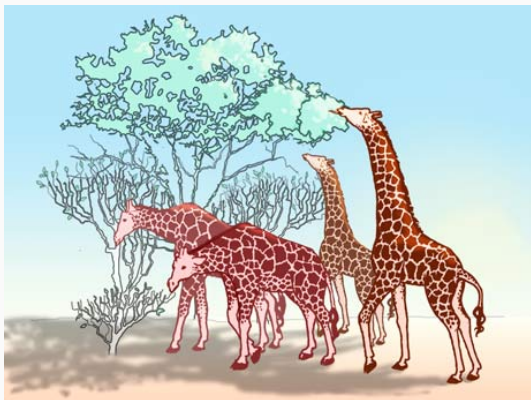
- TTT learns from human and its own tactical HOL4 proofs
- No translation or reconstruction needed - native tactical proofs
- Fully integrated with HOL4 and easy to use
- Similar to rICoP: policy/value learning for applying tactics in a state
- However much more technically challenging - a real breakthrough:
 - tactic and goal state recording
 - tactic argument abstraction
 - absolutization of tactic names
 - nontrivial evaluation issues
 - these issues have often more impact than adding better learners
- policy: which tactic/parameters to choose for a current goal?
- value: how likely is this proof state succeed?
- 66% of HOL4 toplevel proofs in 60s (**better than a hammer!**)
- similar recent work for Isabelle (Nagashima 2018), HOL Light (Google)

Tactician: Tactical Guidance for Coq (Blaauwbroek'20)



- Tactical guidance of Coq proofs
- Technically very challenging to do right - the Coq internals again nontrivial
- 39.3% on the Coq standard library, 56.7% in a union with CoqHammer (orthogonal)
- Fast approximate hashing for k-NN makes a lot of difference
- Fast re-learning more important than “cooler”/slower learners
- Fully integrated with Coq, should work for any development
- **User friendly, installation friendly, integration friendly and maintenance friendly**
- Took several years, but could become a very common tool for Coq formalizers

More Mid-level guidance: BliStr: Blind Strategymaker



- ATP **strategies are programs** specified in rich DSLs - can be **evolved**
- The ATP strategies are like giraffes, the problems are their food
- The better the giraffe specializes for eating problems unsolvable by others, the more it gets fed and further evolved

The E strategy with longest specification in Jan 2012

Longest human-designed strategy:

G-E--_029_K18_F1_PI_AE_SU_R4_CS_SP_S0Y:

```
4 * ConjectureGeneralSymbolWeight (
    SimulateSOS,100,100,100,50,50,10,50,1.5,1.5,1),
3 * ConjectureGeneralSymbolWeight (
    PreferNonGoals,200,100,200,50,50,1,100,1.5,1.5,1),
1 * Clauseweight (PreferProcessed,1,1,1),
1 * FIFOWeight (PreferProcessed)
```

BliStr: Blind Strategymaker

- **Strategies** characterized by the problems they solve
- **Problems** characterized by the strategies that solve them
- Improve on sets of **similar easy** problems to train for **unsolved** problems
- Interleave **low-time training on easy problems** with **high-time evaluation**
- Single strategy evolution done by **ParamILS** - Iterated Local Search (Hutter et al. 2009 – genetic methods work too)
- Thus **co-evolve** the strategies and their training problems
- The hard problems gradually become easier and turn into training data (the trees get lower for a taller giraffe)
- In the end, learn which strategy to use on which problem

The Longest E Strategy After BliStr Evolution

Evolutionarily designed Franken-strategy (29 heuristics combined):

```
6 * ConjectureGeneralSymbolWeight (PreferNonGoals,100,100,100,50,50,1000,100,1.5,1.5,1)
8 * ConjectureGeneralSymbolWeight (PreferNonGoals,200,100,200,50,50,1,100,1.5,1.5,1)
8 * ConjectureGeneralSymbolWeight (SimulateSOS,100,100,100,50,50,50,50,1.5,1.5,1)
4 * ConjectureRelativeSymbolWeight (ConstPrio,0.1, 100, 100, 100, 100, 1.5, 1.5, 1.5)
10 * ConjectureRelativeSymbolWeight (PreferNonGoals,0.5, 100, 100, 100, 100, 1.5, 1.5, 1.5)
2 * ConjectureRelativeSymbolWeight (SimulateSOS,0.5, 100, 100, 100, 100, 1.5, 1.5, 1)
10 * ConjectureSymbolWeight (ConstPrio,10,10,5,5,5,1.5,1.5,1.5)
1 * Clauseweight (ByCreationDate,2,1,0.8)
1 * Clauseweight (ConstPrio,3,1,1)
6 * Clauseweight (ConstPrio,1,1,1)
2 * Clauseweight (PreferProcessed,1,1,1)
6 * FIFOWeight (ByNegLitDist)
1 * FIFOWeight (ConstPrio)
2 * FIFOWeight (SimulateSOS)
8 * OrientLMaxWeight (ConstPrio,2,1,2,1,1)
2 * PNRefinedweight (PreferGoals,1,1,1,2,2,2,0.5)
10 * RelevanceLevelWeight (ConstPrio,2,2,0,2,100,100,100,100,1.5,1.5,1)
8 * RelevanceLevelWeight2 (PreferNonGoals,0,2,1,2,100,100,100,400,1.5,1.5,1)
2 * RelevanceLevelWeight2 (PreferGoals,1,2,1,2,100,100,100,400,1.5,1.5,1)
6 * RelevanceLevelWeight2 (SimulateSOS,0,2,1,2,100,100,100,400,1.5,1.5,1)
8 * RelevanceLevelWeight2 (SimulateSOS,1,2,0,2,100,100,100,400,1.5,1.5,1)
5 * rweight21_g
3 * Refinedweight (PreferNonGoals,1,1,2,1.5,1.5)
1 * Refinedweight (PreferNonGoals,2,1,2,2,2)
2 * Refinedweight (PreferNonGoals,2,1,2,3,0.8)
8 * Refinedweight (PreferGoals,1,2,2,1,0.8)
10 * Refinedweight (PreferGroundGoals,2,1,2,1.0,1)
20 * Refinedweight (SimulateSOS,1,1,2,1.5,2)
```

BliStr Evaluation on 1000 Mizar problems

- Original E coverage: 597 problems
- After 30 hours of strategy growing: 22 strategies covering 670 problems
- Best strategy solves 598 problems (1 more than all original strategies)
- Portfolio of 14 strategies improves E auto-mode by 25% on new problems
- Similar results for the Flyspeck problems
- Future: enrich the DSLs with new methods (ENIGMA/Watch, etc.)
- Be lazy, don't do "hard" theory-driven ATP research (a.k.a: thinking)
- Larry Wall (Programming Perl): *"We will encourage you to develop the three great virtues of a programmer: laziness, impatience, and hubris"*

Outline

Motivation, Learning vs. Reasoning

Computer Understandable (Formal) Math

Bird's-Eye View of ATP and ML

Learning of Theorem Proving - Overview

Demos

High-level Reasoning Guidance: Premise Selection

Low Level Guidance of Theorem Provers

Mid-level Reasoning Guidance

Synthesis and Autoformalization

Symbolic Rewriting with NNs



- Recurrent NNs with attention good at the **inf2formal task**
- Piotrowski 2018/19: Experiments with using RNNs for symbolic rewriting
- We can learn rewrite rules from sufficiently many data
- 80-90% success on AIM datasets, 70-99% on normalizing polynomials
- again, complements symbolic methods like ILP that suffer on big data
- in 2019 similar tasks taken up by Facebook - integration, etc.
- Similar use for **conjecturing** (Chvalovsky et al):
- Learn *consistent translations* between different math contexts:
- additive groups \rightarrow multiplicative groups

Symbolic Rewriting Datasets

Table: Examples in the AIM data set.

Rewrite rule:	Before rewriting:	After rewriting:
$b(s(e, v1), e) = v1$	$k(b(s(e, v1), e), v0)$	$k(v1, v0)$
$o(v0, e) = v0$	$t(v0, o(v1, o(v2, e)))$	$t(v0, o(v1, v2))$

Table: Examples in the polynomial data set.

Before rewriting:	After rewriting:
$(x * (x + 1)) + 1$	$x^2 + x + 1$
$(2 * y) + 1 + (y * y)$	$y^2 + 2 * y + 1$
$(x + 2) * ((2 * x) + 1) + (y + 1)$	$2 * x^2 + 5 * x + y + 3$

Alignment-based Conjecturing with RNNs

- Generate many "translations" of formulas by statistical concept alignments (Gauthier)
- Train RNNs (language models) on such translations
- Try to translate a conjecture by an RNN into different contexts
- If provable there, try to translate back

We can obtain a new valid automatically provable lemma

$$(X \cap Y) \setminus Z = (X \setminus Z) \cap (Y \setminus Z)$$

from

$$(X \cup Y) \setminus Z = (X \setminus Z) \cup (Y \setminus Z)$$

Examples of false but syntactically consistent conjectures:

for n, m being natural numbers holds $n \text{ gcd } m = n \text{ div } m$;

for R being Relation holds

`with_suprema(A) <=> with_suprema(inverse_relation(A))`;

More on Conjecturing in Mathematics

- **Targeted**: generate intermediate lemmas (cuts) for a harder conjecture
- **Unrestricted** (theory exploration):
 - Creation of interesting conjectures based on the previous theory
 - One of the most interesting activities mathematicians do (how?)
 - Higher-level AI/reasoning task - can we learn it?
 - If so, we have solved math:
 - ... just (recursively) **divide** Fermat into many subtasks ...
 - ... and **conquer** (I mean: **hammer**) them away

A bit of conjecturing history

- The topic goes back at least to Lenat (AM) and Fajtlowicz (Graffiti)
- Combined with automated theorem proving by Colton et al. in early 2000s (HR)
- Theory exploration for Isabelle by Johansson et al (Hipster)
- Several learning-based/neural approaches by our groups since 2015:
- Based mainly on learning analogies and informalization followed by probabilistic/neural disambiguation ...
- ... Gauthier, Kaliszyk, Chvalovsky, Piotrowski, Goertzel, Wang, Brown, JU

Conjecturing and Proof Synthesis by Neural Methods

- Karpathy'15 - RNN experiments with generating fake Math over Stacks
- I have tried to use that for formal math in 2016 but it looked weak
- GPT (-2,3) looks stronger
- Renewed experiments in 2020 (JU & J. Jakubuv: First Neural Conjecturing Datasets and Experiments. CICM'20) on:
 - All Mizar articles, stripped of comments and concatenated together (78M)
 - Articles with added context/disambiguation (156M) (types, names, thesis)
 - TPTP proofs of 28271 Mizar/MPTP theorems by E/ENIGMA (658M)
 - Just the conjecture and premises needed for the 28271 proofs printed in prefix notation
- Quite interesting results, server for Mizar authors
- Quickly taken up by others on HOL, Isabelle, MetaMath ...
- Caveat: Watch for "model pretraining" on undisclosed corpora - often GitHub/math repos that may contain (translations of) the testing data

Karpathy'15 - RNN generating fake Math over Stacks

<http://karpathy.github.io/2015/05/21/rnn-effectiveness/>

For $\bigoplus_{i=1, \dots, m}$ where $\mathcal{L}_{m_i} = 0$, hence we can find a closed subset \mathcal{H} in \mathcal{H} and any sets \mathcal{F} on X , U is a closed immersion of S , then $U \rightarrow T$ is a separated algebraic space.

Proof. Proof of (1). It also start we get

$$S = \text{Spec}(R) = U \times_X U \times_X U$$

and the comparably in the fibre product covering we have to prove the lemma generated by $\coprod Z \times_U U \rightarrow V$. Consider the maps M along the set of points Sch_{fppf} and $U \rightarrow U$ is the fibre category of S in U in Section, ?? and the fact that any U affine, see Morphisms, Lemma ?? . Hence we obtain a scheme S and any open subset $W \subset U$ in $\text{Sh}(G)$ such that $\text{Spec}(R') \rightarrow S$ is smooth or an

$$U = \bigcup U_i \times_{S_i} U_i$$

which has a nonzero morphism we may assume that f_i is of finite presentation over S . We claim that $\mathcal{O}_{X,x}$ is a scheme where $x, x', s' \in S'$ such that $\mathcal{O}_{X,x'} \rightarrow \mathcal{O}_{X',x'}$ is separated. By Algebra, Lemma ?? we can define a map of complexes $\text{GL}_{S'}(x'/S'')$ and we win. \square

To prove study we see that $\mathcal{F}|_U$ is a covering of \mathcal{X}' , and T_i is an object of $\mathcal{F}_{X_i/S}$ for $i > 0$ and \mathcal{F}_p exists and let \mathcal{F}_i be a presheaf of \mathcal{O}_X -modules on \mathcal{C} as a \mathcal{F} -module. In particular $\mathcal{F} = U/\mathcal{F}$ we have to show that

$$\tilde{M}^\bullet = \mathcal{I}^\bullet \otimes_{\text{Spec}(k)} \mathcal{O}_{S,s} - i_X^{-1} \mathcal{F}$$

is a unique morphism of algebraic stacks. Note that

$$\text{Arrows} = (\text{Sch}/S)_{fppf}^{\text{opp}}, (\text{Sch}/S)_{fppf}$$

and

$$V = \Gamma(S, \mathcal{O}) \rightarrow (U, \text{Spec}(A))$$

is an open subset of X . Thus U is affine. This is a continuous map of X is the inverse, the groupoid scheme S .

Proof. See discussion of sheaves of sets. \square

The result to prove any open covering follows from the less of Example ?? . It may replace S by $X_{\text{spaces, étale}}$ which gives an open subspace of X and T equal to S_{zar} , see Descent, Lemma ?? . Namely, by Lemma ?? we see that R is geometrically regular over S .

Lemma 0.1. Assume (3) and (3) by the construction in the description.

Suppose $X = \lim X_i$ (by the formal open covering X and a single map $\text{Proj}_X(A) = \text{Spec}(B)$ over U compatible with the complex

$$\text{Set}(A) = \Gamma(X, \mathcal{O}_{X, \mathcal{O}_X}).$$

When in this case of to show that $\mathcal{Q} \rightarrow C_{Z/X}$ is stable under the following result in the second conditions of (1), and (3). This finishes the proof. By Definition ?? (without element is when the closed subschemes are catenary. If T is surjective we may assume that T is connected with residue fields of S . Moreover there exists a closed subspace $Z \subset X$ of X where U in X' is proper (some defining as a closed subset of the uniqueness it suffices to check the fact that the following theorem

(1) f is locally of finite type. Since $S = \text{Spec}(R)$ and $Y = \text{Spec}(R)$.

Proof. This is form all sheaves of sheaves on X . But given a scheme U and a surjective étale morphism $U \rightarrow X$. Let $U \cap U = \prod_{i=1, \dots, n} U_i$ be the scheme X over S at the schemes $X_i \rightarrow X$ and $U = \lim_i X_i$. \square

The following lemma surjective retrocomposes of this implies that $\mathcal{F}_{x_0} = \mathcal{F}_{x_0} = \mathcal{F}_{X, \dots, 0}$.

Lemma 0.2. Let X be a locally Noetherian scheme over S , $E = \mathcal{F}_{X/S}$. Set $\mathcal{I} = \mathcal{I}_1 \subset \mathcal{I}'_n$. Since $\mathcal{I}^n \subset \mathcal{I}^n$ are nonzero over $i_0 \in \mathfrak{p}$ is a subset of $\mathcal{J}_{n,0} \circ \bar{A}_2$ works.

Lemma 0.3. In Situation ?? . Hence we may assume $\mathfrak{q}' = 0$.

Proof. We will use the property we see that \mathfrak{p} is the mex function (??). On the other hand, by Lemma ?? we see that

$$D(\mathcal{O}_X) = \mathcal{O}_X(D)$$

where K is an F -algebra where δ_{n+1} is a scheme over S . \square

Can you find the flaw(s) in this fake GPT-2 proof?

```
Applications Places emacs@dell Wed 15:02 Wed 15:02
File Edit Options Buffers Tools Index Mizar Hide/Show Help
Save Undo
:: generated theorem with "proof"
theorem Th23: :: STIRL2_1:23
for X, Y being finite set st not X is empty & X c= Y
& card X = card Y holds X = Y
proof
  let X, Y be finite set ;
  :: thesis: not X is empty & X c= Y & card X = card Y implies X = Y
  assume that
  A1: not X is empty and A2: X c= Y and A3: card X = card Y ;
  :: thesis: X = Y
  card (Y \ X) = (card Y) - (card X) by A1, A3, CARD_2:44;
  then A4: card (Y \ X) = ((card Y) - 1) - (card X) by CARD_1:30;
  X = Y \ X by A2, A3, Th22;
  hence X = Y by A4, XBOOLE_0:def_10;
  :: thesis: verum
end;
-:--- card_tst.miz 99% L2131 (Mizar Errors:13 hs Undo-Tree)
```

Figure: Fake full declarative GPT-2 "Mizar proof" - typechecks!

Mizar autocompletion server in action

Applications Places

GPT-2 generator trained on Mizar - Chromium

Not secure | grid01.cilrc.cvut.cz:5500

number of samples (fewer is raster)

Temperature (lower is less chaotic)

Length of output (shorter is faster)

Generate

Sample 1

```
theorem Th0 :: CARD_1:333
for M, N being Cardinal holds card M <= M V N
proof
let M, N be Cardinal; ::_thesis: card M <= M V
```

Sample 2

```
theorem Th0 :: CARD_1:333
for M, N being Cardinal holds M * * N is Cardinal
proof
let M, N be Cardinal; ::_thesis: M * * N is Cardinal
cf {
```

Sample 3

```
theorem Th0 :: CARD_1:333
for M, N being Cardinal holds Sum (M --> N) <= M * * N
proof
let M, N be Cardinal; ::_thesis: Sum (M
```

[github]

Figure: MGG - Mizar Gibberish Generator.

Proving the conditioned completions - MizAR hammer

```
Applications Places  
emacs@dell  
File Edit Options Buffers Tools Index Mizar Hide/Show Help  
Save Undo  
begin  
for M, N being Cardinal holds card M c= M ∨ N by XBOOLE_1:7,CARD_3:44,CARD_1:7,CARD_1:3; :: [ATP details]  
for X, Y being finite set st not X is empty & X c= Y & card X = card Y holds X = Y by CARD_FIN:1; :: [ATP details]  
for M, N being Cardinal holds  
( M in N iff card M c= N ) by Unsolved; :: [ATP details]  
for M, N being Cardinal holds  
( M in N iff card M in N ) by CARD_3:44,CARD_1:9; :: [ATP details]  
for M, N being Cardinal holds Sum (M --> N) = M * N by CARD_2:65; :: [ATP details]  
for M, N being Cardinal holds M ∧ (union N) in N by Unsolved; :: [ATP details]  
for M, N being Cardinal holds M * N = N * M by ATP-Unsolved; :: [ATP details]  
-:-- card tst.miz 3% L47 (Mizar Errors:2 hs Undo-Tree)  
Wrote /home/urban/mizwrk/7.13.01_4.181.1147/tst8/card_tst.miz
```

A correct conjecture that was too hard to prove

Kinyon and Stanovsky (algebraists) confirmed that this cut is valid:

```
theorem Th10: :: GROUPE_1:10
for G being finite Group
for N being normal Subgroup of G st
N is Subgroup of center G & G ./ N is cyclic
holds G is commutative
```

The generalization that avoids finiteness:

```
for G being Group
for N being normal Subgroup of G st
N is Subgroup of center G & G ./ N is cyclic
holds G is commutative
```

Gibberish Generator Provoking Algebraists

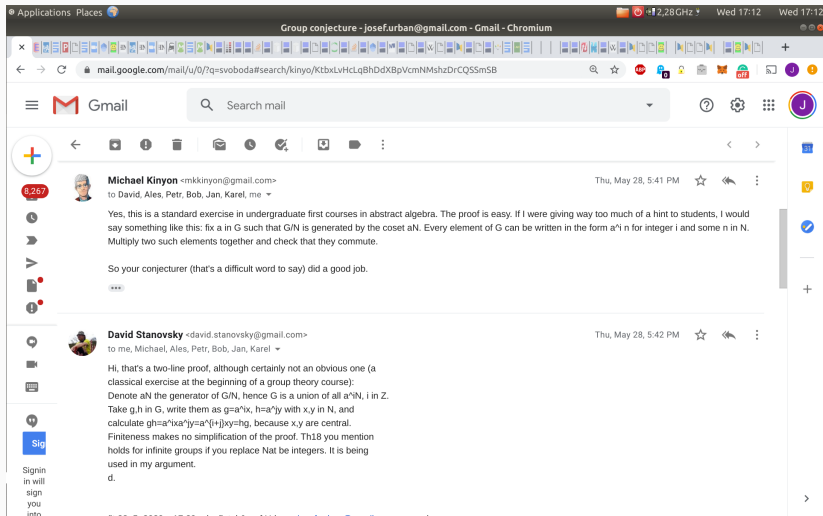


Figure: First successes in making mathematicians comment on AI.

More cuts

- In total 33100 in this experiment
- Ca 9k proved by trained ENIGMA
- Some are clearly false, yet quite natural to ask:

theorem :: SINCOS10:17

sec is increasing on $[0, \pi/2)$

leads to conjecturing the following:

Every differentiable function is increasing.

QSynt: Semantics-Aware Synthesis of Math Objects



- Gauthier'19-22
- Synthesize math expressions based on **semantic** characterizations
- i.e., not just on the **syntactic** descriptions (e.g. proof situations)
- Tree Neural Nets and RL (MCTS, policy/value), used for:
- Guiding synthesis of a diophantine equation characterizing a given set
- Guiding synthesis of combinators for a given lambda expression
- 2022: **invention of programs for OEIS sequences from scratch**
- 50k sequences discovered so far:
<https://www.youtube.com/watch?v=24oejR9wsXs>,
<http://grid01.ciirc.cvut.cz/~thibault/qsynt.html>
- Many conjectures invented: 4 different characterizations of primes
- Non-neural (Turing complete) computing and **semantics collaborates with the statistical learning**

QSynt: synthesizing the programs/expressions

- **Inductively defined** set P of our *programs and subprograms*,
- and an auxiliary set F of binary functions (higher-order arguments)
- are the smallest sets such that $0, 1, 2, x, y \in P$, and if $a, b, c \in P$ and $f, g \in F$ then:

$$a + b, a - b, a \times b, a \text{ div } b, a \text{ mod } b, \text{cond}(a, b, c) \in P$$

$$\lambda(x, y).a \in F, \text{loop}(f, a, b), \text{loop2}(f, g, a, b, c), \text{compr}(f, a) \in P$$

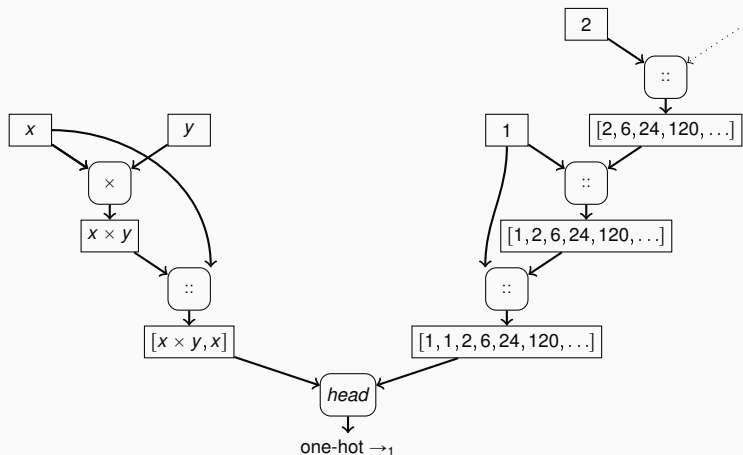
- Programs are built in reverse polish notation
- Start from an empty stack
- Use ML to **repeatedly choose the next operator to push on top of a stack**
- Example: Factorial is $\text{loop}(\lambda(x, y). x \times y, x, 1)$, built by:

$$[] \rightarrow_x [x] \rightarrow_y [x, y] \rightarrow_{\times} [x \times y] \rightarrow_x [x \times y, x]$$

$$\rightarrow_1 [x \times y, x, 1] \rightarrow_{\text{loop}} [\text{loop}(\lambda(x, y). x \times y, x, 1)]$$

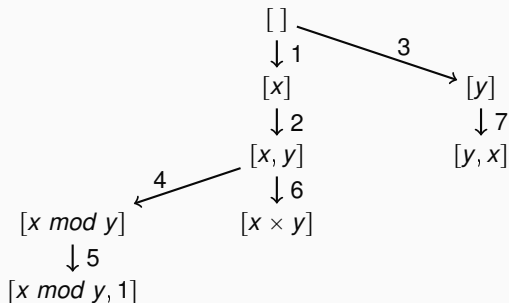
QSynt: Training of the Neural Net Guiding the Search

- The triple $((\text{head}([x \times y, x], [1, 1, 2, 6, 24, 120 \dots]), \rightarrow_1)$ is a training example extracted from the program for factorial $\text{loop}(\lambda(x, y). x \times y, x, 1)$
- \rightarrow_1 is the action (adding 1 to the stack) required on $[x \times y, x]$ to progress towards the construction of $\text{loop}(\lambda(x, y). x \times y, x, 1)$.

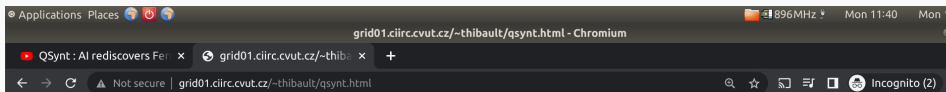


QSynt program search - Monte Carlo search tree

7 iterations of the search loop gradually extending the search tree. The set of the synthesized programs after the 7th iteration is $\{1, x, y, x \times y, x \bmod y\}$.



QSynt web interface for program invention



QSynt: Program Synthesis for Integer Sequences

Propose a sequence of integers:

Timeout (maximum 300s)

Generated integers (maximum 100)

A few sequences you can try:

0 1 1 0 1 0 1 0 0 0 1 0 1 0 0 0 1 0 1

0 1 4 9 16 21 25 28 36 37 49

0 1 3 6 10 15

2 3 5 7 11 13 17 19 23 29 31 37 41 43

1 1 2 6 24 120

2 4 16 256

QSynt inventing Fermat pseudoprimes

Positive integers k such that $2^k \equiv 2 \pmod k$. (341 = 11 * 31 is the first non-prime)

First 16 generated numbers (f(0),f(1),f(2),...):

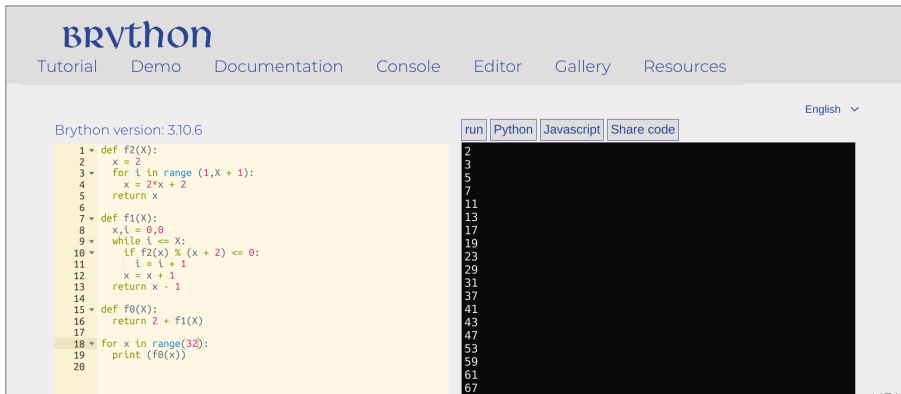
2 3 5 7 11 13 17 19 23 29 31 37 41 43 47 53

Generated sequence matches best with: [A15919](#)(1-75), [A100726](#)(0-59), [A40](#)(0-58)

Program found in 5.81 seconds

$f(x) := 2 + \text{compr}(\backslash x.\text{loop}(\backslash(x,i).2*x + 2, x, 2) \text{ mod } (x + 2), x)$

Run the equivalent Python program [here](#) or in the window below:



The screenshot shows the Brython web interface. At the top, the Brython logo is displayed, followed by navigation links: Tutorial, Demo, Documentation, Console, Editor, Gallery, and Resources. A language selector is set to English. Below the navigation, the Brython version is shown as 3.10.6. There are four buttons: run, Python, Javascript, and Share code. The Python button is active. The main area is split into two panes. The left pane shows Python code for finding Fermat pseudoprimes. The right pane shows the output of the program, which is a list of the first 16 generated numbers: 2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53.

```
1 def f2(X):
2     x = 2
3     for i in range (1,X + 1):
4         x = 2*x + 2
5     return x
6
7 def f1(X):
8     x,i = 0,0
9     while i <= X:
10        if f2(x) % (x + 2) <= 0:
11            i = i + 1
12            x = x + 1
13        return x - 1
14
15 def f0(X):
16     return 2 + f1(X)
17
18 for x in range(32):
19     print (f0(x))
20
```

2
3
5
7
11
13
17
19
23
29
31
37
41
43
47
53

Lucas/Fibonacci characterization of (pseudo)primes

input sequence: 2,3,5,7,11,13,17,19,23,29

invented output program:

```
f(x) := compr(\(x,y).(loop2(\(x,y).x + y, \(x,y).x, x, 1, 2) - 1)
          mod (1 + x), x + 1) + 1
```

human conjecture: x is prime iff? x divides $(\text{Lucas}(x) - 1)$

PARI program:

```
? lucas(n) = fibonacci(n+1)+fibonacci(n-1)
? b(n) = (lucas(n) - 1) % n
```

Counterexamples (Bruckman-Lucas pseudoprimes):

```
? for(n=1,4000,if(b(n)==0,if(isprime(n),0,print(n))))
```

1

705

2465

2737

3745

QSynt inventing primes using Wilson's theorem

n is prime iff $(n - 1)! + 1$ is divisible by n (i.e.: $(n - 1)! \equiv -1 \pmod{n}$)

First 32 generated numbers ($f(0), f(1), f(2), \dots$):

0 1 1 0 1 0 1 0 0 0 1 0 1 0 0 0 1 0 1 0 0 0 1 0 0 0 0 1 0 0 0 0 0 1 0 1 0

Generated sequence matches best with: [A10051](#)(0-100), [A252233](#)(0-29), [A283991](#)(0-24)

Program found in 5.17 seconds

$f(x) := (\text{loop}(\backslash(x,i).x * i, x, x) \bmod (x + 1)) \bmod 2$

Run the equivalent Python program [here](#) or in the window below:

The screenshot shows the Brython web interface. At the top, the word "Brython" is displayed in a large blue font. Below it, there are navigation links: "Tutorial", "Demo", "Documentation", "Console", "Editor", "Gallery", and "Resources". On the right side, there is a language selector set to "English" with a dropdown arrow. Below the navigation, the text "Brython version: 3.10.6" is shown. The main area is divided into two parts: a code editor on the left and a console on the right. The code editor contains the following Python code:

```
1 def f1(X):
2     x = X
3     for i in range(1, X + 1):
4         x = x * i
5     return x
6
7 def f0(X):
8     return (f1(X) % (X + 1)) % 2
9
10 for x in range(32):
11     print (f0(x))
12
```

The console on the right shows the output of the program, which is a sequence of 32 binary digits: 0 1 1 0 1 0 1 0 0 0 1 0 1 0 0 0 1 0 1 0 0 0 1 0 0 0 0 1 0 0 0 0 0 1 0 1 0.

Are two QSynt programs equivalent?

- As with primes, we often find **many programs** for one OEIS sequence
- It may be quite hard to see that the programs **are equivalent**
- A simple example for 0, 2, 4, 6, 8, ... with two programs f and g :
 - $f(0) = 0, f(n) = 2 + f(n - 1)$ if $n > 0$
 - $g(n) = 2 * n$
 - conjecture: $\forall n \in \mathbb{N}. g(n) = f(n)$
- We can ask mathematicians, but we have **thousands of such problems**
- Or we can try to **ask our ATPs** (and thus create a large ATP benchmark)!
- Here is one SMT encoding by Mikolas Janota:

```
(set-logic UFLIA)
(define-fun-rec f ((x Int)) Int (ite (<= x 0) 0 (+ 2 (f (- x 1))))
(assert (exists ((c Int)) (and (> c 0) (not (= (f c) (* 2 c))))))
(check-sat)
```

Inductive proof by Vampire of the $f = g$ equivalence

```
% SZS output start Proof for rec2
1. f(X0) = $ite($lesseq(X0,0), 0,$sum(2,f($difference(X0,1)))) [input]
2. ? [X0 : $int] : ($greater(X0,0) & ~f(X0) = $product(2,X0)) [input]
[...]
43. ~$less(0,X0) | iG0(X0) = $sum(2,iG0($sum(X0,-1))) [evaluation 40]
44. (! [X0 : $int] : (($product(2,X0) = iG0(X0) & ~$less(X0,0)) => $product(2,$sum(X0,1)) = iG0($sum(X0,1)))
    & $product(2,0) = iG0(0)) => ! [X1 : $int] : ($less(0,X1) => $product(2,X1) = iG0(X1)) [induction hypo]
[...]
49. $product(2,0) != iG0(0) | $product(2,$sum(sK3,1)) != iG0($sum(sK3,1)) | ~$less(0,sK1) [resolution 48,41]
50. $product(2,0) != iG0(0) | $product(2,sK3) = iG0(sK3) | ~$less(0,sK1) [resolution 47,41]
51. $product(2,0) != iG0(0) | ~$less(sK3,0) | ~$less(0,sK1) [resolution 46,41]
52. 0 != iG0(0) | $product(2,$sum(sK3,1)) != iG0($sum(sK3,1)) | ~$less(0,sK1) [evaluation 49]
53. 0 != iG0(0) | $product(2,sK3) = iG0(sK3) | ~$less(0,sK1) [evaluation 50]
54. 0 != iG0(0) | ~$less(sK3,0) | ~$less(0,sK1) [evaluation 51]
55. 0 != iG0(0) | ~$less(sK3,0) [subsumption resolution 54,39]
57. 1 <=> $less(sK3,0) [avatar definition]
59. ~$less(sK3,0) <- (~1) [avatar component clause 57]
61. 2 <=> 0 = iG0(0) [avatar definition]
64. ~1 | ~2 [avatar split clause 55,61,57]
65. 0 != iG0(0) | $product(2,sK3) = iG0(sK3) [subsumption resolution 53,39]
67. 3 <=> $product(2,sK3) = iG0(sK3) [avatar definition]
69. $product(2,sK3) = iG0(sK3) <- (3) [avatar component clause 67]
70. 3 | ~2 [avatar split clause 65,61,67]
71. 0 != iG0(0) | $product(2,$sum(sK3,1)) != iG0($sum(sK3,1)) [subsumption resolution 52,39]
72. $product(2,$sum(1,sK3)) != iG0($sum(1,sK3)) | 0 != iG0(0) [forward demodulation 71,5]
74. 4 <=> $product(2,$sum(1,sK3)) = iG0($sum(1,sK3)) [avatar definition]
76. $product(2,$sum(1,sK3)) != iG0($sum(1,sK3)) <- (~4) [avatar component clause 74]
77. ~2 | ~4 [avatar split clause 72,74,61]
82. 0 = iG0(0) [resolution 36,10]
85. 2 [avatar split clause 82,61]
246. iG0($sum(X1,1)) = $sum(2,iG0($sum($sum(X1,1),-1))) | $less(X1,0) [resolution 43,14]
251. $less(X1,0) | iG0($sum(X1,1)) = $sum(2,iG0(X1)) [evaluation 246]
[...]
1176. $false <- (~1, 3, ~4) [subsumption resolution 1175,1052]
1177. 1 | ~3 | 4 [avatar contradiction clause 1176]
1178. $false [avatar sat refutation 64,70,77,85,1177]
% SZS output end Proof for rec2
% Time elapsed: 0.016 s
```


Autoformalization

- Goal: Learn **understanding of informal math** formulas and reasoning
- Experiments with the **CYK chart parser linked to semantic methods**
- Experiments with **neural methods**
- Semantic methods: Type checking, theorem proving
- Corpora: Flyspeck, Mizar, Proofwiki, etc.

Statistical/Semantic Parsing of Informalized HOL

- Training and testing examples exported from Flyspeck formulas
 - Along with their **informalized** versions
- Grammar parse trees
 - Annotate each (nonterminal) symbol with its **HOL type**
 - Also “semantic (formal)” nonterminals annotate overloaded terminals
 - guiding analogy: word-sense disambiguation using CYK is common
- Terminals exactly compose the textual form, for example:

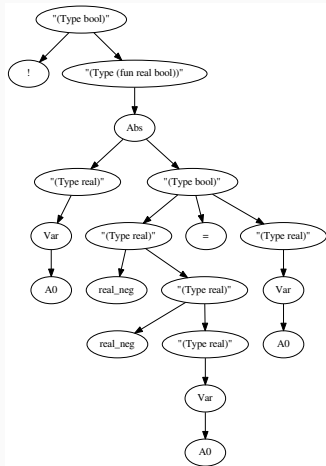
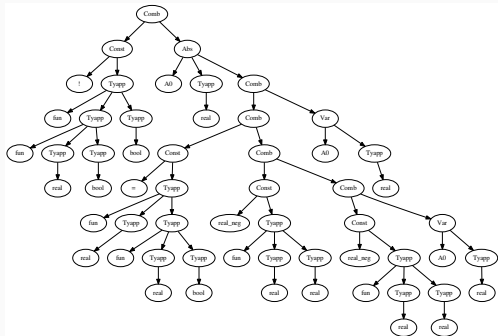
- **REAL_NEGNEG**: $\forall x. --x = x$

```
(Comb (Const "!" (Tyapp "fun" (Tyapp "fun" (Tyapp "real") (Tyapp "bool")))
(Tyapp "bool"))) (Abs "A0" (Tyapp "real") (Comb (Comb (Const "=" (Tyapp "fun"
(Tyapp "real") (Tyapp "fun" (Tyapp "real") (Tyapp "bool")))) (Comb (Const
"real_neg" (Tyapp "fun" (Tyapp "real") (Tyapp "real"))) (Comb (Const
"real_neg" (Tyapp "fun" (Tyapp "real") (Tyapp "real"))) (Var "A0" (Tyapp
"real")))) (Var "A0" (Tyapp "real"))))
```

- **becomes**

```
("(Type bool)" ! ("(Type (fun real bool))" (Abs ("(Type real)"
(Var A0)) ("(Type bool)" ("(Type real)" real_neg ("(Type real)"
real_neg ("(Type real)" (Var A0)))) = ("(Type real)" (Var A0))))))
```

Example grammars



CYK Learning and Parsing (KUV, ITP 17)

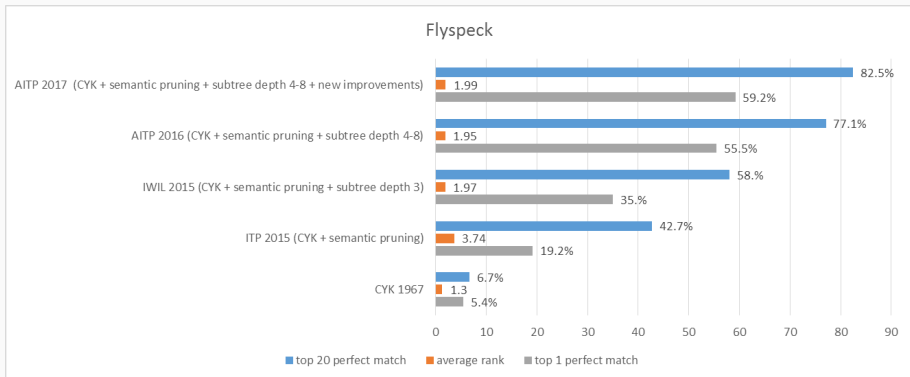
- Induce PCFG (probabilistic context-free grammar) from the trees
 - Grammar rules obtained from the inner nodes of each grammar tree
 - Probabilities are computed from the frequencies
- The PCFG grammar is binarized for efficiency
 - New nonterminals as shortcuts for multiple nonterminals
- CYK: dynamic-programming algorithm for parsing ambiguous sentences
 - input: sentence – a sequence of words and a binarized PCFG
 - output: N most probable parse trees
- Additional semantic pruning
 - Compatible types for free variables in subtrees
 - Optionally merge equivalent subtrees (tricky)
- Allow small probability for each symbol to be a variable
- Top parse trees are de-binarized to the original CFG
 - Transformed to HOL parse trees (preterms, Hindley-Milner)
 - typed checked in HOL and then given to an ATP (hammer)

Autoformalization based on PCFG and semantics

- “`sin (0 * x) = cos pi / 2`”
- produces 16 parses
- of which 11 get type-checked by HOL Light as follows
- with all but three being proved by HOL(y)Hammer
- **demo:** <http://grid01.ciirc.cvut.cz/~mptp/demo.ogv>

```
sin (&0 * A0) = cos (pi / &2) where A0:real
sin (&0 * A0) = cos pi / &2 where A0:real
sin (&0 * &A0) = cos (pi / &2) where A0:num
sin (&0 * &A0) = cos pi / &2 where A0:num
sin (&(0 * A0)) = cos (pi / &2) where A0:num
sin (&(0 * A0)) = cos pi / &2 where A0:num
csin (Cx (&0 * A0)) = ccos (Cx (pi / &2)) where A0:real
csin (Cx (&0) * A0) = ccos (Cx (pi / &2)) where A0:real^2
Cx (sin (&0 * A0)) = ccos (Cx (pi / &2)) where A0:real
csin (Cx (&0 * A0)) = Cx (cos (pi / &2)) where A0:real
csin (Cx (&0) * A0) = Cx (cos (pi / &2)) where A0:real^2
```

Flyspeck Progress



Neural Autoformalization (Wang et al., 2018)

- generate about 1M Latex - Mizar pairs synthetically (quite advanced)
- train neural seq-to-seq translation models (Luong – NMT)
- evaluate on about 100k examples
- many architectures tested, some work much better than others
- very important latest invention: attention in the seq-to-seq models
- more data crucial for neural training
- Recent addition: unsupervised MT methods (Lample et al 2018) – no need for aligned data, improving a lot!
- Type-checking not yet internal (boosting well-typed data externally)

Neural Autoformalization data

Rendered \LaTeX

Mizar

If $X \subseteq Y \subseteq Z$, then $X \subseteq Z$.

`X c= Y & Y c= Z implies X c= Z;`

Tokenized Mizar

`X c= Y & Y c= Z implies X c= Z ;`

\LaTeX

If $\$X \subseteq Y \subseteq Z\$,$ then $\$X \subseteq Z\$.$

Tokenized \LaTeX

`If $ X \subseteq Y \subseteq Z $, then $ X \subseteq Z $.`

Neural Autoformalization results

Parameter	Final Test Perplexity	Final Test BLEU	Identical Statements (%)	Identical No-overlap (%)
128 Units	3.06	41.1	40121 (38.12%)	6458 (13.43%)
256 Units	1.59	64.2	63433 (60.27%)	19685 (40.92%)
512 Units	1.6	67.9	66361 (63.05%)	21506 (44.71%)
1024 Units	1.51	61.6	69179 (65.73%)	22978 (47.77%)
2048 Units	2.02	60	59637 (56.66%)	16284 (33.85%)

Neural Fun – Performance after Some Training

Rendered
L^AT_EX

Input L^AT_EX

Correct

Snapshot-
1000

Snapshot-
2000

Snapshot-
3000

Snapshot-
4000

Snapshot-
5000

Snapshot-
6000

Snapshot-
7000

Suppose s_8 is convergent and s_7 is convergent . Then $\lim(s_8+s_7) = \lim s_8 + \lim s_7$

Suppose $\{ s_{8} \}$ is convergent and $\{ s_{7} \}$ is convergent . Then $\lim (\{ s_{8} \} + \{ s_{7} \}) = \lim \{ s_{8} \} + \lim \{ s_{7} \}$.

seq1 is convergent & seq2 is convergent implies $\lim (seq1 + seq2) = (\lim seq1) + (\lim seq2)$;

$x \text{ in dom } f \text{ implies } (x * y) * (f | (x | (y | (y | y)))) = (x | (y | (y | (y | y))))$;

seq is summable implies seq is summable ;

seq is convergent & $\lim seq = 0c$ implies $seq = seq$;

seq is convergent & $\lim seq = \lim seq$ implies $seq1 + seq2$ is convergent ;

seq1 is convergent & $\lim seq2 = \lim seq2$ implies $\lim_{inf} seq1 = \lim_{inf} seq2$;

seq is convergent & $\lim seq = \lim seq$ implies $seq1 + seq2$ is convergent ;

seq is convergent & seq9 is convergent implies $\lim (seq + seq9) = (\lim seq) + (\lim seq9)$;

Unsupervised NMT Fun on Short Formulas

```
len <* a *> = 1 ;
assume i < len q ;
len <* q *> = 1 ;
s = apply ( v2 , v1 ast t ) ;
s . ( i + 1 ) = tt . ( i + 1 )
1 + j <= len v2 ;
1 + j + 0 <= len v2 + 1 ;
let i be Nat ;
assume v is_applicable_to t ;
let t be type of T ;
a ast t in downarrow t ;
t9 in types a ;
a ast t <= t ;
A is_applicable_to t ;
Carrier ( f ) c= B
u in B or u in { v } ;
F . w in w & F . w in I ;
GG . y in rng HH ;
a * L = Z_ZeroLC ( V ) ;
not u in { v } ;
u <> v ;
v - w = v1 - w1 ;
v + w = v1 + w1 ;
x in A & y in A ;

len <* a *> = 1 ;
i < len q ;
len <* q *> = 1 ;
s = apply ( v2 , v1 ) . t ;
s . ( i + 1 ) = tau1 . ( i + 1 )
1 + j <= len v2 ;
1 + j + 0 <= len v2 + 1 ;
i is_at_least_length_of p ;
not v is applicable ;
t is_orientedpath_of v1 , v2 , T ;
a *' in downarrow t ;
t '2 in types a ;
a *' <= t ;
A is applicable ;
support ppf n c= B
u in B or u in { v } ;
F . w in F & F . w in I ;
G0 . y in rng ( H1 ./ . y ) ;
a * L = ZeroLC ( V ) ;
u >> v ;
u <> v ;
vw = v1 - w1 ;
v + w = v1 + w1 ;
assume [ x , y ] in A ;
```

More Personal Notes (written for AI/TP students)

- *Think globally, act locally.* Big dreams about AI, etc. But act by trying many small steps/experiments. AI/TP is a very **experimental science**.
- Sometimes need to commit a lot. The first Mizar-to-ATP translation took years, but bore a lot of fruit. Today millions USD in Google HOL/RL.
- Follow your dream mercilessly - avoid distractions (stay focused - hard for many smart people). Find/do what you are convinced/passionate about.
- **Avoid** excessive "theorem envy". AI is often *not* Math. We want to *replace* mathematicians, not be them. Always reflect and implement your thinking.
- Many AI improvements come from bringing ideas/systems together:
"Automate, automate, automate!"
- Become a hacker. Learn rapid prototyping. Learn to gain maximum info from initial experiments, then iterate. **"Experience, not only doctrine"**.
- Learn at least one high-level symbolic language - lisp, prolog, ml, haskell. At least one scripting language: perl, python, ruby, shell.
- Stay motivated by reading giants of science: Einstein, Poincare, Russel, Heisenberg, Turing, Deutsch, Dawkins
- Read good sci-fi: Clarke, Heinlein, Stephenson, Stroth, ...

Acknowledgments

- Prague Automated Reasoning Group <http://arg.ciirc.cvut.cz/>:
 - Jan Jakubuv, Chad Brown, Martin Suda, Karel Chvalovsky, Bob Veroff, Zar Goertzel, Bartosz Piotrowski, Lasse Blaauwbroek, Martin Smolik, Jiri Vyskocil, Petr Pudlak, David Stanovsky, Krystof Hoder, ...
- HOL(y)Hammer group in Innsbruck:
 - Cezary Kaliszyk, Thibault Gauthier, Michael Faerber, Yutaka Nagashima, Shawn Wang
- ATP and ITP people:
 - Stephan Schulz, Geoff Sutcliffe, Andrej Voronkov, Kostya Korovin, Larry Paulson, Jasmin Blanchette, John Harrison, Tom Hales, Tobias Nipkow, Andrzej Trybulec, Piotr Rudnicki, Adam Pease, ...
- Learning2Reason people at Radboud University Nijmegen:
 - Herman Geuvers, Tom Heskes, Daniel Kuehlwein, Evgeni Tsivtsivadze,
- Google Research: Christian Szegedy, Geoffrey Irving, Alex Alemi, Francois Chollet, Sarah Loos
- ... and many more ...
- Funding: Marie-Curie, NWO, ERC, OPVVV

Some General and Hammer/Tactical References

- J. C. Blanchette, C. Kaliszyk, L. C. Paulson, J. Urban: Hammering towards QED. *J. Formalized Reasoning* 9(1): 101-148 (2016)
- Cezary Kaliszyk, Josef Urban: Learning-Assisted Automated Reasoning with Flyspeck. *J. Autom. Reason.* 53(2): 173-213 (2014)
- Cezary Kaliszyk, Josef Urban: MizAR 40 for Mizar 40. *J. Autom. Reason.* 55(3): 245-256 (2015)
- Cezary Kaliszyk, Josef Urban: Learning-assisted theorem proving with millions of lemmas. *J. Symb. Comput.* 69: 109-128 (2015)
- Jasmin Christian Blanchette, David Greenaway, Cezary Kaliszyk, Daniel Kühlwein, Josef Urban: A Learning-Based Fact Selector for Isabelle/HOL. *J. Autom. Reason.* 57(3): 219-244 (2016)
- Bartosz Piotrowski, Josef Urban: ATPboost: Learning Premise Selection in Binary Setting with ATP Feedback. *IJCAR 2018*: 566-574
- T. Gauthier, C. Kaliszyk, J. Urban, R. Kumar, M. Norrish: Learning to Prove with Tactics. *CoRR* abs/1804.00596 (2018).
- Lasse Blaauwbroek, Josef Urban, Herman Geuvers: Tactic Learning and Proving for the Coq Proof Assistant. *LPAR 2020*: 138-150
- Lasse Blaauwbroek, Josef Urban, Herman Geuvers: The Tactician (extended version): A Seamless, Interactive Tactic Learner and Prover for Coq. *CoRR* abs/2008.00120 (2020)
- L. Czajka, C. Kaliszyk: Hammer for Coq: Automation for Dependent Type Theory. *J. Autom. Reasoning* 61(1-4): 423-453 (2018)
- G. Irving, C. Szegedy, A. Alemi, N. Eén, F. Chollet, J. Urban: DeepMath - Deep Sequence Models for Premise Selection. *NIPS 2016*: 2235-2243
- C. Kaliszyk, J. Urban, J. Vyskocil: Efficient Semantic Features for Automated Reasoning over Large Theories. *IJCAI 2015*: 3084-3090
- J. Urban, G. Sutcliffe, P. Pudlák, J. Vyskocil: MaLAREa SG1- Machine Learner for Automated Reasoning with Semantic Guidance. *IJCAR 2008*: 441-456
- J. Urban, J. Vyskocil: Theorem Proving in Large Formal Mathematics as an Emerging AI Field. *LNCS* 7788, 240-257, 2013.

Some References on E/ENIGMA, CoPs and Related

- Stephan Schulz: System Description: E 1.8. LPAR 2013: 735-743
- S. Schulz, Simon Cruanes, Petar Vukmirovic: Faster, Higher, Stronger: E 2.3. CADE 2019: 495-507
- J. Jakubuv, J. Urban: Extending E Prover with Similarity Based Clause Selection Strategies. CICM 2016: 151-156
- J. Jakubuv, J. Urban: ENIGMA: Efficient Learning-Based Inference Guiding Machine. CICM 2017: 292-302
- Cezary Kaliszyk, Josef Urban, Henryk Michalewski, Miroslav Olsák: Reinforcement Learning of Theorem Proving. NeurIPS 2018: 8836-8847
- Zarathustra Goertzel, Jan Jakubuv, Stephan Schulz, Josef Urban: ProofWatch: Watchlist Guidance for Large Theories in E. ITP 2018: 270-288
- S. M. Loos, G. Irving, C. Szegedy, C. Kaliszyk: Deep Network Guided Proof Search. LPAR 2017: 85-105
- Karel Chvalovský, Jan Jakubuv, Martin Suda, Josef Urban: ENIGMA-NG: Efficient Neural and Gradient-Boosted Inference Guidance for E. CADE 2019: 197-215
- Jan Jakubuv, Josef Urban: Hammering Mizar by Learning Clause Guidance. ITP 2019: 34:1-34:8
- Zarathustra Goertzel, Jan Jakubuv, Josef Urban: ENIGMAWatch: ProofWatch Meets ENIGMA. TABLEAUX 2019: 374-388
- Zarathustra Amadeus Goertzel: Make E Smart Again (Short Paper). IJCAR (2) 2020: 408-415
- Jan Jakubuv, Karel Chvalovský, Miroslav Olsák, Bartosz Piotrowski, Martin Suda, Josef Urban: ENIGMA Anonymous: Symbol-Independent Inference Guiding Machine. IJCAR (2) 2020: 448-463
- Zsolt Zombori, Adrián Csiszárík, Henryk Michalewski, Cezary Kaliszyk, Josef Urban: Towards Finding Longer Proofs. CoRR abs/1905.13100 (2019)
- Zsolt Zombori, Josef Urban, Chad E. Brown: Prolog Technology Reinforcement Learning Prover - (System Description). IJCAR (2) 2020: 489-507
- Miroslav Olsák, Cezary Kaliszyk, Josef Urban: Property Invariant Embedding for Automated Reasoning. ECAI 2020: 1395-1402

Some Conjecturing References

- Douglas Bruce Lenat. *AM: An Artificial Intelligence Approach to Discovery in Mathematics as Heuristic Search*. PhD thesis, Stanford, 1976.
- Siemion Fajtlowicz. On conjectures of Graffiti. *Annals of Discrete Mathematics*, 72(1–3):113–118, 1988.
- Simon Colton. *Automated Theory Formation in Pure Mathematics*. Distinguished Dissertations. Springer London, 2012.
- Moa Johansson, Dan Rosén, Nicholas Smallbone, and Koen Claessen. Hipster: Integrating theory exploration in a proof assistant. In *CICM 2014*, pages 108–122, 2014.
- Thibault Gauthier, Cezary Kaliszyk, and Josef Urban. Initial experiments with statistical conjecturing over large formal corpora. In *CICM'16 WiP Proceedings*, pages 219–228, 2016.
- Thibault Gauthier, Cezary Kaliszyk: Sharing HOL4 and HOL Light Proof Knowledge. *LPAR 2015*: 372-386
- Thibault Gauthier. Deep reinforcement learning in HOL4. *CoRR*, abs/1910.11797, 2019.
- Chad E. Brown and Thibault Gauthier. Self-learned formula synthesis in set theory. *CoRR*, abs/1912.01525, 2019.
- Bartosz Piotrowski, Josef Urban, Chad E. Brown, Cezary Kaliszyk: Can Neural Networks Learn Symbolic Rewriting? AITP 2019, CoRR abs/1911.04873 (2019)
- Zarathustra Goertzel and Josef Urban. Usefulness of Lemmas via Graph Neural Networks (Extended Abstract). AITP 2019.
- Karel Chvalovský, Thibault Gauthier and Josef Urban: First Experiments with Data Driven Conjecturing (Extended Abstract). AITP 2019.
- Thibault Gauthier: Deep Reinforcement Learning for Synthesizing Functions in Higher-Order Logic. *LPAR 2020*: 230-248
- Bartosz Piotrowski, Josef Urban: Guiding Inferences in Connection Tableau by Recurrent Neural Networks. *CICM 2020*: 309-314
- Josef Urban, Jan Jakubuv: First Neural Conjecturing Datasets and Experiments. *CICM 2020*: 315-323

References on PCFG and Neural Autoformalization

- Cezary Kaliszyk, Josef Urban, Jirí Vyskocil: Learning to Parse on Aligned Corpora (Rough Diamond). ITP 2015: 227-233
- Cezary Kaliszyk, Josef Urban, Jirí Vyskocil, Herman Geuvers: Developing Corpus-Based Translation Methods between Informal and Formal Mathematics: Project Description. CICM 2014: 435-439
- C. Kaliszyk, J. Urban, J. Vyskocil: Automating Formalization by Statistical and Semantic Parsing of Mathematics. ITP 2017: 12-27
- Cezary Kaliszyk, Josef Urban, Jirí Vyskocil: System Description: Statistical Parsing of Informalized Mizar Formulas. SYNASC 2017: 169-172
- Q. Wang, C. Kaliszyk, J. Urban: First Experiments with Neural Translation of Informal to Formal Mathematics. CICM 2018: 255-270
- Qingxiang Wang, Chad E. Brown, Cezary Kaliszyk, Josef Urban: Exploration of neural machine translation in autoformalization of mathematics in Mizar. CPP 2020: 85-98

Thanks and Advertisement

- Thanks for your attention!
- **AITP – Artificial Intelligence and Theorem Proving**
- September 4–9, 2022, Aussois, France, aitp-conference.org
- ATP/ITP/Math vs AI/Machine-Learning people, Computational linguists
- Discussion-oriented and experimental
- Grown to 80 people in 2019
- Hybrid in 2022 as in 2021 and 2020
- Invited talks by J. Araujo, K. Buzzard, J. Brandstetter, W. Dean and A. Naibo, M. Rawson, T. Ringer, S. Wolfram