Some Combinations of Machine Learning and Theorem Proving Methods

Josef Urban

Czech Technical University in Prague

Colloquium at the University of Florence
April 29, 2025, Florence

European Research Council
Established by the European Commission

Outline

Computer Understandable (Formal) Math

Learning of Theorem Proving

Examples and Demos

High-level Reasoning Guidance: Premise Selection

Low Level Guidance of Theorem Provers

Mid-level Reasoning Guidance

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

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

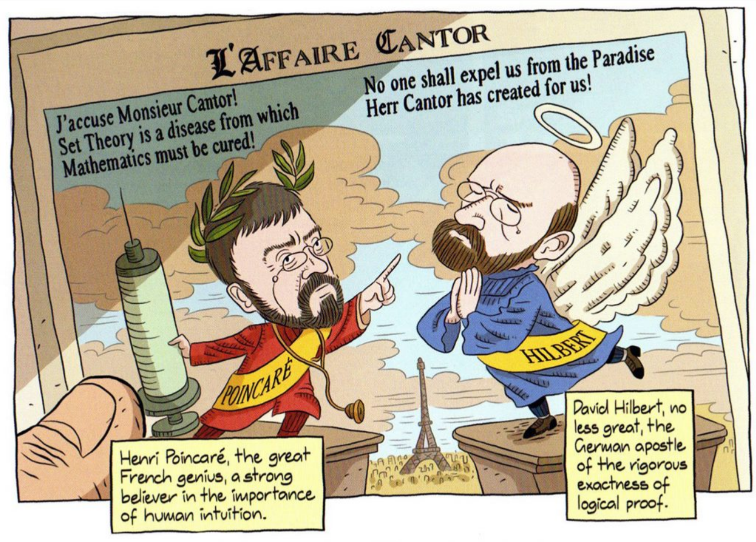
[Adapted from: *Logicomix: An Epic Search for Truth* by A. Doxiadis]

How Do We Automate Math, Science, Programming?

- 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!

Intuition vs Formal Reasoning – Poincaré vs Hilbert

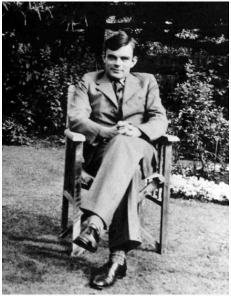


[Adapted from: *Logicomix: An Epic Search for Truth* by A. Doxiadis]

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 ...

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 the newly available computers
- 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

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) ∉ ℚ"
proof
  assume "sqrt (real 2) ∈ ℚ"
  then obtain m n :: nat where
    n_nonzero: "n ≠ 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 cdiv co wceq cn wrex cz cexp
      cmulc sqr2irrlem3 sqr2irrlem5 bi2rexa mtbir cc0 clt wbr wa wi wb nngt0t
      adantr cr ax0re ltmuldivt mp3an1 nnret zret syl2an mpd ancoms 2re 2pos
      sqrgt0i breq2 mpbii syl5bir cc nncnt mulzer2t syl breq1d adantl sylibd
      exp r19.23adv anc2li elnnz syl6ibr impac r19.22i2 mto elq df-nel mpbir )
      CDEZFGWDFHZIWEWDAJZBJZKLZMZBNOZAPOZWKWJANOZWLWFCQLCWGCQLRLMZBNOANOABSWIWM
      ABNNWFWGTUAUBWJWJAPNWFPHZWJWFNHZWNWJWNUCWFUDUEZUFWOWNWJWPWNWIWPBNWNWGNHZW
      IWPUGWNWQUFZWIUCWGRLZWFUDUEZWPWRWTUCWHUDUEZWIWQWNWTXAUHZWQWNUFUCWGUDUEZXB
      WQXCWNWGUIUJWGUKHZWFUKHZXCXBUGZWQWNUCUKHXDXEXFULUCWGWFUMUNWGUOWFUPUQURUSW
      IUCWDUDUEXACUTVAVBWDWHUCUDVCVDVEWQWTWPUHWNWQWSUCWFUDWQWGVFHWSUCMWGVGWGVHV
      IVJVKVLVMVNVOWFVPVQVRVSVTABWDWAUBWDFWBWC $.
      $( [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	5 $\vdash \neg \exists x \in \mathbb{N}\, \exists y \in \mathbb{N}\, (x \uparrow 2) = (2 \cdot (y \uparrow 2))$
2		sqr2irrlem5 10840	6 $\vdash ((x \in \mathbb{N} \wedge y \in \mathbb{N}) \to ((\surd`2) = (x / y) \leftrightarrow (x \uparrow 2) = (2 \cdot (y \uparrow 2))))$
3	2	2rexbiia 2329	5 $\vdash (\exists x \in \mathbb{N}\, \exists y \in \mathbb{N}\, (\surd`2) = (x / y) \leftrightarrow \exists x \in \mathbb{N}\, \exists y \in \mathbb{N}\, (x \uparrow 2) = (2 \cdot (y \uparrow 2)))$
4	1, 3	mtbir 288	4 $\vdash \neg \exists x \in \mathbb{N}\, \exists y \in \mathbb{N}\, (\surd`2) = (x / y)$
5		2re 8838	12 $\vdash 2 \in \mathbb{R}$
6		2pos 8849	12 $\vdash 0 < 2$
7	5, 6	sqrgt0ii 10213	11 $\vdash 0 < (\surd`2)$
8		breq2 3595	11 $\vdash ((\surd`2) = (x / y) \to (0 < (\surd`2) \leftrightarrow 0 < (x / y)))$
9	7, 8	mpbii 200	10 $\vdash ((\surd`2) = (x / y) \to 0 < (x / y))$
10		zre 9029	12 $\vdash (x \in \mathbb{Z} \to x \in \mathbb{R})$
11	10	adantr 444	11 $\vdash ((x \in \mathbb{Z} \wedge y \in \mathbb{N}) \to x \in \mathbb{R})$
12		nnre 8788	12 $\vdash (y \in \mathbb{N} \to y \in \mathbb{R})$
13	12	adantl 445	11 $\vdash ((x \in \mathbb{Z} \wedge y \in \mathbb{N}) \to y \in \mathbb{R})$
14		nngt0 8807	12 $\vdash (y \in \mathbb{N} \to 0 < y)$
15	14	adantl 445	11 $\vdash ((x \in \mathbb{Z} \wedge y \in \mathbb{N}) \to 0 < y)$
16		gt0div 8683	11 $\vdash ((x \in \mathbb{R} \wedge y \in \mathbb{R} \wedge 0 < y) \to (0 < x \leftrightarrow 0 < (x / y)))$
17	11, 13, 15, 16	syl3anc 1145	10 $\vdash ((x \in \mathbb{Z} \wedge y \in \mathbb{N}) \to (0 < x \leftrightarrow 0 < (x / y)))$
18	9, 17	syl5ibr 210	9 $\vdash ((x \in \mathbb{Z} \wedge y \in \mathbb{N}) \to ((\surd`2) = (x / y) \to 0 < x))$
19		simpl 436	9 $\vdash ((x \in \mathbb{Z} \wedge y \in \mathbb{N}) \to x \in \mathbb{Z})$
20	18, 19	jctild 522	8 $\vdash ((x \in \mathbb{Z} \wedge y \in \mathbb{N}) \to ((\surd`2) = (x / y) \to (x \in \mathbb{Z} \wedge 0 < x)))$
21		elnnz 9035	8 $\vdash (x \in \mathbb{N} \leftrightarrow (x \in \mathbb{Z} \wedge 0 < x))$
22	20, 21	syl6ibr 216	7 $\vdash ((x \in \mathbb{Z} \wedge y \in \mathbb{N}) \to ((\surd`2) = (x / y) \to x \in \mathbb{N}))$
23	22	rexlimdva 2414	6 $\vdash (x \in \mathbb{Z} \to (\exists y \in \mathbb{N}\, (\surd`2) = (x / y) \to x \in \mathbb{N}))$
24	23	impac 598	5 $\vdash ((x \in \mathbb{Z} \wedge \exists y \in \mathbb{N}\, (\surd`2) = (x / y)) \to (x \in \mathbb{N} \wedge \exists y \in \mathbb{N}\, (\surd`2) = (x / y)))$
25	24	reximi2 2396	4 $\vdash (\exists x \in \mathbb{Z}\, \exists y \in \mathbb{N}\, (\surd`2) = (x / y) \to \exists x \in \mathbb{N}\, \exists y \in \mathbb{N}\, (\surd`2) = (x / y))$
26	4, 25	mto 165	3 $\vdash \neg \exists x \in \mathbb{Z}\, \exists y \in \mathbb{N}\, (\surd`2) = (x / y)$
27		elq 9308	3 $\vdash ((\surd`2) \in \mathbb{Q} \leftrightarrow \exists x \in \mathbb{Z}\, \exists y \in \mathbb{N}\, (\surd`2) = (x / y))$
28	26, 27	mtbir 288	2 $\vdash \neg (\surd`2) \in \mathbb{Q}$
29		df-nel 2210	2 $\vdash ((\surd`2) \notin \mathbb{Q} \leftrightarrow \neg (\surd`2) \in \mathbb{Q})$
30	28, 29	mpbir 198	1 $\vdash (\surd`2) \notin \mathbb{Q}$

Colors of variables: wff set class

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

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

SCI NEWS

HOME ASTRONOMY SPACE EXPLORATION ARCHAEOLOGY PALEONTOLOGY BIOLOGY PHYSICS MEDICINE GENETICS GEOLOGY MORE

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 Emerge

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

Mirarce 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

Today's Applications

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

search

Home » Other Sciences » Mathematics » October 12, 2012

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

October 12, 2012 by Rob Knies, Microsoft

reddit

Favorites

Email

Print

PDF

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

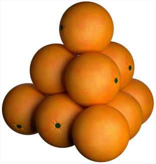
Dark matter 'hurricane' offers chance to detect axions 18 hours ago 36

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

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!

History and Motivation for AI/ML/TP

- Intuition vs Formal Reasoning – Poincaré vs Hilbert, Science & Method
- Turing's 1950 paper: Learning Machines, learn Chess?, undecidability??
- Lenat, Langley, etc: 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 explainable rules/heuristics from 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, feedback loops, internal guidance, ...
- More details – AGI'18 keynote: `https://bit.ly/3qifhg4`
- AI vs DL: Ben Goertzel's Prague talk: `https://youtu.be/Zt2HSTuGBn8`
- Big AI visions: automate/verify math, science, law, (Leibniz, McCarthy, ..)
- Practical impact: boost today's large ITP verification projects

Why Do This Today?

1. Practically Useful for Verification of Complex HW/SW and Math
    - Formal Proof of the Kepler Conjecture (2014 – Hales – 20k lemmas)
    - Formal Proof of the Feit-Thompson Theorem (2 books, 2012 – Gonthier)
    - Verification of several math textbooks and CS algorithms
    - Verification of compilers (CompCert)
    - Verification of OS microkernels (seL4), HW chips (Intel), transport, finance,
    - Verification of cryptographic protocols (Amazon), etc.
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
    - automate/verify them, include law, etc. (Leibniz, McCarthy, ..)
        - What are the components (inductive/deductive thinking)?
        - How to combine them together?

Outline

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
- **mid-level**: invent suitable ATP strategies 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
- ...

Sample of Learning Approaches

- **k-nearest neighbor** – find the *k* nearest neighbors to the query, combine their solutions (simple but very good in ITP when terminology changes)
- **naive Bayes** – compute probabilities of outcomes assuming complete (naive) independence of characterizing features (just multiplying probabilities)
- **support vector machines** – find a good classifying hyperplane, possibly after non-linear transformation of the data (*kernel methods*)
- **neural networks** (statistical ML) – backpropagation, deep learning, convolutional, recurrent, etc.
- **decision trees, random forests** – find good classifying attributes (and/or their values); more explainable
- **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 use to learn the function from input to output (“garbage in garbage out”)
- Feature discovery – a big field
- Deep Learning – design neural architectures that automatically find important high-level features for a task
- 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 patterns/abstractions
- how do we represent math objects (formulas, proofs, ideas) in our mind?

Outline

AI/TP Examples and Demos

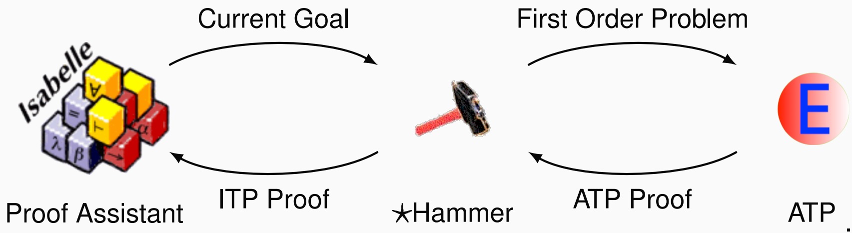
- ENIGMA/hammer proofs of Pythagoras : `https://bit.ly/2MVPAn7` (more at `http://grid01.ciirc.cvut.cz/~mptp/enigma-ex.pdf`) and simplified Carmichael `https://bit.ly/3oGBdRz`,
- 3-phase ENIGMA: `https://bit.ly/3C0Lwa8`, `https://bit.ly/3BWqR6K`
- Long trig proof from 1k axioms: `https://bit.ly/2YZ0OgX`
- Extreme Deepire/AVATAR proof of $\epsilon_0 = \omega^{\omega^{\omega^{\cdot^{\cdot^{\cdot}}}}}$ `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`
- Tactician for Coq: `https://blaauwbroek.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

High-level ATP guidance: Premise Selection

- Early 2003: Can existing ATPs be used over the freshly translated 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).
- Extensive human (math) knowledge obsolete?? (cf. Watson, Debater, ..)
- Since 2004 (my PhD): many examples of nontrivial alternative proofs proposed by the AIs - in Mizar, Flyspeck, Isabelle, ..
- The premise selection algorithms see *wider* than humans

Today's AI-ATP systems ($\star$-Hammers)



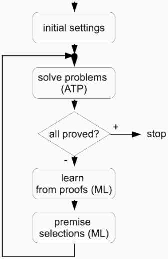
How much can it do?

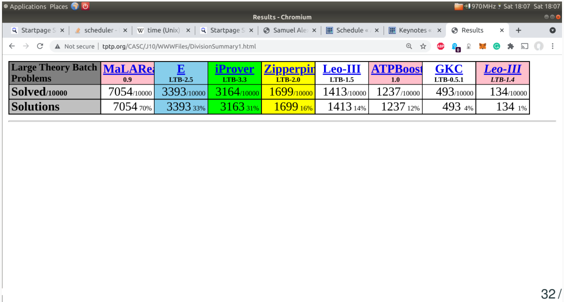
- Mizar / MML – MizAR
- Isabelle (Auth, Jinja) – Sledgehammer
- Flyspeck (including core HOL Light and Multivariate) – HOL(y)Hammer
- HOL4 (Gauthier and Kaliszyk)
- CoqHammer (Czajka and Kaliszyk) - about 40% on Coq standard library

$\approx$ 40-45% success by 2016, 60% on Mizar as of 2021

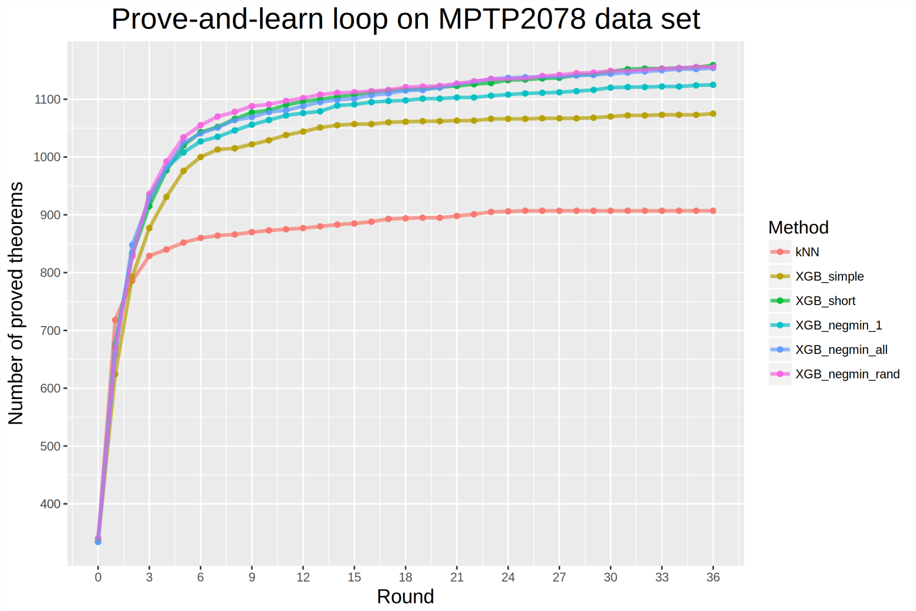
High-level feedback loops – MALARea, ATPBoost

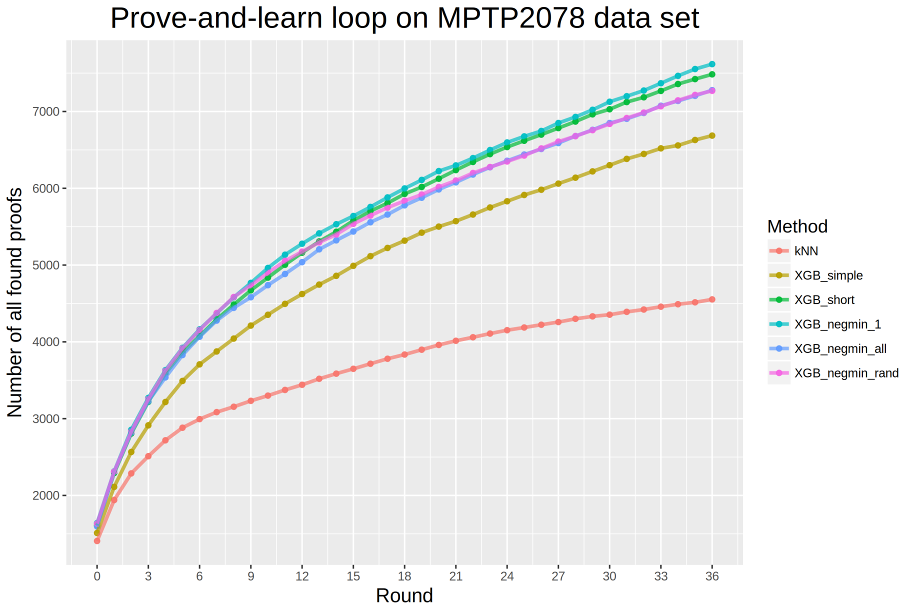
- Machine Learner for Autom. Reasoning (2006) – infinite hammering
- feedback loop interleaving ATP with learning premise selection
- both syntactic and semantic features for characterizing formulas:
- evolving set of finite (counter)models in which formulas evaluated
- winning AI/ATP benchmarks (MPTPChallenge,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

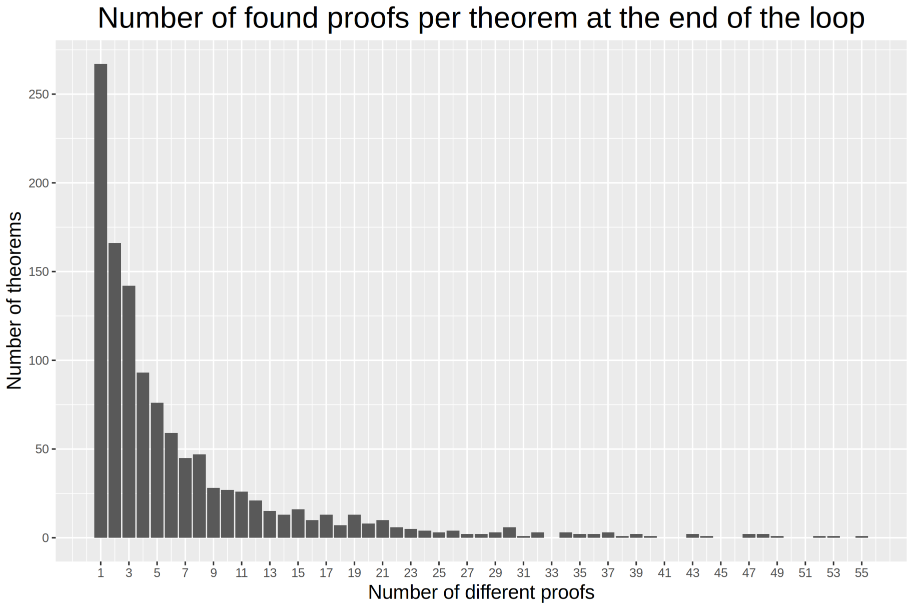




Large Theory Batch Problems	MaLARea 0.9	E LTB-2.5	iProver LTB-3.3	Zipperpin LTB-2.0	Leo-III LTB-1.5	ATPBoost 1.0	GKC LTB-0.5.1	*Leo-III LTB-1.4*
Solved/10000	7054/10000	3393/10000	3164/10000	1699/10000	1413/10000	1237/10000	493/10000	134/10000
Solutions	7054 70%	3393 33%	3163 31%	1699 16%	1413 14%	1237 12%	493 4%	134 1%


Prove-and-learn loop on MPTP2078 data set
Number of proved theorems
Round
Method
kNN
XGB_simple
XGB_short
XGB_negmin_1
XGB_negmin_all
XGB_negmin_rand
400
500
600
700
800
900
1000
1100
0
3
6
9
12
15
18
21
24
27
30
33
36


Prove-and-learn loop on MPTP2078 data set
Number of all found proofs
7000
6000
5000
4000
3000
2000
0
3
6
9
12
15
18
21
24
27
30
33
36
Round
Method
kNN
XGB_simple
XGB_short
XGB_negmin_1
XGB_negmin_all
XGB_negmin_rand


Number of found proofs per theorem at the end of the loop
Number of theorems
Number of different proofs
0
50
100
150
200
250
1
3
5
7
9
11
13
15
17
19
21
23
25
27
29
31
33
35
37
39
41
43
45
47
49
51
53
55

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

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:

$c_1 : P(x)$

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

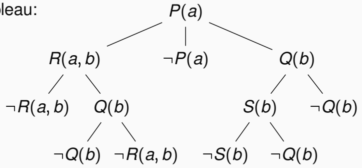
$c_3 : S(x) \lor \neg Q(b)$

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

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

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

Closed Connection Tableau:



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

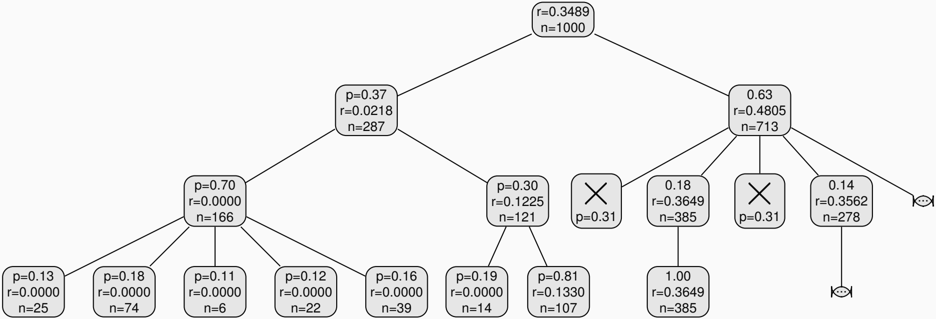
Statistical Guidance of Connection Tableau – rlCoP

- 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:

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

- 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 – rlCoP

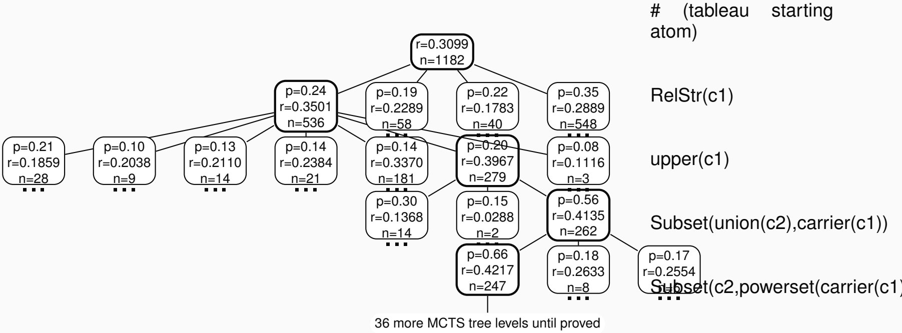
- On 32k Mizar40 problems using 200k inference limit
- nonlearning CoPs:

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

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

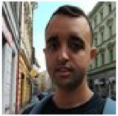
Iteration	1	2	3	4	5	6	7	8
Training proved	12325	13749	14155	14363	14403	14431	14342	**14498**
Testing proved	1354	1519	1566	1595	**1624**	1586	1582	1591

More trees


r=0.3099
n=1182
p=0.24
r=0.3501
n=536
p=0.19
r=0.2289
n=58
p=0.22
r=0.1783
n=40
p=0.35
r=0.2889
n=548
p=0.21
r=0.1859
n=28
p=0.10
r=0.2038
n=9
p=0.13
r=0.2110
n=14
p=0.14
r=0.2384
n=21
p=0.14
r=0.3370
n=181
p=0.20
r=0.3967
n=279
p=0.08
r=0.1116
n=3
p=0.30
r=0.1368
n=14
p=0.15
r=0.0288
n=2
p=0.56
r=0.4135
n=262
p=0.66
r=0.4217
n=247
p=0.18
r=0.2633
n=8
p=0.17
r=0.2554
36 more MCTS tree levels until proved
# (tableau starting atom)
RelStr(c1)
upper(c1)
Subset(union(c2),carrier(c1))
Subset(c2,powerset(carrier(c1)

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 ($\sim$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`

	$\mathcal{S}$	$\mathcal{S} \odot \mathcal{M}_9^0$	$\mathcal{S} \oplus \mathcal{M}_9^0$	$\mathcal{S} \odot \mathcal{M}_9^1$	$\mathcal{S} \oplus \mathcal{M}_9^1$	$\mathcal{S} \odot \mathcal{M}_9^2$	$\mathcal{S} \oplus \mathcal{M}_9^2$	$\mathcal{S} \odot \mathcal{M}_9^3$	$\mathcal{S} \oplus \mathcal{M}_9^3$
solved	**14933**	16574	20366	21564	22839	22413	23467	22910	23753
$\mathcal{S}\%$	+0%	+10.5%	+35.8%	+43.8%	+52.3%	+49.4%	+56.5%	+52.8%	+58.4
$\mathcal{S}+$	+0	+4364	+6215	+7774	+8414	+8407	+8964	+8822	+9274
$\mathcal{S}-$	-0	-2723	-782	-1143	-508	-927	-430	-845	-454

	$\mathcal{S} \odot \mathcal{M}_{12}^3$	$\mathcal{S} \oplus \mathcal{M}_{12}^3$	$\mathcal{S} \odot \mathcal{M}_{16}^3$	$\mathcal{S} \oplus \mathcal{M}_{16}^3$
solved	24159	24701	25100	**25397**
$\mathcal{S}\%$	+61.1%	+64.8%	+68.0%	**+70.0%**
$\mathcal{S}+$	+9761	+10063	+10476	+10647
$\mathcal{S}-$	-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

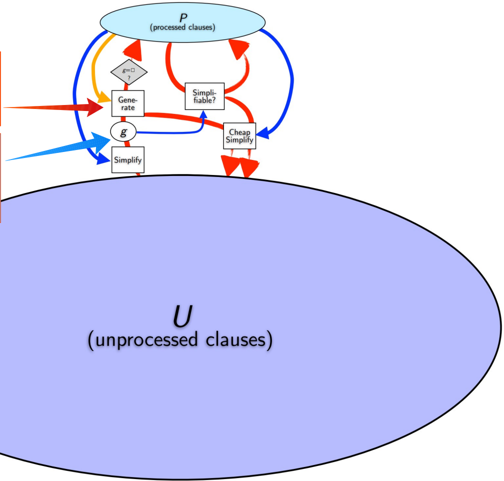
3-phase ENIGMA

Parental Guidance Filter:

Fast – GBDT

Clause Selection Models:

2-phase – GBDT + GNN



More Low-Level Guidance of Various Creatures

- Neural (TNN) clause selection in Vampire (Deepire - M. Suda):
  Learn from clause *derivation trees only*
  Not looking at what it says, just who its ancestors were.
- Fast and surprisingly good: Extreme Deepire/AVATAR proof of
  $\epsilon_0 = \omega^{\omega^{\omega^{\cdot^{\cdot^{\cdot}}}}}$ `https://bit.ly/3Ne4WNX`
- 1193-long proof takes *about the same resources as one GPT-3/4 reply*
- GNN-based guidance in iProver (Chvalovsky, Korovin, Piepenbrock)
- New (*dynamic data*) way of training
- Led to doubled real-time performance of iProver's instantiation mode
- CVC5: neural & GBDT instantiation guidance (Piepenbrock, Jakubuv)
- very recently 20% improvement on Mizar
- Hints method for Otter/Prover9 (Veroff):
- boost inferences on clauses that match a lemma used in a related proof
- 100k-step long proofs in the AIM project (2021)
- symbolic ML - can be combined with statistical - proof completion vectors

Behold: 1-CPU vampiric "GenAI" proving $\epsilon_0 = \omega^{\omega^{\omega^{\cdot^{\cdot^{\cdot}}}}}$

```
% Refutation found. Thanks to Tanya!
% SZS status Theorem for t36_ordinal5
% SZS output start Proof for t36_ordinal5
fof(f2863755,plain,( $false), inference(avatar_sat_refutation,
[... 2500 lines of proof ...]
% SZS output end Proof for t36_ordinal5
% ------------------------------
% Version: Vampire 4.5.1 (commit 110f4142 on 2020-10-16 16:55:15 +0200)
% Termination reason: Refutation
% Input formulas: 73
% Proof axioms: 49
% Proof steps: 1193
% Main loop iterations started: 38065
% Generated clauses: 2392519
% SAT solver time: 181.936 s
% congruence closure: 167.665 s ( own 156.041 s )
% neural model evaluation: 18.493 s
% other: 503.976 s ( own 26.185 s )
```

Outline

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 rlCoP: 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 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

Conjecturing and Proof Synthesis by Neural Language models

- 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 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 ...

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

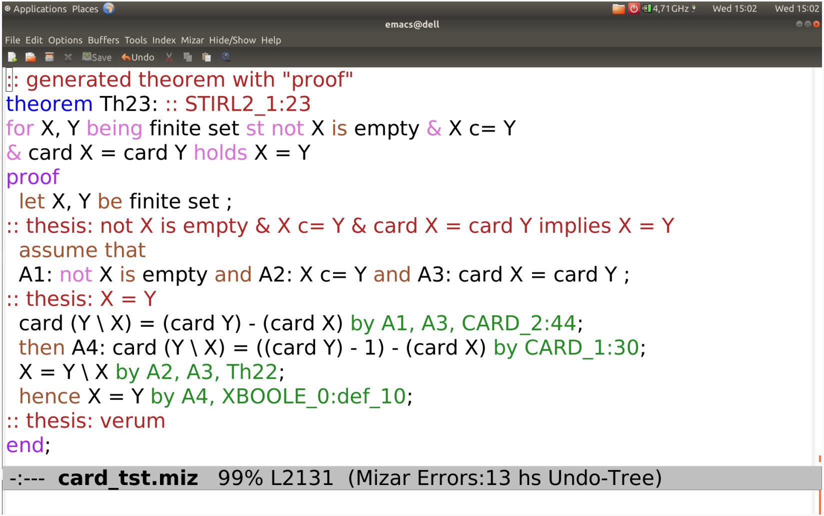


Figure: Fake full declarative GPT-2 “proof” - typechecks!

A correct conjecture that was too hard to prove

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

```
theorem Th10: :: GROUPP_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
```

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.
```

Neural Autoformalization (Wang et al., 2018)

- generate ca 1M Latex/Mizar pairs based on Bancerek's work
- 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 very important for neural training – our biggest bottleneck (you can help!)
- Recent addition: unsupervised methods (Lample et all 2018) – no need for aligned data!

Neural Autoformalization data

Rendered $\LaTeX$	If $X \subseteq Y \subseteq Z$, then $X \subseteq Z$.
Mizar	`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 LaTeX	Suppose s_8 is convergent and s_7 is convergent . Then $\mathop{\rm lim}(s_8 + s_7) = \mathop{\rm lim} s_8 + \mathop{\rm lim} s_7$
Input LaTeX	`Suppose $ { s _ { 8 } } $ is convergent and $ { s _ { 7 } } $ is convergent . Then $ \mathop { \rm lim } ( { s _ { 8 } } { + } { s _ { 7 } } ) \mathrel { = } \mathop { \rm lim } { s _ { 8 } } { + } \mathop { \rm lim } { s _ { 7 } } $ .`
Correct	`seq1 is convergent & seq2 is convergent implies lim ( seq1 + seq2 ) = ( lim seq1 ) + ( lim seq2 ) ;`
Snapshot-1000	`x in dom f implies ( x * y ) * ( f | ( x | ( y | ( y | y ) ) ) ) = ( x | ( y | ( y | ( y | y ) ) ) ) ) ;`
Snapshot-2000	`seq is summable implies seq is summable ;`
Snapshot-3000	`seq is convergent & lim seq = 0c implies seq = seq ;`
Snapshot-4000	`seq is convergent & lim seq = lim seq implies seq1 + seq2 is convergent ;`
Snapshot-5000	`seq1 is convergent & lim seq2 = lim seq2 implies lim_inf seq1 = lim_inf seq2 ;`
Snapshot-6000	`seq is convergent & lim seq = lim seq implies seq1 + seq2 is convergent ;`
Snapshot-7000	`seq is convergent & seq9 is convergent implies lim ( seq + seq9 ) = ( lim seq ) + ( lim seq9 ) ;`

QSynt: Semantics-Aware Synthesis of Math Objects

- Gauthier (et al) 2019-24
- Synthesize math expressions based on semantic characterizations
- i.e., not just on the syntactic descriptions (e.g. proof situations)
- Tree Neural Nets and Monte Carlo Tree Search (a la AlphaZero)
- Recently also various (small) *language models* with their search methods
- Invent programs for OEIS sequences FROM SCRATCH (no LLM cheats)
- 126k OEIS sequences (out of 350k) solved so far (670 iterations): `https://www.youtube.com/watch?v=24oejR9wsXs`, `http://grid01.ciirc.cvut.cz/~thibault/qsynt.html`
- ~4.5M explanations invented: 50+ different characterizations of primes
- Non-neural (Turing complete) symbolic computing and semantics collaborate with the statistical/neural learning
- Program evolution governed by high-level criteria (Occam, efficiency)

OEIS: $\geq$ 350000 finite sequences

The OEIS is supported by the many generous donors to the OEIS Foundation.

THE ON-LINE ENCYCLOPEDIA OF INTEGER SEQUENCES ®

founded in 1964 by N. J. A. Sloane

2 3 5 7 11 | Search | Hints

(Greetings from The On-Line Encyclopedia of Integer Sequences!)

Search: **seq:2,3,5,7,11**

Displaying 1-10 of 1163 results found. page 1 2 3 4 5 6 7 8 9 10 ... 117

Sort: relevance | references | number | modified | created Format: long | short | data

A000040 The prime numbers. +30
(Formerly M0652 N0241) 10150

2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61, 67, 71, 73, 79, 83, 89, 97, 101, 103, 107, 109, 113, 127, 131, 137, 139, 149, 151, 157, 163, 167, 173, 179, 181, 191, 193, 197, 199, 211, 223, 227, 229, 233, 239, 241, 251, 257, 263, 269, 271 (list; graph; refs; listen; history; text; internal format)

OFFSET 1,1

COMMENTS See A065091 for comments, formulas etc. concerning only odd primes. For all information concerning prime powers, see A000961. For contributions concerning "almost primes" see A002808.
A number p is prime if (and only if) it is greater than 1 and has no positive divisors except 1 and p.
A natural number is prime if and only if it has exactly two (positive) divisors.
A prime has exactly one proper positive divisor, 1.

Generating programs for OEIS sequences

$0, 1, 3, 6, 10, 15, 21, \ldots$

An undesirable large program:

```
if x = 0 then 0 else
if x = 1 then 1 else
if x = 2 then 3 else
if x = 3 then 6 else ...
```

Small program (Occam's Razor):

$$\sum_{i=1}^{n} i$$

Fast program (efficiency criteria):

$$\frac{n \times n + n}{2}$$

Programming language

- Constants: $0, 1, 2$
- Variables: x, y
- Arithmetic: $+, -, \times, div, mod$
- Condition : if $\ldots \leq 0$ then ... else ...
- $loop(f, a, b) := u_a$ where $u_0 = b$,

$$u_n = f(u_{n-1}, n)$$

- Two other loop constructs: *loop2*, a while loop

Example:

$2^{\mathbf{x}} = \prod_{y=1}^{x} 2 = loop(2 \times x, \mathbf{x}, 1)$
$\mathbf{x}! = \prod_{y=1}^{x} y = loop(y \times x, \mathbf{x}, 1)$

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:

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

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

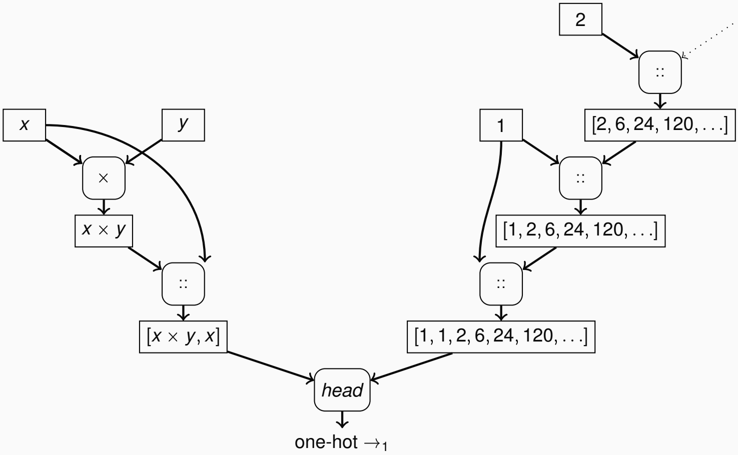
- Programs are built in reverse polish notation
- Start from an empty stack
- Use ML to repeatedly choose the next operator to push on top of a stack
- Example: Factorial is $loop(\lambda(x, y). \ x \times y, x, 1)$, built by:

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

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

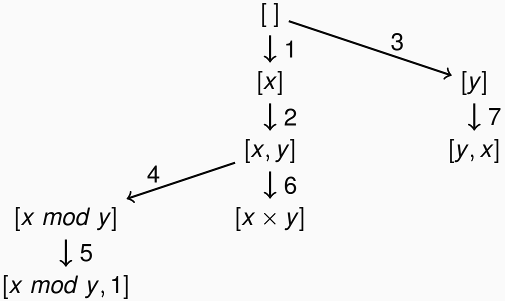
QSynt: Training of the Neural Net Guiding the Search

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



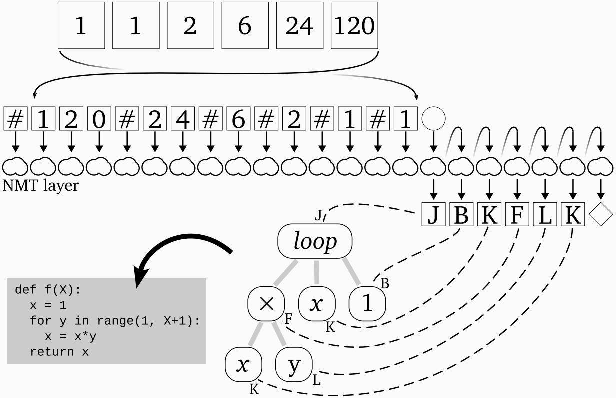
QSynt program search - Monte Carlo search tree

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



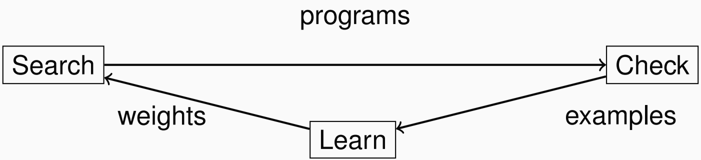
Encoding OEIS for Language Models

- Input sequence is a series of digits
- Separated by an additional token # at the integer boundaries
- Output program is a sequence of tokens in Polish notation
- Parsed by us to a syntax tree and translatable to Python
- Example: $a(n) = n!$




```
def f(X):
  x = 1
  for y in range(1, X+1):
    x = x*y
  return x
```

Search-Verify-Train Positive Feedback Loop



- Analogous to our Prove/Learn feedback loops in learning-guided proving (since 2006 – Machine Learner for Automated Reasoning – MaLARea))
- However, the OEIS setting allows much faster feedback on *symbolic conjecturing*

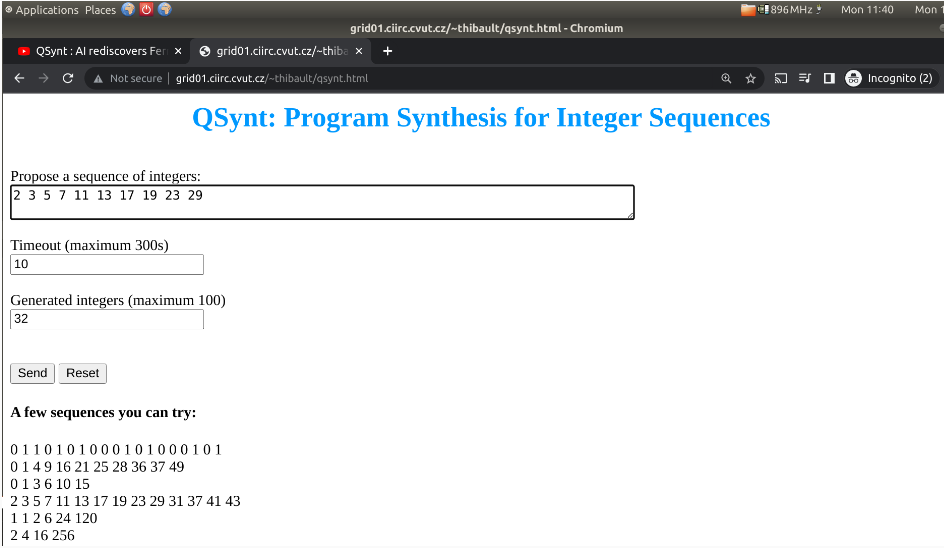
Search-Verify-Train Feedback Loop for OEIS

- **search phase**: LM synthesizes many programs for input sequences
- typically 240 candidate programs for each input using beam search
- 84M programs for OEIS in several hours on the GPU (depends on model)
- **checking phase**: the millions of programs efficiently evaluated
- resource limits used, fast indexing structures for OEIS sequences
- check if the program generates *any* OEIS sequence (hindsight replay)
- we keep the shortest (Occams's razor) and fastest program (efficiency)
- from iter. 501, we also keep the program with the best speed/length ratio
- **learning phase**: LM trains to translate the "solved" OEIS sequences into the best program(s) generating them
- from iter. 336: train LMs to transform (generalization, optimization)
- our learned version of human-coded methods like ILP and compilation

Search-Verify-Train Feedback Loop

- The weights of the LM either trained from scratch or continuously updated
- This yields *new minds vs seasoned experts* (who have seen it all)
- We also train experts on varied selections of data, in varied ways
- Orthogonality: common in theorem proving - different experts help
- Each iteration of the self-learning loop discovers more solutions
- ... also improves/optimizes existing solutions
- The alien mathematician thus self-evolves
- Occam's razor and efficiency are used for its weak supervision
- Quite different from today's LLM approaches:
- LLMs do one-time training on everything human-invented
- Our alien instead starts from zero knowledge
- Evolves increasingly nontrivial skills, may diverge from humans
- Turing complete (unlike Go/Chess) – arbitrary complex algorithms

QSynt web interface for program invention



QSynt inventing Fermat pseudoprimes

Positive integers k such that $2^k \equiv 2 \mod k$. ($341 = 11 * 31$ 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 + compr(\x.loop(\(x,i).2*x + 2, x, 2) mod (x + 2), x)
Run the equivalent Python program here or in the window below:
```

Brython

Tutorial Demo Documentation Console Editor Gallery Resources

English

Brython version: 3.10.6

run Python Javascript Share code

```python
def f2(X):
  x = 2
  for i in range (1,X + 1):
    x = 2*x + 2
  return x

def f1(X):
  x,i = 0,0
  while i <= X:
    if f2(x) % (x + 2) <= 0:
      i = i + 1
    x = x + 1
  return x - 1

def f0(X):
  return 2 + f1(X)

for x in range(32):
  print (f0(x))

```

```
2
3
5
7
11
13
17
19
23
29
31
37
41
43
47
53
59
61
67
```

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 (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.: $(n-1)! \equiv -1 \mod n$)

```
First 32 generated numbers (f(0),f(1),f(2),...):
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 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) := (loop(\(x,i).x * i, x, x) mod (x + 1)) mod 2
Run the equivalent Python program here or in the window below:
```

Speed Evolution – Technology Breakthroughs

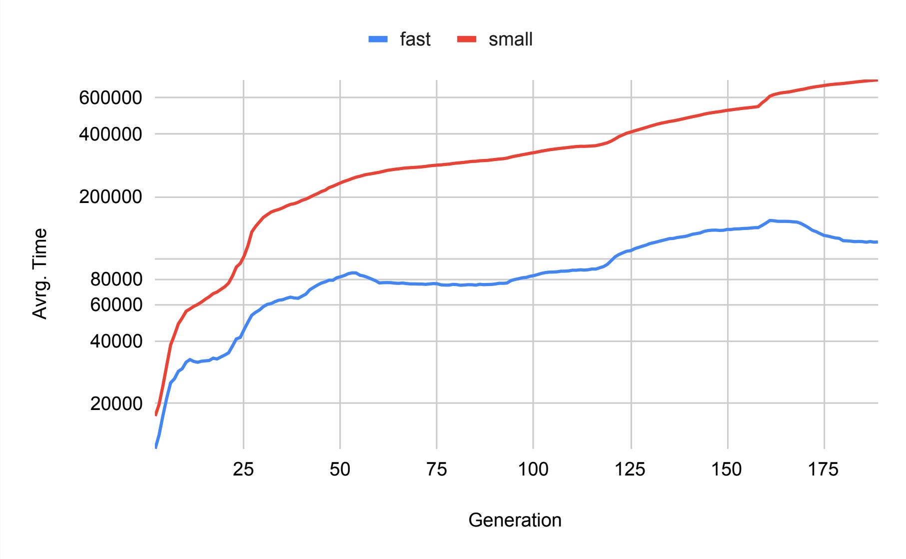
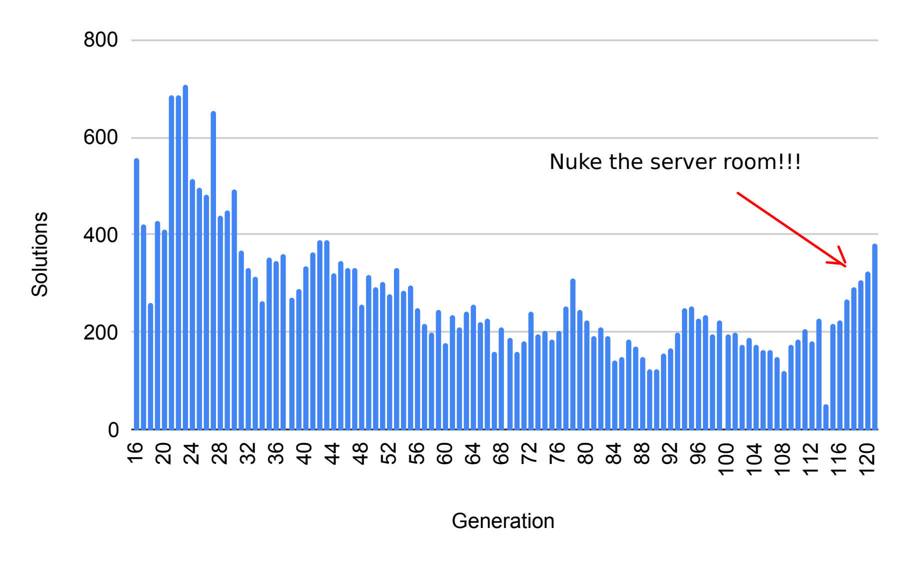
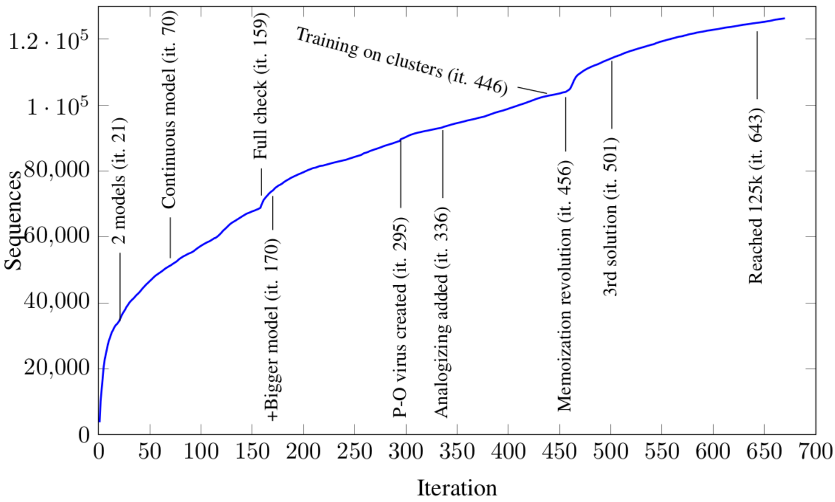


Figure: Avrg. time in iterations

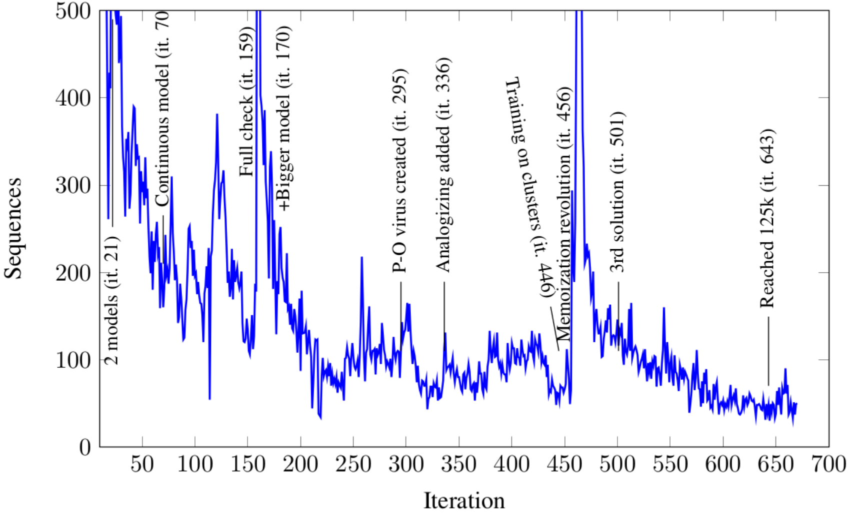
Singularity Take-Off X-mas Card



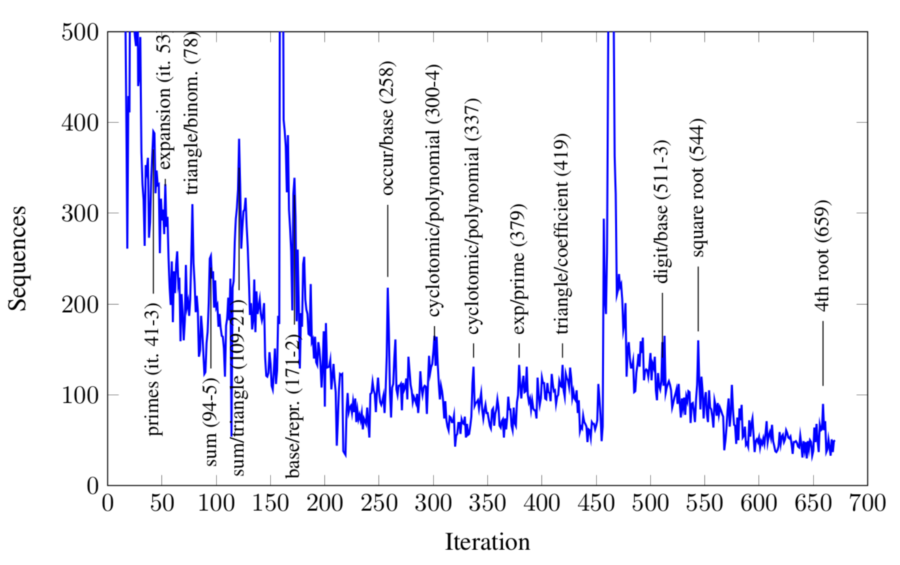
Human Made Technology Jumps



Human Made Technology Jumps



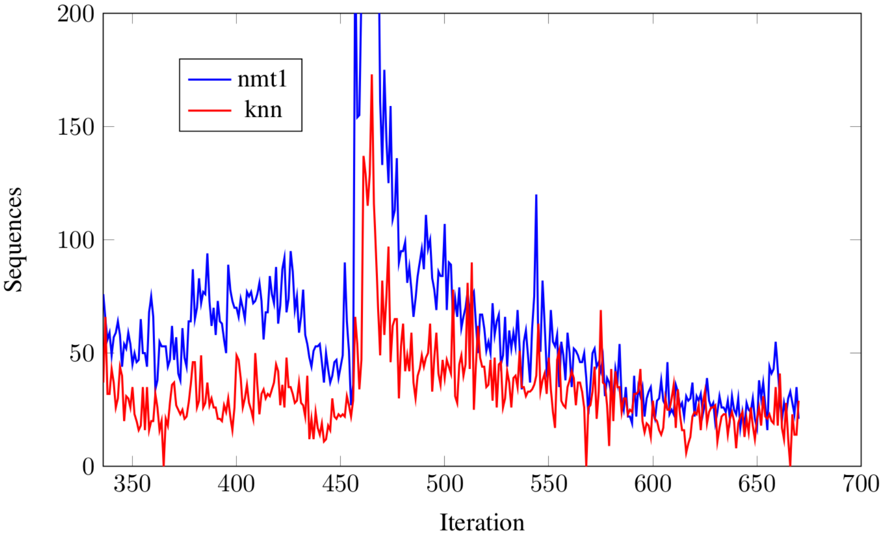
Some Automatic Technology Jumps



Some Invented Explanations

- `https://oeis.org/A4578`: Expansion of sqrt(8) in base 3:
  loop2(((y * y) div (x + y)) + y, y, x + x, 2, loop((1 + 2) * x, x, 2)) mod (1 + 2)
- `https://oeis.org/A4001`: Hofstadter-Conway $10k seq: a(n) = a(a(n-1)) + a(n-a(n-1)) with a(1) = a(2) = 1:
  loop(push(loop(pop(x), y-x,pop(x)),x) + loop(pop(x), x-1, x), x - 1, 1)
- `https://oeis.org/A40`: prime numbers:
  2 + compr((loop(x * y, x, 2) + x) mod (2 + x), x)
- `https://oeis.org/A30184`: Expand $\eta(q) * \eta(q^3) * \eta(q^5) * \eta(q^{15})$ in powers of q (elliptic curves):
  loop(push(loop((pop(x) * loop(if (pop(x) mod y) <= 0 then (x - loop(if (x mod (1 + (y + y))) <= 0 then (x + x) else x, 2, y)) else x, y, push(0, y))) + x, y, push(0, x)), x) div y, x, 1)
- `https://oeis.org/A51023`: Wolfram's $30k Rule 30 automaton:
  loop2(y, y div 2, x, 1, loop2(loop2((((y div (0 - (2 + 2))) mod 2) + x) + x, y div 2, y, 1, loop2(((y mod 2) + x) + x, y div 2, y, 1, x)), 2 + y, x, 0, 1)) mod 2
- `https://oeis.org/A2580`: $\sqrt[3]{2}$ Hales's blog: `https://t.ly/tHs1d`

Translation vs Transformation



Generalization of the Solutions to Larger Indices

- Are the programs correct?
- Can we experimentally verify Occam's razor?
  (implications for how we should be designing ML/AI systems!)
- OEIS provides additional terms for some of the OEIS entries
- Among 78118 solutions, 40,577 of them have a b-file with 100 terms
- We evaluate both the small and the fast programs on them
- Here, 14,701 small and 11,056 fast programs time out.
- 90.57% of the remaining slow programs check
- 77.51% for the fast programs
- This means that **SHORTER EXPLANATIONS ARE MORE RELIABLE!**
  (Occam was right, so why is everybody building trillion-param LLMs???)
- Common error: reliance on an approximation of a real number, such as π.

Are two QSynt programs equivalent?

- As with primes, we often find many programs for one OEIS sequence
- Currently we have almost 4.5M programs for the 126k sequences
- It may be quite hard to see that the programs are equivalent
- Extend to Schmidhuber's Gödel Machine?
- A simple example for $0, 2, 4, 6, 8, \dots$ 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 Janota & Gauthier:

```
(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
```

Infinite Math-Nerd Sniping

- We have 4.5M problems for math nerds like this one:
- JU: *This thing works for the first 1k values (just checked) - any idea why?*
- `https://oeis.org/A004578` - Expansion of sqrt(8) in base 3.
- loop2(((y * y) div (x + y)) + y, y, x + x, 2, loop((1 + 2) * x, x, 2)) mod (1 + 2)
- MO: *Not a proof, just a rough idea: The program iterates the function q |-> 2+q / 1+q, where q is a rational number. This converges to sqrt(2). The number q is represented by an integer 'a' such that* $a = 3^x * (2 * q)$, *where 'x' is the input. Once the approximation is good enough,* $a = floor(3^x * sqrt(8))$, *so a mod 3 is the digit we want.*

Serious Math Conjecturing – Elliptic Curves

- Sander Dahmen: *Here are some OEIS labels related to elliptic curves (and hence modular forms), ordered by difficulty. It would be interesting to know if some of these appear in your results.*
- A006571 A030187 A030184 A128263 A187096 A251913
- JU: *We have the first three:*
- A6571 : loop((push(loop((pop(x) * loop(if (pop(x) mod y) <= 0 then ((if (y mod loop(1 + (x + x), 2, 2)) <= 0 then (x - y) else x) - y) else x, y, push(0, y))) + x, y, push(0, x)), x) * 2) div y, x, 1)
- A30187 : loop(push(loop((pop(x) * loop(if (pop(x) mod y) <= 0 then (x - loop(if (x mod (((2 + y) * y) - 1)) <= 0 then (x + x) else x, 2, y)) else x, y, push(0, y))) + x, y, push(0, x)), x) div y, x, 1)
- A30184 : loop(push(loop((pop(x) * loop(if (pop(x) mod y) <= 0 then (x - loop(if (x mod (1 + (y + y))) <= 0 then (x + x) else x, 2, y)) else x, y, push(0, y))) + x, y, push(0, x)), x) div y, x, 1)

A6571: Expansion of $q * Product_{k>=1}(1-q^k)^2 * (1-q^{11*k})^2$
A30187: Expansion of $\eta(q) * \eta(q^2) * \eta(q^7) * \eta(q^{14})$ in powers of q.
A30184: Expansion of $\eta(q) * \eta(q^3) * \eta(q^5) * \eta(q^{15})$ in powers of q.

More Bragging

- Hofstadter-Conway $10000 sequence: a(n) = a(a(n-1)) + a(n-a(n-1)) with a(1) = a(2) = 1.
- D. R. Hofstadter, Analogies and Sequences: Intertwined Patterns of Integers and Patterns of Thought Processes, Lecture in DIMACS Conference on Challenges of Identifying Integer Sequences, 2014.

```
Date: Sun, Mar 17, 2024
To: <dughof@indiana.edu>

Dear Douglas,

our system [1] has today (iteration 552) found a solution of
https://oeis.org/A004074. The solution in Thibault’s programming
language [1] (with push/pop added on top of [1]) is:

((2*loop(push(loop(pop(x),x-1,x),x)+loop(pop(x),y-x,pop(x)),x-1,1))-1)-x

The related A4001 was solved in iteration 463 and the solution is:
loop(push(loop(pop(x), y-x,pop(x)),x) + loop(pop(x), x-1, x), x - 1, 1)
```

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`
- Big challenge: Learn complicated symbolic algorithms (not black box - motivates also our OEIS research)

Thanks and Advertisement

- Thanks for your attention!
- AITP – Artificial Intelligence and Theorem Proving
- August 31 - September 5, 2025, 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