Machine Learning and Theorem Proving

Josef Urban

Czech Technical University in Prague

The 20th Reasoning Web Summer School
September 19–22, 2024, Bucharest
https://t.ly/yKHBm

European Research Council
Established by the European Commission

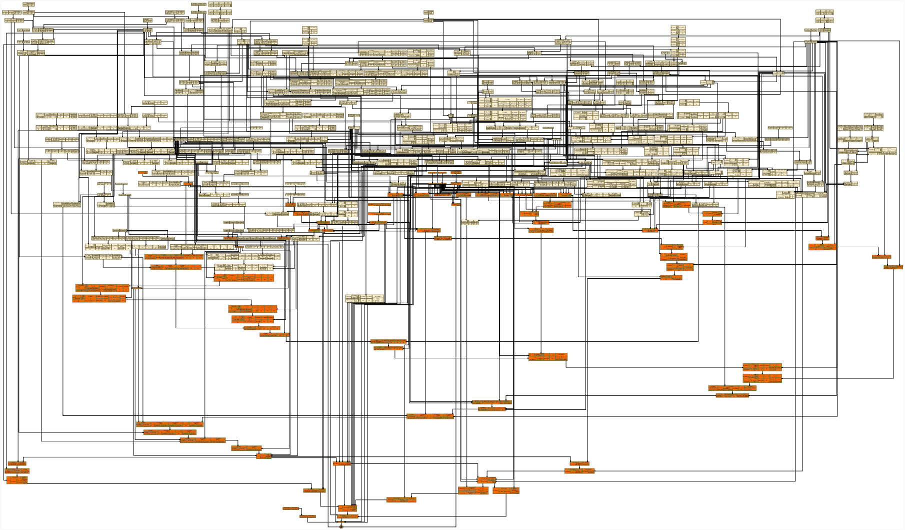
Quick intro: *Prove/Learn feedback loop* on formal math

- Done on 57880 Mizar Mathematical Library formal math problems
- Efficient ML-guidance inside the best ATPs (E prover and more)
- Training of the ML-guidance is *interleaved* with proving harder problems
- Ultimately a 70% improvement over the original strategy:
- ... from 14933 proofs to 25397 proofs (all 10s CPU - no cheating)
- 75% of the Mizar corpus reached in July 2021 - higher times and many runs: `https://github.com/ai4reason/ATP_Proofs`
- Details in our Mizar60 paper: `https://arxiv.org/abs/2303.06686`

	$\mathcal{S}$	$\mathcal{S} \odot \mathcal{M}_9^0$	$\mathcal{S} \oplus \mathcal{M}_9^0$	$\mathcal{S} \odot \mathcal{M}_9^1$	$\mathcal{S} \oplus \mathcal{M}_9^1$	$\mathcal{S} \odot \mathcal{M}_9^2$	$\mathcal{S} \oplus \mathcal{M}_9^2$	$\mathcal{S} \odot \mathcal{M}_9^3$	$\mathcal{S} \oplus \mathcal{M}_9^3$
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

Can you do this in 4 minutes? (example proof)



Can you do this in 4 minutes?

```
theorem Th31: :: BORSUK_5:31
 for A being Subset of R^1
 for a, b being real number st a < b & A = RAT (a,b) holds
 Cl A = [.a,b.]
proof
 let A be Subset of R^1; :: thesis:
 let a, b be real number ; :: thesis:
 assume that
 A1: a < b and
 A2: A = RAT (a,b) ; :: thesis:
 reconsider ab = ].a,b.[, RT = RAT as Subset of R^1 by NUMBERS:12, TOPMETR:17;
 reconsider RR = RAT /\ ].a,b.[ as Subset of R^1 by TOPMETR:17;
 A3: the carrier of R^1 /\ (Cl ab) = Cl ab by XBOOLE_1:28;
 A4: Cl RR c= (Cl RT) /\ (Cl ab) by PRE_TOPC:21;
 thus Cl A c= [.a,b.] :: according to XBOOLE_0:def 10 :: thesis:
 proof
  let x be set ; :: according to TARSKI:def 3 :: thesis:
  assume x in Cl A ; :: thesis:
  then x in (Cl RT) /\ (Cl ab) by A2, A4;
  then x in the carrier of R^1 /\ (Cl ab) by Th15;
  hence x in [.a,b.] by A1, A3, Th16; :: thesis:
 end;
 thus [.a,b.] c= Cl A :: thesis:
 proof
  let x be set ; :: according to TARSKI:def 3 :: thesis:
  assume A5: x in [.a,b.] ; :: thesis:
  then reconsider p = x as Element of RealSpace by METRIC_1:def 13;
  A6: a <= p by A5, XXREAL_1:1;
  A7: p <= b by A5, XXREAL_1:1;
  per cases by A7, XXREAL_0:1;
   suppose A8: p < b ; :: thesis:
    now :: thesis:
     let r be real number ; :: thesis:
     reconsider pp = p + r as Element of RealSpace by METRIC_1:def 13, XREAL_0:def 1;
     set pr = min (pp,((p + b) / 2));
     A9: min (pp,((p + b) / 2)) <= (p + b) / 2 by XXREAL_0:17;
     assume A10: r > 0 ; :: thesis:
     p < min (pp,((p + b) / 2))
     proof
      per cases by XXREAL_0:15;
       suppose min (pp,((p + b) / 2)) = pp ; :: thesis:
        hence p < min (pp,((p + b) / 2)) by A10, XREAL_1:29; :: thesis:
       end;
       suppose min (pp,((p + b) / 2)) = (p + b) / 2 ; :: thesis:
        hence p < min (pp,((p + b) / 2)) by A8, XREAL_1:226; :: thesis:
       end;
      end;
     end;
     then consider Q being rational number such that
     A11: p < Q and
     A12: Q < min (pp,((p + b) / 2)) by RAT_1:7;
     (p + b) / 2 < b by A8, XREAL_1:226;
     then min (pp,((p + b) / 2)) < b by A9, XXREAL_0:2;
     then A13: Q < b by A12, XXREAL_0:2;
     min (pp,((p + b) / 2)) <= pp by XXREAL_0:17;
     then A14: (min (pp,((p + b) / 2))) - p <= pp - p by XREAL_1:9;
     reconsider P = Q as Element of RealSpace by METRIC_1:def 13, XREAL_0:def 1;
     P - p < (min (pp,((p + b) / 2))) - p by A12, XREAL_1:9;
     then P - p < r by A14, XXREAL_0:2;
     then dist (p,P) < r by A11, Th14;
     then A15: P in Ball (p,r) by METRIC_1:11;
     a < Q by A6, A11, XXREAL_0:2;
     then A16: Q in ].a,b.[ by A13, XXREAL_1:4;
     Q in RAT by RAT_1:def 2;
     then Q in A by A2, A16, XBOOLE_0:def 4;
     hence Ball (p,r) meets A by A15, XBOOLE_0:3; :: thesis:
    end;
    hence x in Cl A by GOBOARD6:92, TOPMETR:def 6; :: thesis:
```

Can you do this in 4 minutes?

README.md

Topology - the closure of rationals on (a,b) is [a,b]

359-long proof in 234s using 3-phase ENIGMA, shifting context and aggressive subsumption.

for A being Subset of R^1 for a, b being real number st a < b & A = RAT (a,b) holds Cl A = [.a,b.]

The Mizar proof takes 80 lines:

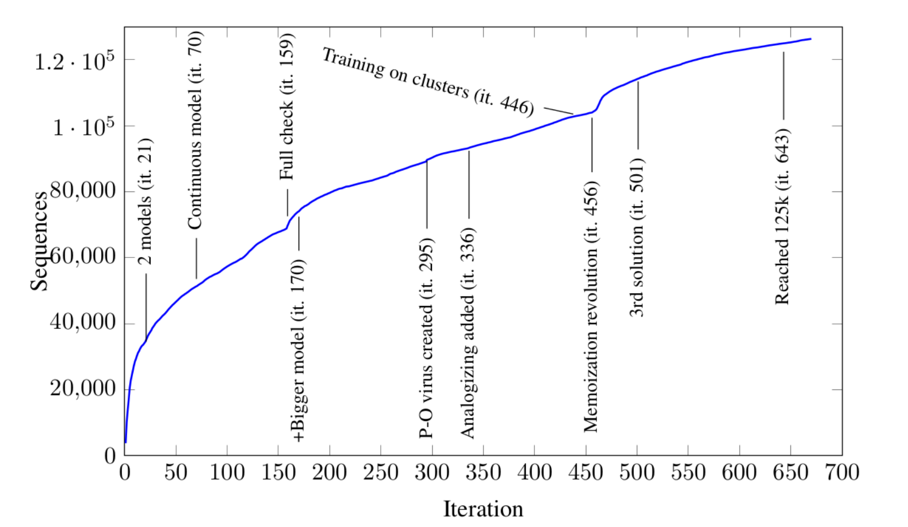
http://grid01.ciirc.cvut.cz/~mptp/7.13.01_4.181.1147/html/borsuk_5.html#T31

E proof (3-phase parental+lgb+gnn-server plus shifting context plus aggr subsumption) using 38 of the 101 heuristically selected premises (subproblem minimization): http://grid01.ciirc.cvut.cz/~mptp/enigma_prf/t31_borsuk_5

/local1/mptp/parents/out2/2pb3l8-query1024-ctx1536-w0-coop-srv-local1-f1711-jj1-zar-parents_nothr_gnnm2_solo1_0.05_0.005_0.1_fw.minsub65all_240s_fw/t31_borsuk_5

```
# Proof object clause steps             : 359
# Proof object initial clauses used     : 56
# Proof object initial formulas used    : 38
# Proof object simplifying inferences   : 180
# Parsed axioms                         : 101
# Initial clauses in saturation         : 153
# Processed clauses                     : 7274
# ...remaining for further processing   : 4883
# Generated clauses                     : 438702
# ...frozen by parental guidance        : 133869
# ...aggressively subsumed              : 83871
# User time                 : 234.274 s
```

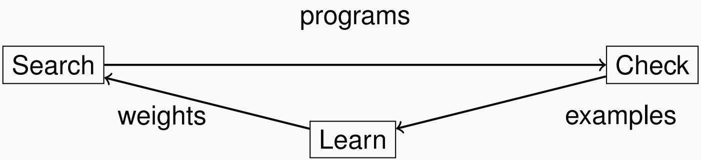
Intro2: *Search/Check/Learn feedback loop* on OEIS



Intro2: *Search/Check/Learn feedback loop* on OEIS

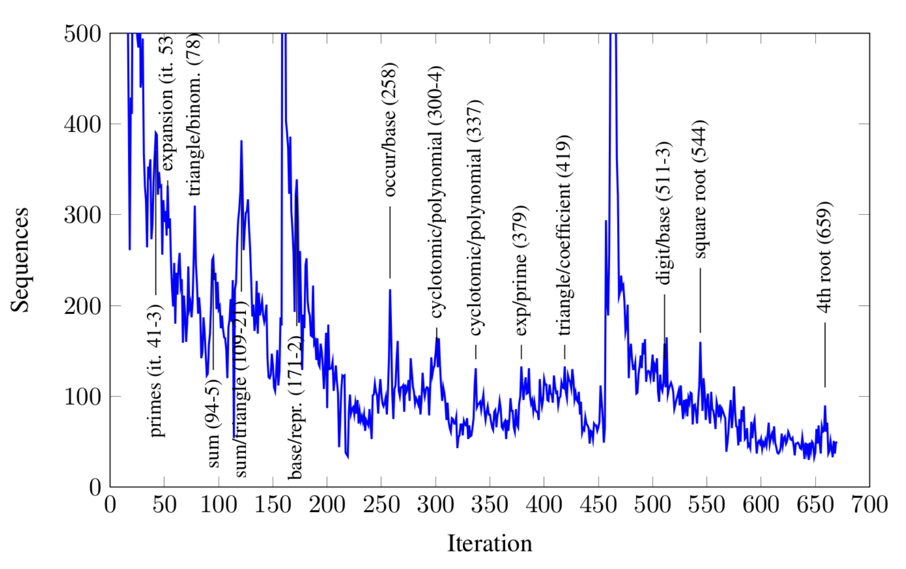
- A machine can find explanations for over 125k OEIS sequences
- This is done *from scratch*, without any domain knowledge
- N. Sloane: The OEIS: *A Fingerprint File for Mathematics* (2021)
- About 350k integer sequences in 2021 from all parts of math
- We use a simple Search-Verify-Train positive feedback loop
- 670 iterations and still refuses to plateau - counters RL wisdom
- Since it interleaves symbolic breakthroughs and statistical learning?
- The electricity bill is only $1k-$3k, you can do this at home
- ~4.5M explanations invented: 50+ different characterizations of primes
- Program evolution governed by high-level criteria (Occam, efficiency)
- Connections to Solomonoff Induction, AIXI, Gödel Machine?

Search-Verify-Train Positive Feedback Loop



- Small Turing-complete DSL for our programs, e.g.:
  $2^{\mathbf{x}} = \prod_{y=1}^{x} 2 = \mathit{loop}(2 \times x, \mathbf{x}, 1)$
  $\mathbf{x}! = \prod_{y=1}^{x} y = \mathit{loop}(y \times x, \mathbf{x}, 1)$
- 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*

Some Automatic Technology Jumps



Some Automatic Technology Jumps

- iter 53: expansion/prime: A29363 Expansion of $1/((1 - x^4)(1 - x^7)(1 - x^9)(1 - x^{10}))$
- iter 78: triangle/binomial: A38313 Triangle whose (i,j)-th entry is $binomial(i, j) * 10^{i-j} * 11^j$
- iter 94-5: sum: A100192 $a(n) = Sum_{k=0..n} binomial(2n, n + k) * 2^k$
- 109-121: sum/triangle: A182013 Triangle of partial sums of Motzkin numbers
- 171-2: base/representation: A39080 n st base-9 repr. has the same number of 0's and 4's
- 258: occur/base: A44533 n st "2,0" occurs in the base 7 repr of n but not of $n + 1$
- 300-304: cyclotomic/polynomial: A14620 Inverse of 611th cyclotomic polynomial
- 379: exp/prime: A124214 E.g.f.: $exp(x)/(2 - exp(3 * x))^{1/3}$
- 419: triangle/coefficient: A15129 Triangle of (Gaussian) q-binomial coefficients for $q = -13$
- 511,3: digit/base/prime: A260044 Primes with decimal digits in 0,1,3.
- 544: square root: A10538 Decimal expansion of square root of 87.
- 659: 4th root: A11084 Decimal expansion of 4th root of 93.

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)
```

Outline

Quick Intro

Motivation, Learning vs. Reasoning

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

Quotes: Learning vs. Reasoning vs. Guessing

“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

Certainly, let us learn proving, but also let us learn guessing.

– G. Polya - Mathematics and Plausible Reasoning

Galileo once said, "Mathematics is the language of Science." Hence, facing the same laws of the physical world, alien mathematics must have a good deal of similarity to ours.

– R. Hamming - Mathematics on a Distant Planet

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?

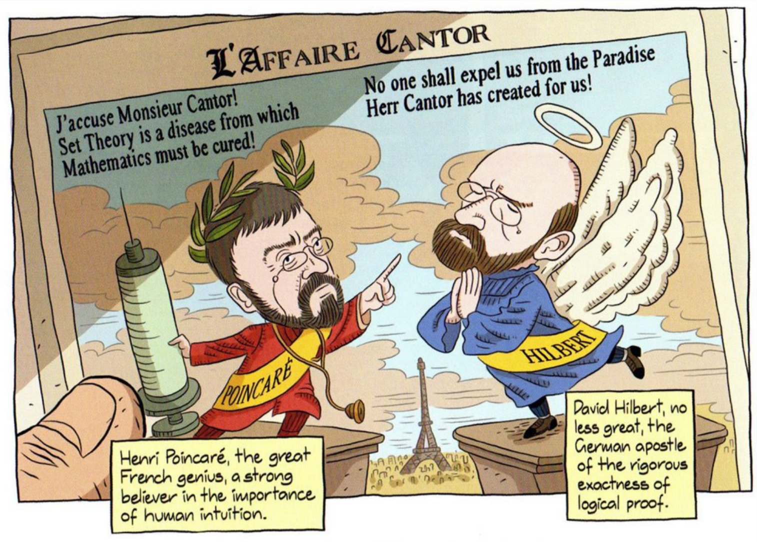
- What is mathematical and scientific thinking?
- Pattern-matching, analogy, induction from examples
- Deductive reasoning
- Complicated feedback loops between induction and deduction
- Using a lot of previous knowledge - both for induction and deduction

- 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 (Tree NNs for ATP, E prover, ...)
- 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, ...
- Buzzword bingo timeline: AI vs ML vs NNs vs DL vs LLMs vs AGI vs ...?
  See Ben Goertzel's 2018 Prague talk: `https://youtu.be/Zt2HSTuGBn8`

Intuition vs Formal Reasoning – Poincaré vs Hilbert

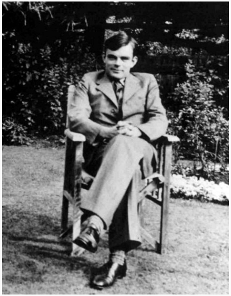


[Adapted from: *Logicomix: An Epic Search for Truth* by A. Doxiadis]

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 ...
- (formal) math as a universal/science-complete game, *semantic sweetspot*

Why Combine Learning and Reasoning 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

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
- Theoretically complete: will solve arbitrary solvable problem (AGI??)
- BUT: Combinatorial explosion, esp. 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 ...

First Order – Automated Theorem Proving (ATP)

- try to infer conjecture C from axioms Ax: $Ax \vdash C$
- most classical methods proceed by refutation: $Ax \wedge \neg C \vdash \bot$
- $Ax \wedge \neg C$ 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:
- $\neg man(X) \vee mortal(X), man(socrates) \vdash mortal(socrates)$
- **saturation-style** (resolution/superposition) provers generate inferences/clauses, looking for the contradiction (empty clause)
- **tableaux, connection** calculus – often implement backtracking (more suitable for RL/MCTS)
- **instantiation-based** – systematically add (or guess) ground instances and use SAT solvers to check satisfiability
- **combined approaches** – SAT run often inside the ATP (generalized splitting, AVATAR, iProver, SMT, etc.)

The CADE ATP System Competition (CASC)

Higher-order Theorems	Zipperpin 2.0	*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 0.5.1	E 2.5	CSE_E 1.2	iProver 3.3	GKC 0.5.1	CVC4 1.8	Zipperpin 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	111/5
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 2

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	Twee 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	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%

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 Innner Mapping loop, then Q is nilpotent of class $\leqslant 3$.
- 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, old!) – backprop, SGD, 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, often SoTA
- **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, good for *online learning* (important in ITP)
- **naive Bayes** – compute probabilities of outcomes assuming complete (naive) independence of characterizing features, i.e., just multiplying probabilities: $P(y|\mathbf{x}) = P(x_1|y) * P(x_2|y) * \ldots * P(x_n|y) * P(y)/P(\mathbf{x})$
- **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, online/incremental, 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, a bit overshadowed by DL
- **Deep Learning (DL)** – deep neural nets that automatically find important high-level features for a task, can be structured (tree/graph NNs)
- **Data Augmentation and Selection** – how do we generate/select more/better data to learn on?
- **Latent Semantics, PCA, dimensionality reduction**: use linear algebra (eigenvector decomposition) to discover the most similar features, make approximate equivalence classes from them; or just use *hashing*
- **word2vec and related/neural 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 data (formulas, proofs, models) in our mind?

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, 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, Sledgehammer
- Flyspeck (including core HOL Light and Multivariate) – since 2012
- HOL4 – since 2014, TacticToe (2017), CakeML – 2017, GRUNGE – 2019
- Coq – since 2013/2016 (CoqHammer - 2016, Tactician - 2020)
- ACL2 – 2014?
- Lean?, Stacks?, Arxiv?, ProofWiki?, ...

AITP Challenges/Bets from 2014

- 3 AITP bets for 10k EUR from my 2014 talk at Institut Henri Poincare (`tinyurl.com/yb55b3jv`)
  - 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)

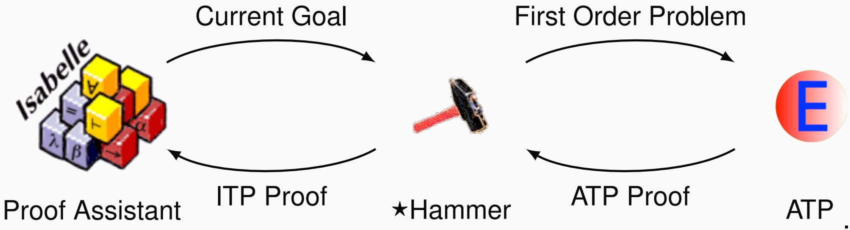
Outline

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`
- TacticToe longer: `https://www.youtube.com/watch?v=BO4Y8ynwT6Y`
- 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

Today's AI-ATP systems (⋆-Hammers)



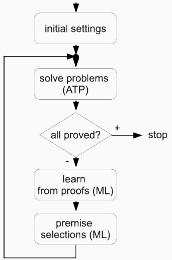
How much can it do?

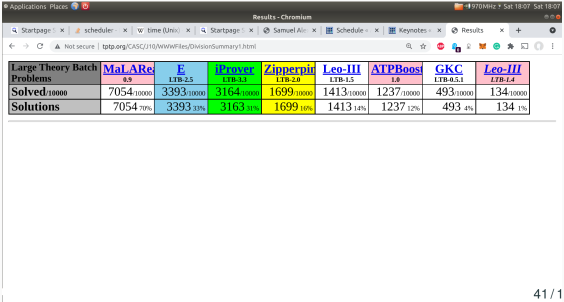
- Mizar / MML – MizAR
- Isabelle (Auth, Jinja) – Sledgehammer
- Flyspeck (including core HOL Light and Multivariate) – HOL(y)Hammer
- HOL4 (Gauthier and Kaliszyk)
- CoqHammer (Czajka and Kaliszyk) - about 40% on Coq standard library

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

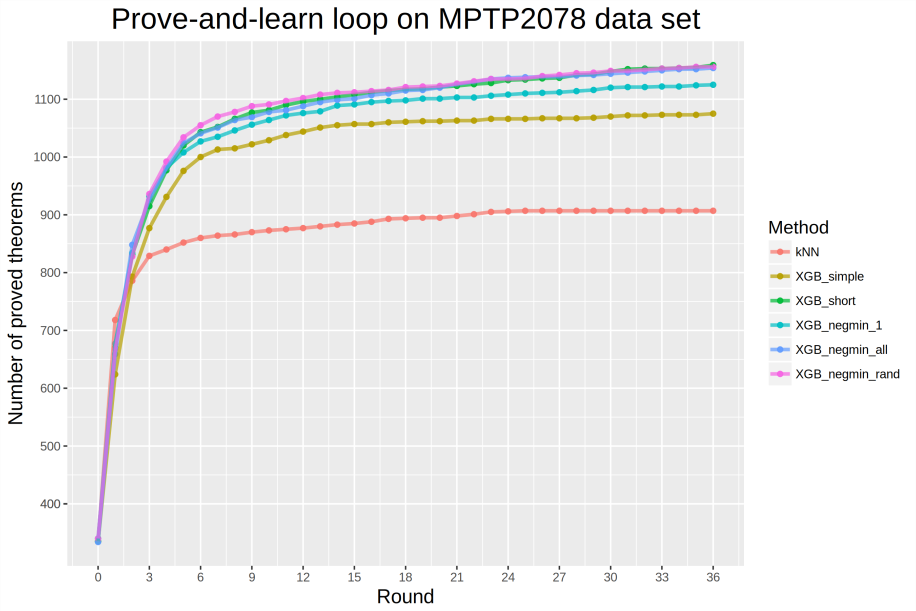
High-level feedback loops – MALARea, ATPBoost

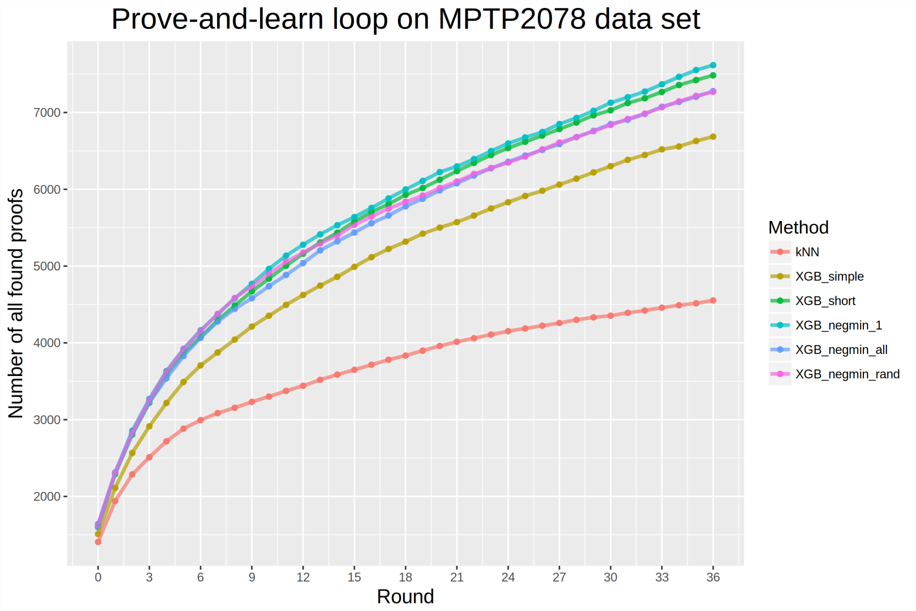
- Machine Learner for Autom. Reasoning (2006) – infinite hammering
- feedback loop interleaving ATP with learning premise selection
- both syntactic and semantic features for characterizing formulas:
- evolving set of finite (counter)models in which formulas evaluated
- winning AI/ATP benchmarks (MPTPChallenge,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

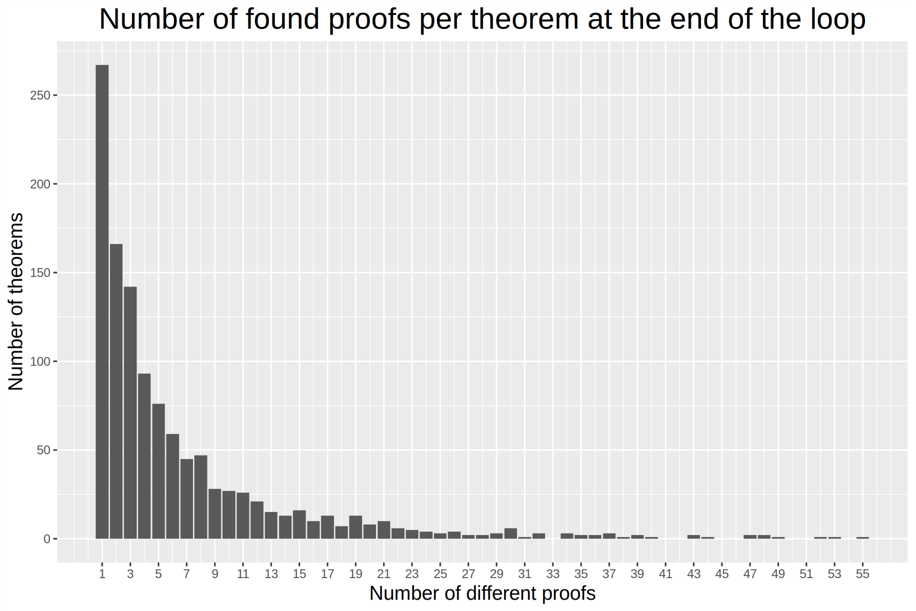




Large Theory Batch Problems	MaLARea 0.9	E LTB-2.5	iProver LTB-3.3	Zipperpin LTB-2.0	Leo-III LTB-1.5	ATPBoost 1.0	GKC LTB-0.5.1	Leo-III LTB-1.4
Solved/10000	7054/10000	3393/10000	3164/10000	1699/10000	1413/10000	1237/10000	493/10000	134/10000
Solutions	7054 70%	3393 33%	3163 31%	1699 16%	1413 14%	1237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


Prove-and-learn loop on MPTP2078 data set
Number of proved theorems
1100
1000
900
800
700
600
500
400
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


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
250
200
150
100
50
0
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
Number of different proofs

Finding shorter proofs: 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]);;
```

Finding shorter proofs: 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)
```

Various Improvements and Additions

- Model-based features for semantic similarity [IJCAR'08]
- Features encoding term matching/unification [IJCAI'15]
- Various learners: weighted k-NN, boosted trees (LightGBM,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, neural models, definitional embeddings (with Google)
- Learning in binary setting from many alternative proofs
- Negative/positive mining (ATPBoost - Piotrowski & JU, 2018)
- Stateful neural methods: RNNs and Transformers (Piotrowski & JU, 2020) (smooth transition from fact selection to conjecturing – Jakubuv & JU 2020)
- Currently strongest: Name-independent graph neural nets (Olsak, 2020) (generalize very well to new terminology/lemmas)

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:

$c_1 : P(x)$

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

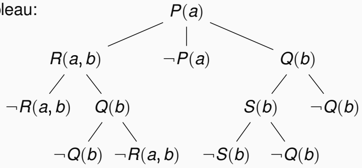
$c_3 : S(x) \vee \neg Q(b)$

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

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

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

Closed Connection Tableau:



leanCoP: Minimal Prolog 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
- Fairly Efficient MaLeCoP = **FEMaLeCoP**
- both prover and naive Bayes in OCAML, fast indexing, 40% slower
- 15% real-time improvement over leanCoP on the MPTP2078 problems
- using iterative deepening - enumerate shorter proofs before longer ones

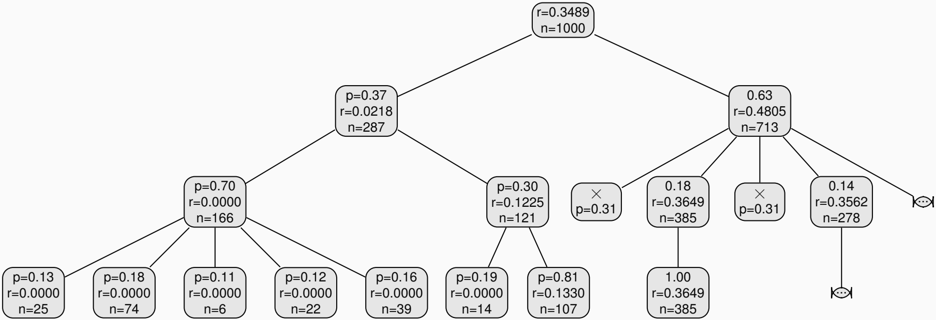
Statistical Guidance of Connection Tableau – rlCoP

- 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) (inspired by AlphaGo/Zero)
- 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
- More recently fun with GNNs (Olsak, Rawson, Zombori, ...)

Tree Example



Statistical Guidance of Connection Tableau – rlCoP

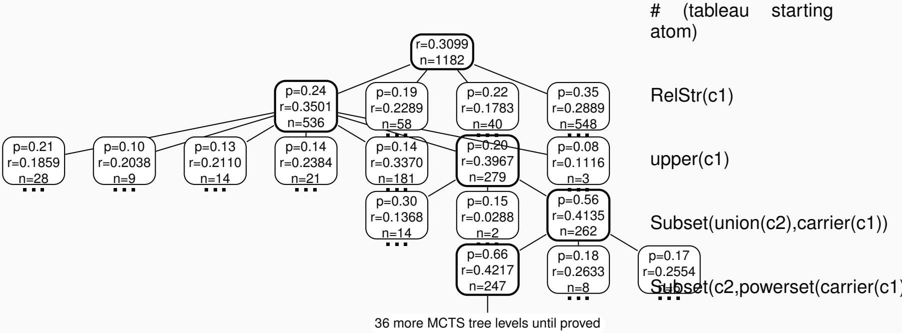
- On 32k Mizar40 problems using 200k inference limit
- nonlearning CoPs:

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



# (tableau starting atom)
RelStr(c1)
upper(c1)
Subset(union(c2),carrier(c1))
Subset(c2,powerset(carrier(c1)
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

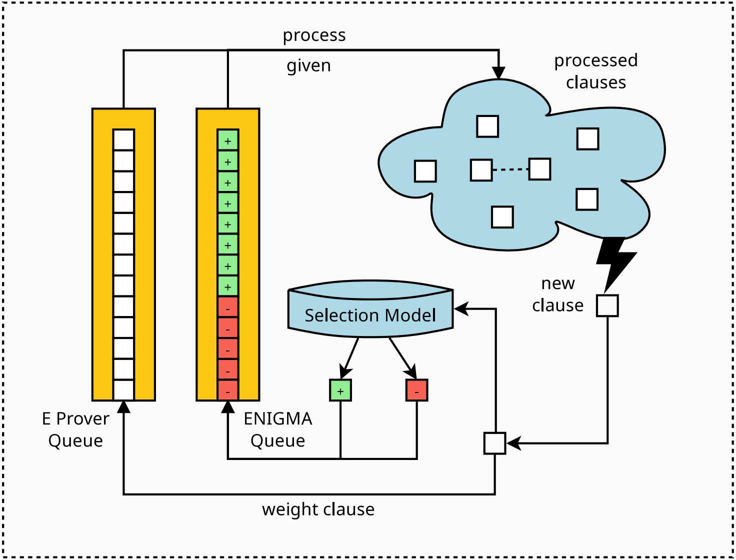
ENIGMA (2017): Guiding the Best ATPs like E Prover

Basic Saturation Loop – Given Clause Loop (E, Vampire, SPASS, Prover9, ...)

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

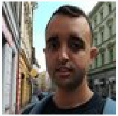
Typically, U grows quadratically wrt. P
1M clauses in U in 10s common – choosing good g gets hard – use ML!

ENIGMA: ML-based Given Clause Guidance



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

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

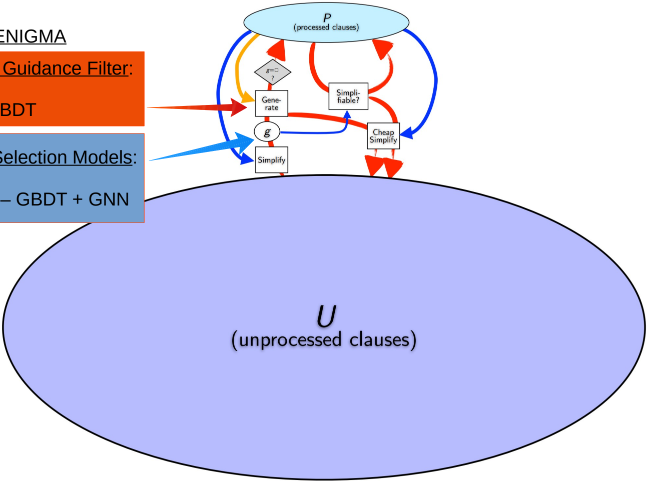
3-phase ENIGMA

Parental Guidance Filter:

Fast – GBDT

Clause Selection Models:

2-phase – GBDT + GNN



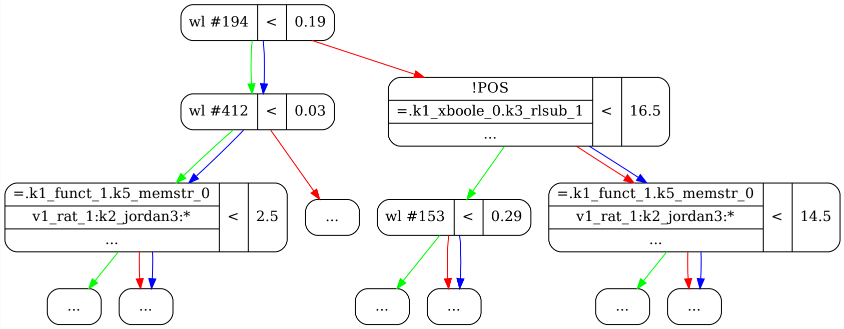
More Low-Level Guidance of Various Creatures

- 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
- 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

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 (hints) 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 in E
- 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
- Use the *proof completion ratios* as features for characterizing proof state
- Instead of just *static* conjecture features - the proof vectors evolve
- **EnigmaWatch**: Feed them to ML systems too (much more *semantic*)

Example of an XGBoost decision tree



Outline

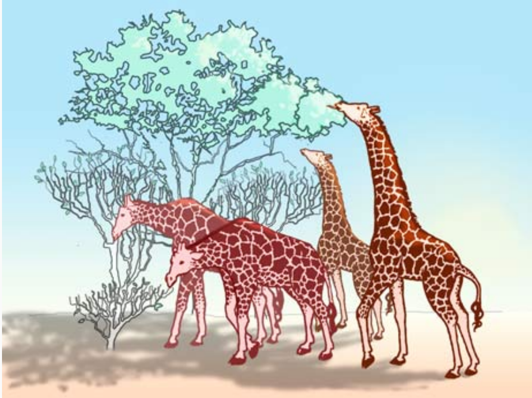
TacticToe: mid-level ITP Guidance (Gauthier'17,18)

- 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 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,
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)
```

Outline

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

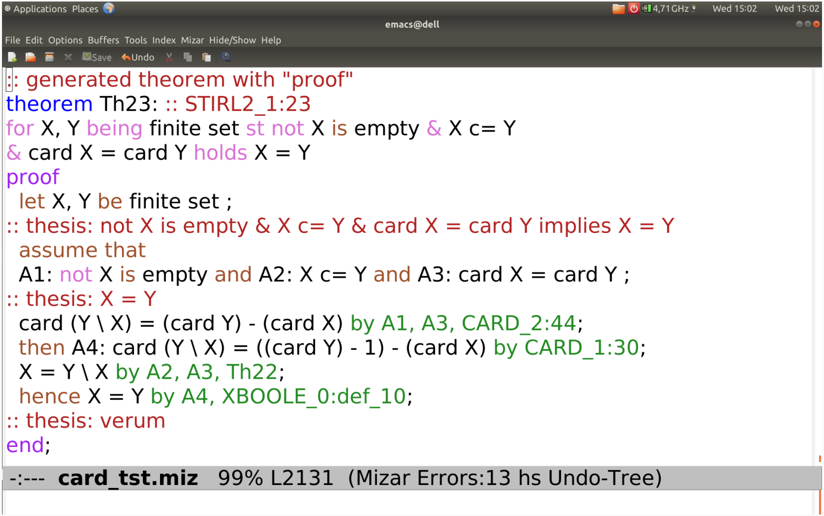
A bit of conjecturing history

- 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

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




```
:: 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;
```

Figure: Fake full declarative GPT-2 “Mizar 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.
```

QSynt: Semantics-Aware Synthesis of Math Objects

- Long AGI'24 talk on OEIS: `https://t.ly/nnwrZ`
- 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)
- 127k OEIS sequences (out of 350k) solved so far (700 iterations): `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: $\geqslant$ 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 $\ldots \leqslant 0$ 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:

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

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

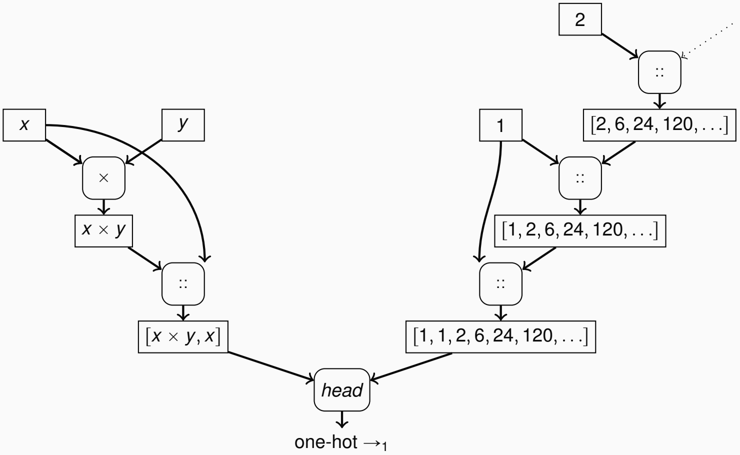
- Programs are built in reverse polish notation
- Start from an empty stack
- Use ML to repeatedly choose the next operator to push on top of a stack
- Example: Factorial is $\textit{loop}(\lambda(x, y).\ x \times y, x, 1)$, built by:

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

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

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 $loop(\lambda(x, y).\ x \times y, x, 1)$.

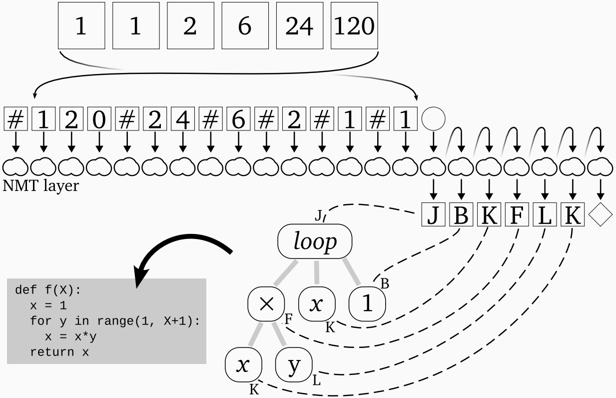


7 iterations of the tree search gradually extending the search tree. The set of the synthesized programs after the 7th iteration is $\{1, x, y, x \times y, x\ mod\ y\}$.

$$
\begin{tikzcd}
 & {[\ ]} \arrow[d, "1"] \arrow[dr, "3"] & \\
 & {[x]} \arrow[d, "2"] & {[y]} \arrow[d, "7"] \\
 & {[x, y]} \arrow[dl, "4"'] \arrow[d, "6"] & {[y, x]} \\
{[x\ mod\ y]} \arrow[d, "5"] & {[x \times y]} & \\
{[x\ mod\ y, 1]} & &
\end{tikzcd}
$$

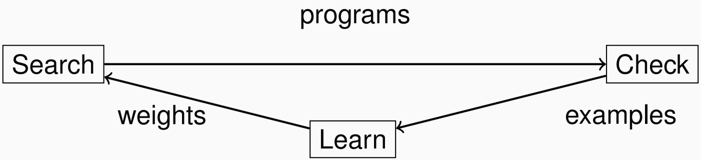
Encoding OEIS for Language Models

- 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 Feedback Loop



Analogous to our Prove/Learn feedback loops in learning-guided proving (since 2006 – MaLARea)

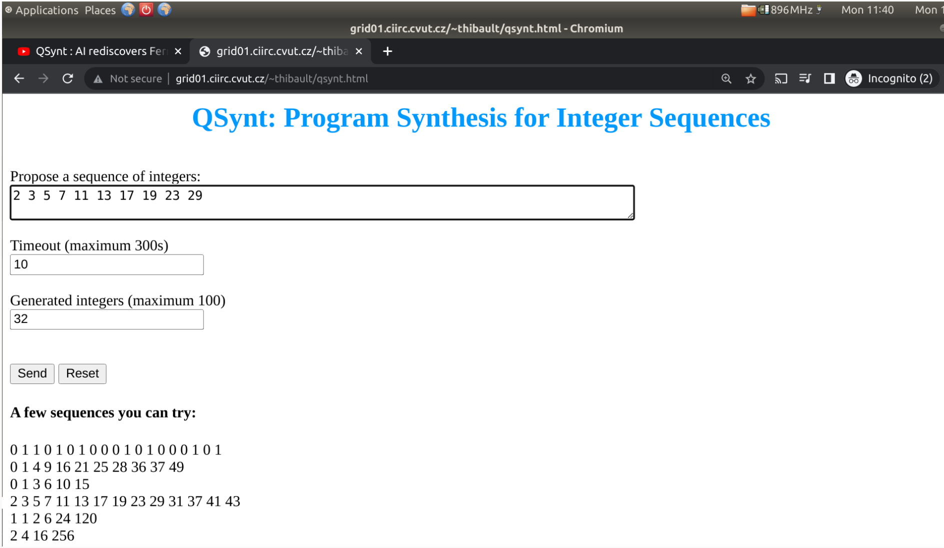
Search-Verify-Train Feedback Loop for OEIS

- **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)
- **learning phase**: LM trains to translate the “solved” OEIS sequences into the best program(s) generating them

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:
```

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:
```

Brython

Tutorial Demo Documentation Console Editor Gallery Resources

English

Brython version: 3.10.6

run Python Javascript Share code

```python
def f1(X):
  x = X
  for i in range (1,X + 1):
    x = x * i
  return x

def f0(X):
  return (f1(X) % (X + 1)) % 2

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

```
0
1
1
0
1
0
1
0
0
0
1
0
```

Five Different Self-Learning Runs

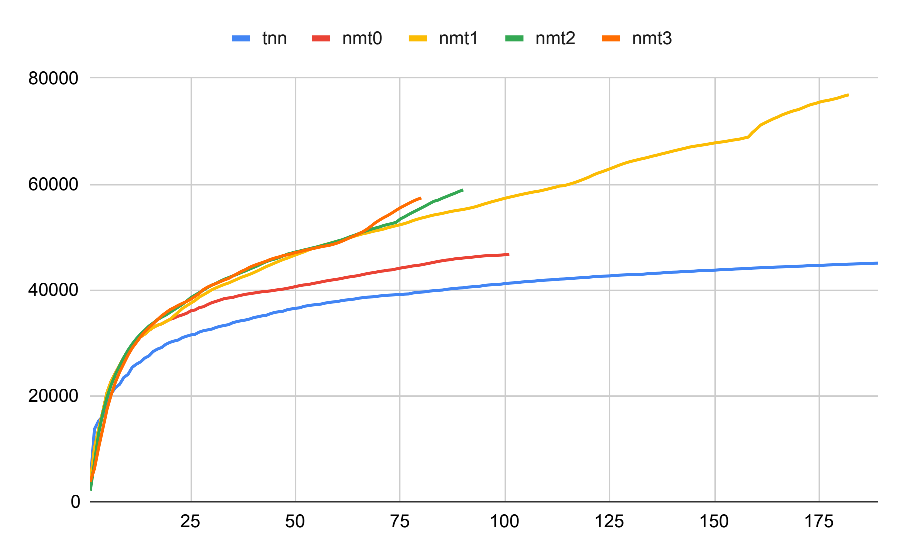


Figure: Cumulative counts of solutions.

Five Different Self-Learning Runs

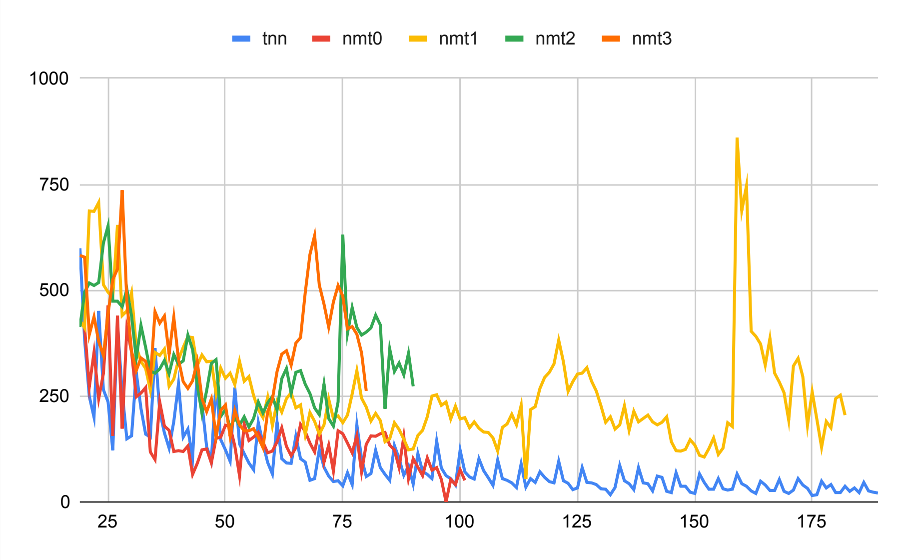


Figure: Increments of solutions.

Size Evolution

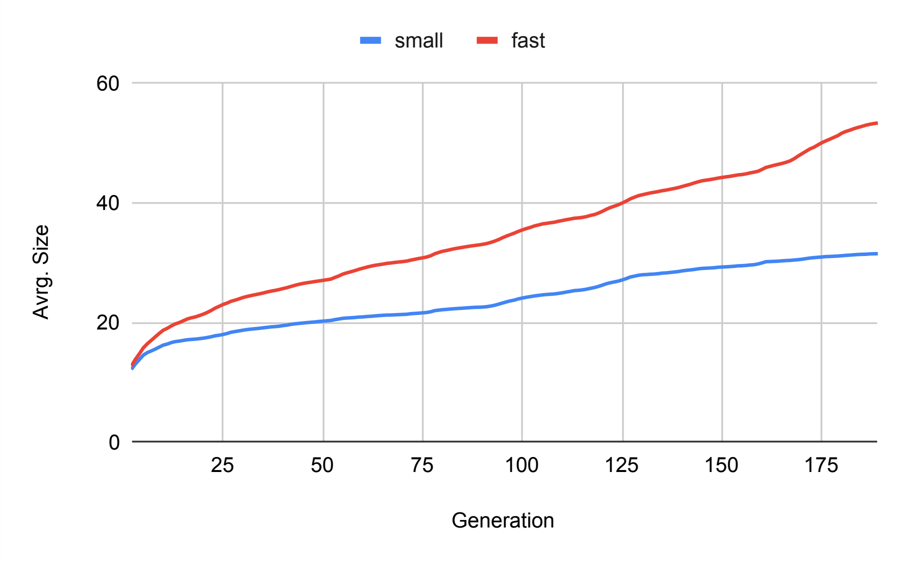


Figure: Avrg. size in iterations

Speed Evolution – Technology Breakthroughs

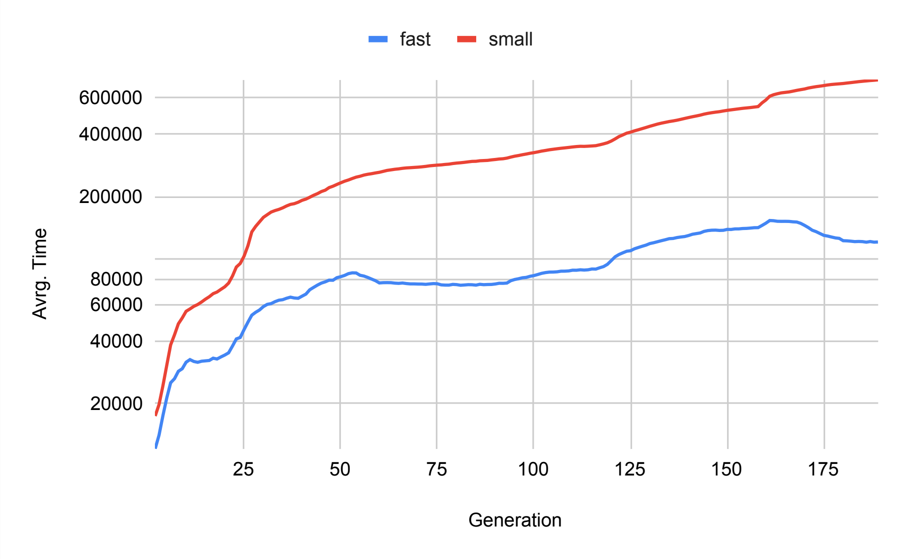
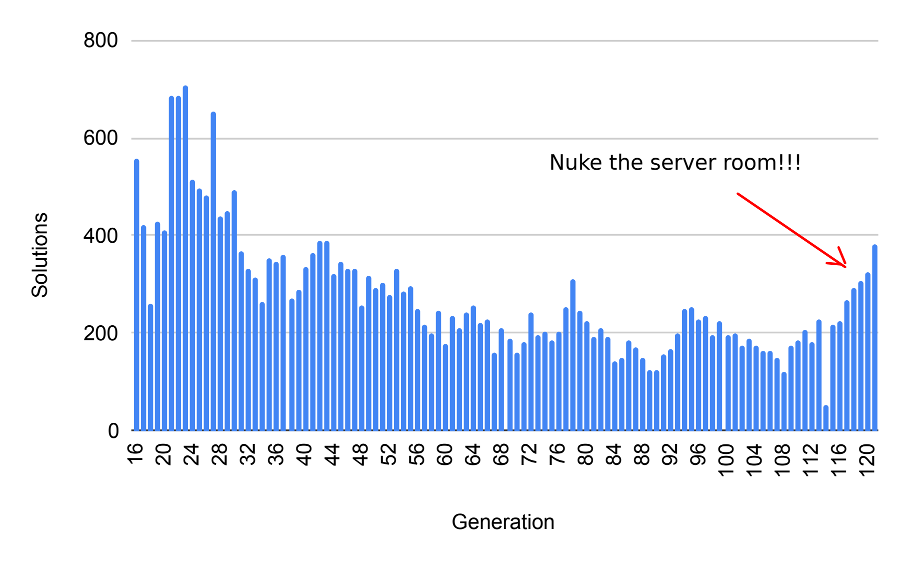
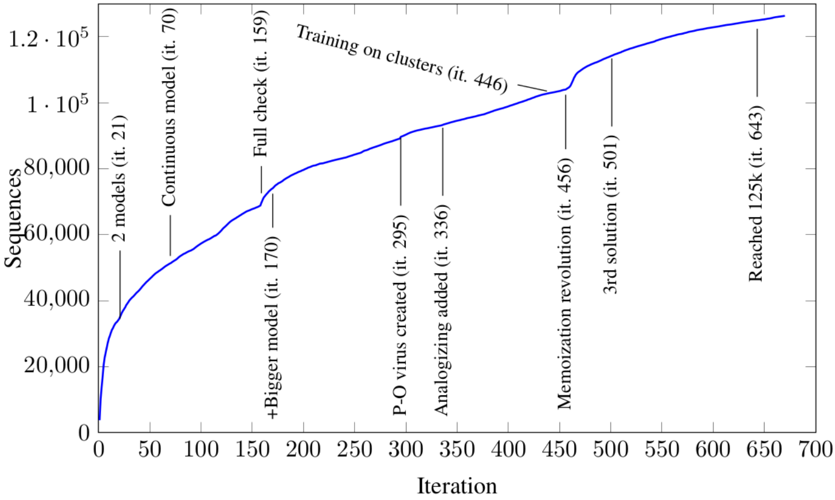


Figure: Avrg. time in iterations

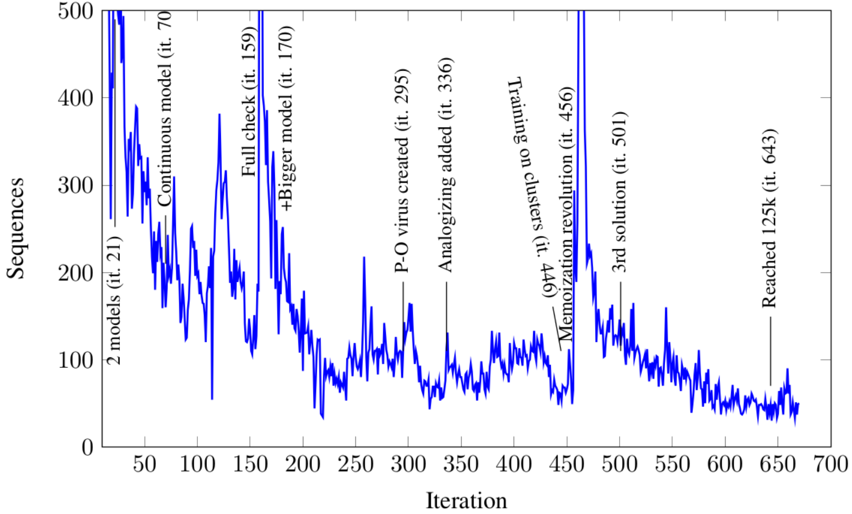
Singularity Take-Off X-mas Card



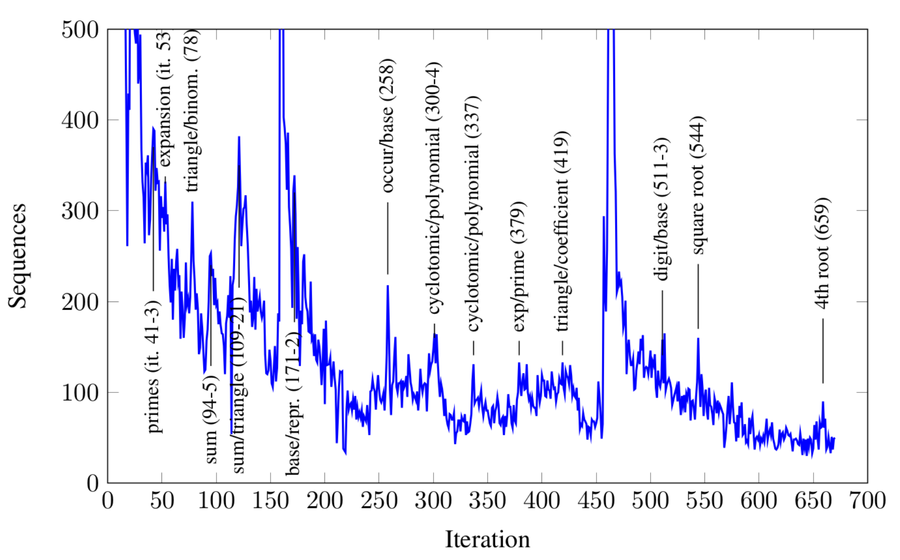
Human Made Technology Jumps



Human Made Technology Jumps



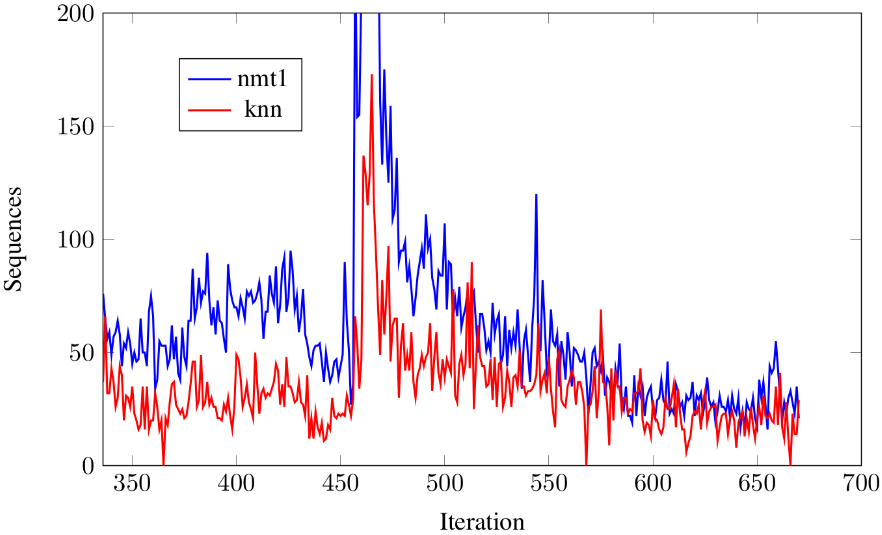
Some Automatic Technology Jumps



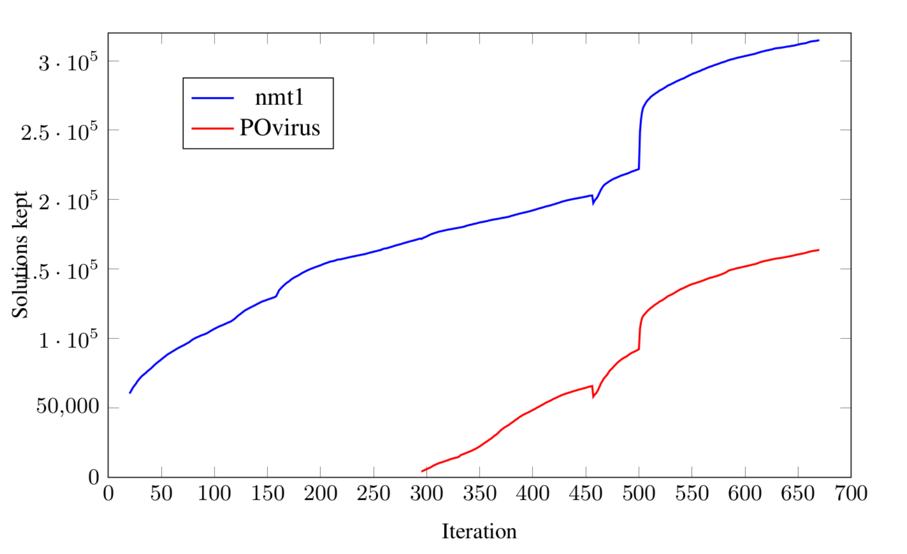
Some Automatic Technology Jumps

- iter 53: expansion/prime: A29363 Expansion of $1/((1-x^4)(1-x^7)(1-x^9)(1-x^{10}))$
- iter 78: triangle/binomial: A38313 Triangle whose (i,j)-th entry is $binomial(i,j) * 10^{i-j} * 11^j$
- iter 94-5: sum: A100192 $a(n) = Sum_{k=0..n} binomial(2n, n+k) * 2^k$
- 109-121: sum/triangle: A182013 Triangle of partial sums of Motzkin numbers
- 171-2: base/representation: A39080 n st base-9 repr. has the same number of 0's and 4's
- 258: occur/base: A44533 n st "2,0" occurs in the base 7 repr of n but not of $n+1$
- 300-304: cyclotomic/polynomial: A14620 Inverse of 611th cyclotomic polynomial
- 379: exp/prime: A124214 E.g.f.: $exp(x)/(2 - exp(3 * x))^{1/3}$
- 419: triangle/coefficient: A15129 Triangle of (Gaussian) q-binomial coefficients for $q = -13$
- 511,3: digit/base/prime: A260044 Primes with decimal digits in 0,1,3.
- 544: square root: A10538 Decimal expansion of square root of 87.
- 659: 4th root: A11084 Decimal expansion of 4th root of 93.

Translation vs Transformation



PO-virus Infection Rates



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 2M programs for the 100k sequences
- It may be quite hard to see that the programs are equivalent
- A simple example for $0, 2, 4, 6, 8, \ldots$ 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
```

80 Programs That Have Most Evolved

120	https://oeis.org/A238952	101	https://oeis.org/A97012	98	https://oeis.org/A17666
117	https://oeis.org/A35218	101	https://oeis.org/A71190	98	https://oeis.org/A113184
116	https://oeis.org/A1001	101	https://oeis.org/A70824	97	https://oeis.org/A82
112	https://oeis.org/A35178	101	https://oeis.org/A64987	97	https://oeis.org/A6579
111	https://oeis.org/A88580	101	https://oeis.org/A57660	97	https://oeis.org/A56595
111	https://oeis.org/A62069	101	https://oeis.org/A54024	97	https://oeis.org/A293228
111	https://oeis.org/A163109	101	https://oeis.org/A53222	97	https://oeis.org/A27847
111	https://oeis.org/A1615	101	https://oeis.org/A50457	97	https://oeis.org/A23645
109	https://oeis.org/A66446	101	https://oeis.org/A23888	97	https://oeis.org/A10
108	https://oeis.org/A48250	101	https://oeis.org/A209295	96	https://oeis.org/A92403
108	https://oeis.org/A321516	101	https://oeis.org/A206787	96	https://oeis.org/A90395
108	https://oeis.org/A2654	100	https://oeis.org/A99184	96	https://oeis.org/A83919
107	https://oeis.org/A75653	100	https://oeis.org/A63659	96	https://oeis.org/A7862
107	https://oeis.org/A60278	100	https://oeis.org/A62968	96	https://oeis.org/A78306
107	https://oeis.org/A23890	100	https://oeis.org/A35154	96	https://oeis.org/A69930
106	https://oeis.org/A62011	100	https://oeis.org/A339965	96	https://oeis.org/A69192
106	https://oeis.org/A346613	100	https://oeis.org/A277791	96	https://oeis.org/A54519
106	https://oeis.org/A344465	100	https://oeis.org/A230593	96	https://oeis.org/A53158
105	https://oeis.org/A49820	100	https://oeis.org/A182627	96	https://oeis.org/A351267
104	https://oeis.org/A55155	99	https://oeis.org/A9191	96	https://oeis.org/A334136
104	https://oeis.org/A349215	99	https://oeis.org/A82051	96	https://oeis.org/A33272
104	https://oeis.org/A143348	99	https://oeis.org/A62354	96	https://oeis.org/A325939
103	https://oeis.org/A92517	99	https://oeis.org/A247146	96	https://oeis.org/A211779
103	https://oeis.org/A64840	99	https://oeis.org/A211261	96	https://oeis.org/A186099
102	https://oeis.org/A9194	99	https://oeis.org/A147588	96	https://oeis.org/A143152
102	https://oeis.org/A51953	98	https://oeis.org/A318446	96	https://oeis.org/A125168
102	https://oeis.org/A155085	98	https://oeis.org/A203		

Evolution and Proliferation of Primes and Others

`https://bit.ly/3XHZsjK`: triangle coding, sigma (sum of divisors), primes. `https://bit.ly/3iJ4oGd` (the first 24, now 50)

Nr	Program
P1	(if x <= 0 then 2 else 1) + (compr (((loop (x + x) (x mod 2) (loop (x * x) 1 (loop (x + x) (x div 2) 1))) + x) mod (1 + x)) x)
P2	1 + (compr ((((loop (x * x) 1 (loop (x + x) (x div 2) 1)) + x) * x) mod (1 + x)) (1 + x))
P3	1 + (compr (((loop (x * x) 1 (loop (x + x) (x div 2) 1)) + x) mod (1 + x)) (1 + x))
P4	2 + (compr ((loop2 (1 + (if (x mod (1 + y)) <= 0 then 0 else x)) (y - 1) x 1 x) mod (1 + x)) x)
P5	1 + (compr ((loop (if (x mod (1 + y)) <= 0 then (1 + y) else x) x (1 + x)) mod (1 + x)) (1 + x))
P6	1 + (compr ((loop (if (x mod (1 + y)) <= 0 then (1 + y) else x) (2 + (x div (2 + (2 + 2)))) (1 + x)) mod (1 + x)) (1 + x))
P7	compr ((1 + (loop (if (x mod (1 + y)) <= 0 then (1 + y) else x) x x)) mod (1 + x)) (2 + x)
P8	1 + (compr ((loop (if (x mod (1 + y)) <= 0 then (1 + y) else x) (1 + ((2 + x) div (2 + (2 + 2)))) (1 + x)) mod (1 + x)) (1 + x))
P9	compr (x - (loop (if (x mod (1 + y)) <= 0 then (1 + y) else x) x x)) (2 + x)
P10	compr (x - (loop (if (x mod (1 + y)) <= 0 then 2 else x) (x div 2) x)) (2 + x)
P11	1 + (compr ((loop (if (x mod (1 + y)) <= 0 then (1 + y) else x) (1 + (x div (2 + (2 + 2)))) (1 + x)) mod (1 + x)) (1 + x))
P12	compr ((x - (loop (if (x mod (1 + y)) <= 0 then y else x) x x)) - 2) (2 + x)
P13	1 + (compr ((loop (if (x mod (1 + y)) <= 0 then (1 + y) else x) (2 + (x div (2 * (2 + (2 + 2))))) (1 + x)) mod (1 + x)) (1 + x))
P14	compr ((x - (loop (if (x mod (1 + y)) <= 0 then y else x) x x)) - 1) (2 + x)
P15	1 + (compr (x - (loop (if (x mod (1 + y)) <= 0 then (1 + y) else x) (2 + (x div (2 * (2 + (2 + 2))))) (1 + x))) (1 + x))
P16	compr (2 - (loop (if (x mod (1 + y)) <= 0 then 0 else x) (x - 2) x)) x
P17	1 + (compr (x - (loop (if (x mod (1 + y)) <= 0 then 2 else x) (2 + (x div (2 * (2 + (2 + 2))))) (1 + x))) (1 + x))
P18	1 + (compr (x - (loop (if (x mod (1 + y)) <= 0 then 2 else x) (1 + (2 + (x div (2 * (2 * (2 + 2)))))) (1 + x))) (1 + x))
P19	1 + (compr (x - (loop2 (loop (if (x mod (1 + y)) <= 0 then 2 else x) (2 + (y div (2 * (2 + (2 + 2))))) (1 + y)) 0 (1 - (x mod 2)) 1 x)) (1 + x))
P20	1 + (compr (x - (loop2 (loop (if (x mod (1 + y)) <= 0 then 2 else x) (1 + (2 + (y div (2 * (2 * (2 + 2)))))) (1 + y)) 0 (1 - (x mod 2)) 1 x)) (1 + x))
P21	1 + (compr (x - (loop2 (loop (if (x mod (2 + y)) <= 0 then 2 else x) (2 + (y div (2 * ((2 + 2) + (2 + 2))))) (1 + y)) 0 (1 - (x mod 2)) 1 x)) (1 + x))
P22	1 + (compr (x - (loop2 (loop (if (x mod (2 + y)) <= 0 then 2 else x) (2 + (y div (2 * (2 * (2 + 2))))) (1 + y)) 0 (1 - (x mod 2)) 1 x)) (1 + x))
P23	2 + (compr (loop (x - (if (x mod (1 + y)) <= 0 then 0 else 1)) x x) x)
P24	loop (1 + x) (1 - x) (1 + (2 * (compr (x - (loop (if (x mod (2 + y)) <= 0 then 1 else x) (2 + (x div (2 * (2 + 2)))) (1 + (x + x)))) x)))

Evolution and Proliferation of Primes

Iter	P1	P2	P3	P4	P5	P6	P7	P8	P9	P10	P11	P12	P13	P14	P15	P16	P17	P18	P19	P20	P21	P22	P23	P24
25	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
26	6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
27	7	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
28	8	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
29	9	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
30	10	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
31	4	6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
32	6	6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
33	8	1	6	6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
34	12	4	6	6	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
35	7	12	6	0	9	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
36	4	10	6	0	17	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
37	3	4	6	0	18	6	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
38	2	3	1	0	12	18	11	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
39	2	3	1	0	9	56	31	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
40	2	5	2	0	7	59	49	9	1	0	2	0	1	0	0	0	0	0	0	0	0	0	0	0
41	1	2	3	0	4	52	58	42	23	0	13	0	8	0	0	0	0	0	0	0	0	0	0	0
42	0	2	4	0	3	44	50	38	60	8	11	0	55	0	0	0	0	0	0	0	0	0	0	0
43	0	2	12	0	0	37	55	14	116	35	16	7	90	0	0	0	0	0	0	0	0	0	0	0
44	0	2	13	0	0	28	40	6	176	73	19	8	122	9	12	0	0	0	0	0	0	0	0	0
45	0	2	9	0	0	19	24	4	147	185	26	16	94	25	29	0	7	0	0	0	0	0	0	0
46	0	2	4	0	0	11	14	0	101	256	21	14	66	64	30	0	29	0	0	0	0	0	0	0
47	0	0	0	0	0	9	4	0	55	290	23	3	43	116	16	6	62	14	0	0	0	0	0	0
48	0	0	0	0	0	8	0	0	22	261	16	0	34	192	10	6	89	30	0	0	0	0	0	0
49	0	0	0	0	0	8	0	0	6	195	11	0	36	225	8	6	99	34	0	0	0	0	0	0
50	0	0	0	0	0	5	0	0	2	154	8	0	29	168	6	6	108	39	0	0	0	0	0	0
51	0	0	0	0	0	4	0	0	0	121	7	0	21	97	6	6	113	43	0	0	0	0	0	0
52	0	0	0	0	0	2	0	0	0	118	8	0	12	62	6	6	110	51	0	0	0	0	0	0
53	0	0	0	0	0	1	0	0	0	59	7	0	15	33	6	6	125	62	0	0	0	0	0	0
54	0	0	0	0	0	1	0	0	0	41	4	0	16	17	6	9	137	72	0	0	0	0	0	0
55	0	0	0	0	0	2	0	0	0	32	4	0	15	9	6	17	147	82	0	0	0	0	0	0
56	0	0	0	0	0	1	0	0	0	29	4	0	10	7	6	39	152	98	0	0	0	0	0	0

Selection of 123 Solved Sequences

https://github.com/Anon52MI4/oeis-alien

Table: Samples of the solved sequences.

https://oeis.org/A317485	Number of Hamiltonian paths in the n-Bruhat graph.
https://oeis.org/A349073	a(n) = U(2*n, n), where U(n, x) is the Chebyshev polynomial of the second kind.
https://oeis.org/A293339	Greatest integer k such that $k/2^n < 1/e$.
https://oeis.org/A1848	Crystal ball sequence for 6-dimensional cubic lattice.
https://oeis.org/A8628	Molien series for A_5.
https://oeis.org/A259445	Multiplicative with $a(n) = n$ if n is odd and $a(2^s) = 2$.
https://oeis.org/A314106	Coordination sequence Gal.6.199.4 where G.u.t.v denotes the coordination sequence for a vertex of type v in tiling number t in the Galebach list of u-uniform tilings
https://oeis.org/A311889	Coordination sequence Gal.6.129.2 where G.u.t.v denotes the coordination sequence for a vertex of type v in tiling number t in the Galebach list of u-uniform tilings.
https://oeis.org/A315334	Coordination sequence Gal.6.623.2 where G.u.t.v denotes the coordination sequence for a vertex of type v in tiling number t in the Galebach list of u-uniform tilings.
https://oeis.org/A315742	Coordination sequence Gal.5.302.5 where G.u.t.v denotes the coordination sequence for a vertex of type v in tiling number t in the Galebach list of u-uniform tilings.
https://oeis.org/A004165	OEIS writing backward
https://oeis.org/A83186	Sum of first n primes whose indices are primes.
https://oeis.org/A88176	Primes such that the previous two primes are a twin prime pair.
https://oeis.org/A96282	Sums of successive twin primes of order 2.
https://oeis.org/A53176	Primes p such that $2p + 1$ is composite.
https://oeis.org/A267262	Total number of OFF (white) cells after n iterations of the "Rule 111" elementary cellular automaton starting with a single ON (black) cell.

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 all 2018) – no need for aligned data, improving a lot!
- Type-checking not yet internal (boosting well-typed data externally)

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 $\lim(s_8+s_7) = \lim s_8 + \lim s_7$
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 ) ;`

Unsupervised NMT Fun on Short Formulas

```
len <* a *> = 1 ;                   len <* a *> = 1 ;
assume i < len q ;                  i < len q ;
len <* q *> = 1 ;                   len <* q *> = 1 ;
s = apply ( v2 , v1 ast t ) ;       s = apply ( v2 , v1 ) . t ;
s . ( i + 1 ) = tt . ( i + 1 )      s . ( i + 1 ) = tau1 . ( i + 1 )
1 + j <= len v2 ;                   1 + j <= len v2 ;
1 + j + 0 <= len v2 + 1 ;           1 + j + 0 <= len v2 + 1 ;
let i be Nat ;                      i is_at_least_length_of p ;
assume v is_applicable_to t ;       not v is applicable ;
let t be type of T ;                t is_orientedpath_of v1 , v2 , T ;
a ast t in downarrow t ;            a *' in downarrow t ;
t9 in types a ;                     t '2 in types a ;
a ast t <= t ;                      a *' <= t ;
A is_applicable_to t ;              A is applicable ;
Carrier ( f ) c= B                  support ppf n c= B
u in B or u in { v } ;              u in B or u in { v } ;
F . w in w & F . w in I ;           F . w in F & F . w in I ;
GG . y in rng HH ;                  G0 . y in rng ( H1 ./. y ) ;
a * L = Z_ZeroLC ( V ) ;            a * L = ZeroLC ( V ) ;
not u in { v } ;                    u >> v ;
u <> v ;                            u <> v ;
v - w = v1 - w1 ;                   vw = v1 - w1 ;
v + w = v1 + w1 ;                   v + w = v1 + w1 ;
x in A & y in A ;                   assume [ x , y ] in A ;
```

Acknowledgments

- Prague Automated Reasoning Group `http://arg.ciirc.cvut.cz/`:
  - Jan Jakubuv, Martin Suda, Mikolas Janota, Chad Brown, Karel Chvalovsky, Bob Veroff, Zar Goertzel, Bartosz Piotrowski, Lasse Blaauwbroek, Jelle Piepenbrock, 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 References

- Lasse Blaauwbroek, David M. Cerna, Thibault Gauthier, Jan Jakubuv, Cezary Kaliszyk, Martin Suda, Josef Urban: Learning Guided Automated Reasoning: A Brief Survey. Logics and Type Systems in Theory and Practice 2024: 54-83
- J. Urban: AI4REASON ERC project's final report. http://ai4reason.org/PR_CORE_SCIENTIFIC_4.pdf
- Zar Goerzel's PhD thesis (nice intro/overview): Learning Inference Guidance in Automated Theorem Proving. https://dspace.cvut.cz/bitstream/handle/10467/111606/F3-D-2023-Goertzel-Zarathustra-AITP_Doctoral_Thesis_ZAG.pdf
- Jan Jakubuv, Karel Chvalovský, Zarathustra Amadeus Goertzel, Cezary Kaliszyk, Mirek Olsák, Bartosz Piotrowski, Stephan Schulz, Martin Suda, Josef Urban: MizAR 60 for Mizar 50. ITP 2023: 19:1-19:22
- Karel Chvalovský, Konstantin Korovin, Jelle Piepenbrock, Josef Urban: Guiding an Instantiation Prover with Graph Neural Networks. LPAR 2023: 112-123
- Thibault Gauthier, Miroslav Olsák, Josef Urban: Alien coding. Int. J. Approx. Reason. 162: 109009 (2023).
- Thibault Gauthier, Josef Urban: Learning Program Synthesis for Integer Sequences from Scratch. AAAI 2023: 7670-7677
- Thibault Gauthier, Chad E. Brown, Mikolas Janota, Josef Urban: A Mathematical Benchmark for Inductive Theorem Provers. LPAR 2023: 224-237
- Lasse Blaauwbroek, Mirek Olsák, Jason Rute, Fidel Ivan Schaposnik Massolo, Jelle Piepenbrock, Vasily Pestun: Graph2Tac: Online Representation Learning of Formal Math Concepts. ICML 2024

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 Csiszárik, 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 (Extende 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!
- To push AI methods in math and theorem proving, we organize:
- AITP – Artificial Intelligence and Theorem Proving
- September 2025, Aussois, France, `aitp-conference.org`
- ATP/ITP/Math vs AI/ML/AGI people, Computational linguists
- Discussion-oriented and experimental