

ADVENTURES IN LEARNING PROVING, INSTANTIATION AND SYNTHESIS

Josef Urban

Czech Technical University in Prague

The University of Melbourne
December 6, 2024



A Match Made in Heaven or a Deal with the Devil?

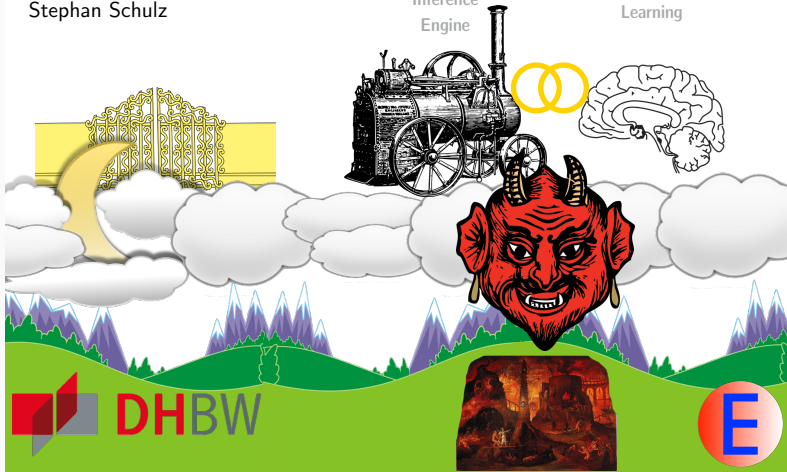
Deduction and Induction

A Match Made in Heaven or a Deal with the Devil?

Stephan Schulz

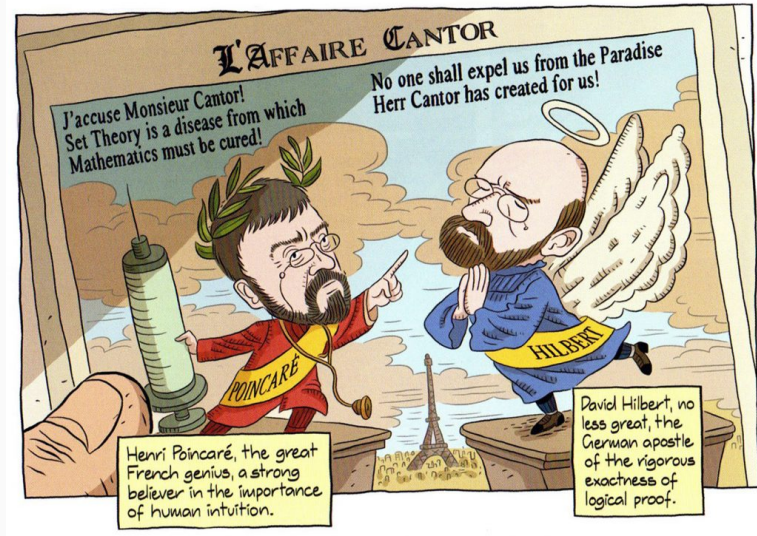
The
Inference
Engine

Machine
Learning



[Stephan Schulz's talk at AITP'16]

Intuition vs Formal Reasoning – Poincaré vs Hilbert



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

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

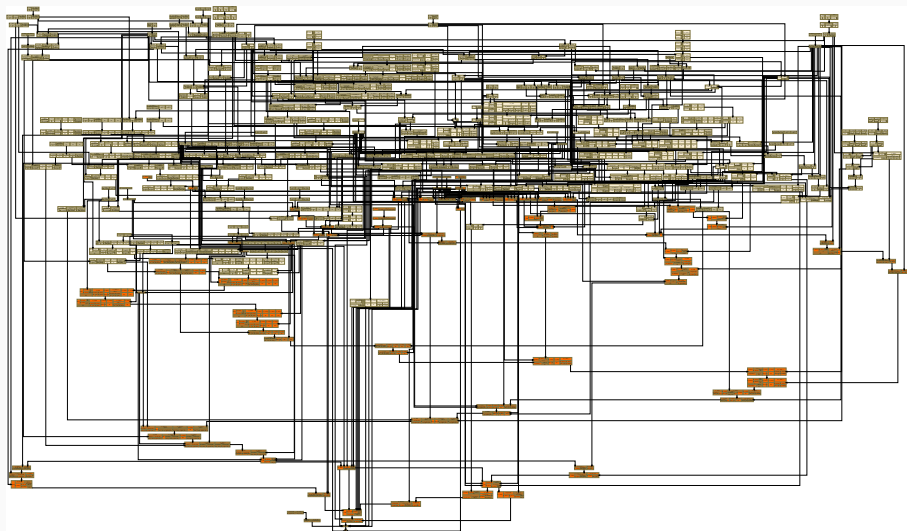
- Done on 57880 Mizar Mathematical Library formal math problems in 2019
- Efficient **ML-guidance inside the best ATPs** like E prover (ENIGMA)
- *Training* of the ML-guidance is *interleaved* with *proving* harder problems
- Ultimately a **70% improvement** over the original E strategy:
- ... from 14933 proofs to 25397 proofs (all in 10s CPU - no cheating)

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

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

- **75% of the Mizar corpus** (43414) reached in July 2021 - higher times and many prove/learn cycles: https://github.com/ai4reason/ATP_Proofs
- Details in our Mizar60 paper: <https://arxiv.org/abs/2303.06686>

Can you do this in 4 minutes? (359-step ATP proof)



Can you do this in 4 minutes? (human-written code)

```
theorem 7h31: BORSUK 5:31
  For A being Subset of  $\mathbb{R}^1$ 
  for a, b being real number st a < b & A = RAT (a,b) holds
  Cl A = [.a,b.]
proof
  let A be Subset of  $\mathbb{R}^1$ ; :: 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  $\mathbb{R}^1$  by NUMBERS:12, TOPMETR:17;
  reconsider RR = RAT /\ [.a,b.] as Subset of  $\mathbb{R}^1$  by TOPMETR:17;
  A3: the carrier of  $\mathbb{R}^1 \setminus \text{Cl } ab$  = Cl ab by XREAL_1:20;
  A4: Cl RR c= (Cl RT) /\ (Cl ab) by XREAL_1:22;
  thus Cl A c= [.a,b.] :: according to XREAL_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  $\mathbb{R}^1 \setminus \text{Cl } ab$  by A3;
  hence x in [.a,b.] by A1, A3, A4; :: 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: p <= p by A5, XREAL_1:1;
  A7: p <= b by A5, XREAL_1:1;
  per cases by A7, XREAL_0:1;
  suppose AB: 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 XREAL_0:17;
  assume A10: r > 0 ; :: thesis:
  p < min (pp,((p + b) / 2))
  proof
    per cases by XREAL_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 AB, XREAL_1:29; :: 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 AB, XREAL_1:29;
  then min (pp,((p + b) / 2)) < b by A9, XREAL_0:2;
  then A13: Q < b by A12, XREAL_0:2;
  min (pp,((p + b) / 2)) <= pp by XREAL_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, XREAL_0:2;
  then dist (p,P) < r by A11, A14;
  then A15: P in Ball (p,r) by METRIC_1:11;
  a < Q by A6, A11, XREAL_0:2;
  then A16: Q in [.a,b.] by A13, XREAL_1:4;
  Q in RAT by RAT_1:def 2;
  then Q in A by A2, A16, XREAL_0:def 4;
  hence Ball (p,r) meets A by A15, XREAL_0:13; :: thesis:
end;
hence x in Cl A by GOBOARD6:92, TOPMETR:62 ; :: thesis:
```

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

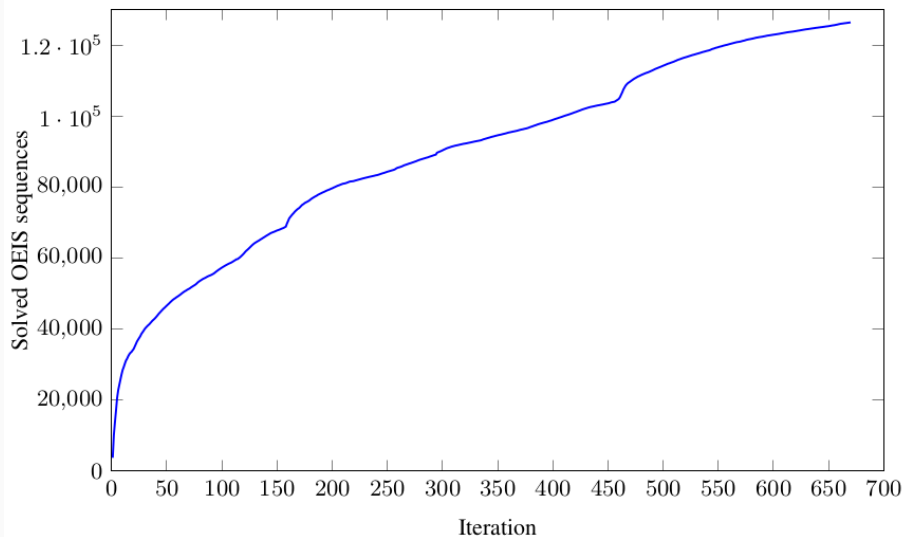
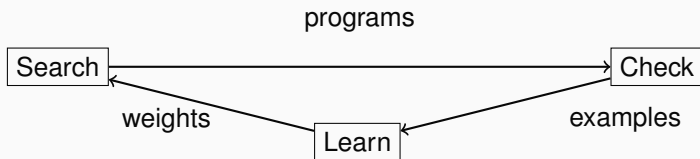


Figure 12: Number y of solved OEIS sequences after x iterations

Search-Verify-Train Positive Feedback Loop (OEIS)

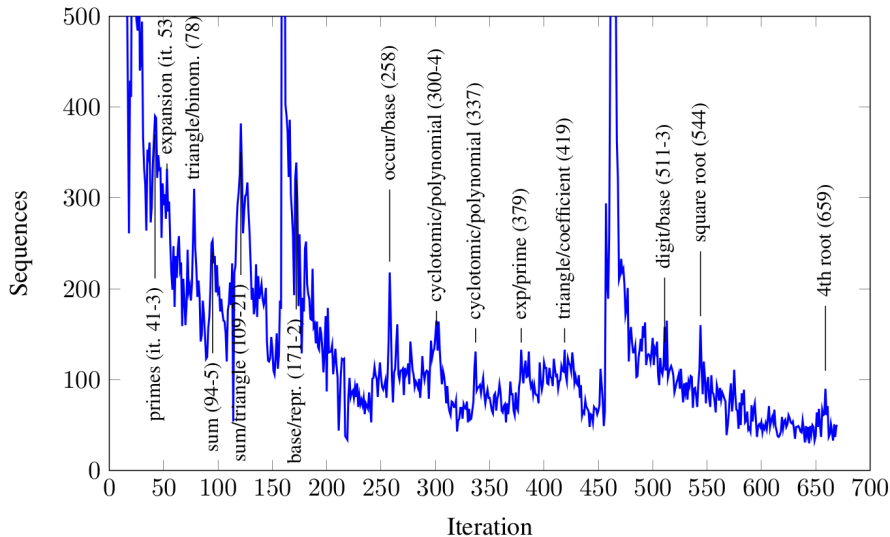


- Small Turing-complete DSL for our programs, e.g.:
 $2^x = \prod_{y=1}^x 2 = \text{loop}(2 \times x, \mathbf{x}, 1)$
 $x! = \prod_{y=1}^x y = \text{loop}(y \times x, \mathbf{x}, 1)$
- **Analogous** to our Prove/Learn feedback loops in learning-guided proving (since 2006 – **Machine Learner for Automated Reasoning** – MaLAREa))
- However, OEIS allows much faster feedback on *symbolic conjecturing*
- **670 iterations and still refuses to plateau** - counters RL wisdom?
- Since it **interleaves symbolic breakthroughs and statistical learning**?
- Cheap: The electricity bill is only \$1k-\$3k, you can do this at home
- ~4.5M explanations invented: **50+ different characterizations of primes**

Some Invented Explanations for OEIS (“Alien Coder”)

- <https://oeis.org/A4578>: Expansion of $\sqrt{8}$ in base 3:
loop2(((y * y) div (x + y)) + y, y, x + x, 2, loop((1 + 2) * x, x, 2)) mod (1 + 2)
- <https://oeis.org/A4001>: Hofstadter-Conway \$10k seq: $a(n) = a(a(n-1)) + a(n-a(n-1))$ with $a(1) = a(2) = 1$:
loop(push(loop(pop(x), y-x, pop(x)), x) + loop(pop(x), x-1, x), x - 1, 1)
- <https://oeis.org/A40>: prime numbers:
2 + compr((loop(x * y, x, 2) + x) mod (2 + x), x)
- <https://oeis.org/A30184>: Expand $\eta(q) * \eta(q^3) * \eta(q^5) * \eta(q^{15})$ in powers of q (elliptic curves):
loop(push(loop((pop(x) * loop(if (pop(x) mod y) <= 0 then (x - loop(if (x mod (1 + (y + y))) <= 0 then (x + x) else x, 2, y)) else x, y, push(0, y))) + x, y, push(0, x)), x) div y, x, 1)
- <https://oeis.org/A51023>: Wolfram's \$30k Rule 30 automaton:
loop2(y, y div 2, x, 1, loop2(loop2(((y div (0 - (2 + 2))) mod 2) + x) + x, y div 2, y, 1, loop2(((y mod 2) + x) + x, y div 2, y, 1, x)), 2 + y, x, 0, 1)) mod 2

Some Automatic Technology Jumps



Outline

Quick Intro

Motivation, Learning vs. Reasoning

Learning of Theorem Proving - Overview

Demos

High-level Reasoning Guidance: Premise Selection

Low Level Guidance of Theorem Provers

Mid-level Reasoning Guidance

Synthesis

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*

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 corpora: Mizar, Isabelle, HOL, Coq, Lean ... 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>

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 et al – 20k lemmas)
- Formal Proof of the Feit-Thompson Theorem (2012 – Gonthier et al)
- 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?
- As of 2022/23: Overlaps/analogies/differences with LLMs?

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: $P(y|\mathbf{x}) = P(x_1|y) * P(x_2|y) * \dots * P(x_n|y) * P(y)/P(\mathbf{x})$
- **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

Quick Intro

Motivation, Learning vs. Reasoning

Learning of Theorem Proving - Overview

Demos

High-level Reasoning Guidance: Premise Selection

Low Level Guidance of Theorem Provers

Mid-level Reasoning Guidance

Synthesis

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
- **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
- ...

Outline

Quick Intro

Motivation, Learning vs. Reasoning

Learning of Theorem Proving - Overview

Demos

High-level Reasoning Guidance: Premise Selection

Low Level Guidance of Theorem Provers

Mid-level Reasoning Guidance

Synthesis

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 $\epsilon_0 = \omega^{\omega^{\omega^{\dots}}}$** <https://bit.ly/3Ne4WNX>
- **Hammering demo**: <http://grid01.ciirc.cvut.cz/~mptp/out4.ogv>
- **TacticToe on HOL4**:
http://grid01.ciirc.cvut.cz/~mptp/tactictoe_demo.ogv
- **TacticToe longer**: <https://www.youtube.com/watch?v=BO4Y8ynwT6Y>
- **Tactician for Coq**:
<https://blaaubroek.eu/papers/cicm2020/demo.mp4>,
<https://coq-tactician.github.io/demo.html>
- **Inf2formal over HOL Light**:
<http://grid01.ciirc.cvut.cz/~mptp/demo.ogv>
- **QSynt: AI rediscovers the Fermat primality test**:
<https://www.youtube.com/watch?v=24oejR9wsXs>

Outline

Quick Intro

Motivation, Learning vs. Reasoning

Learning of Theorem Proving - Overview

Demos

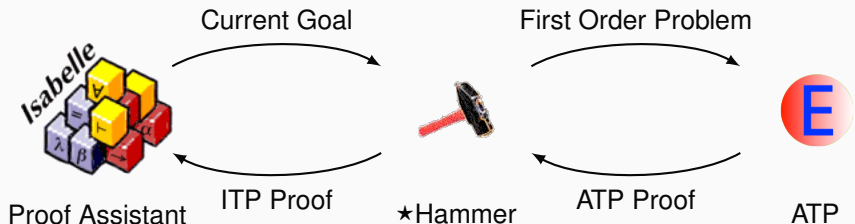
High-level Reasoning Guidance: Premise Selection

Low Level Guidance of Theorem Provers

Mid-level Reasoning Guidance

Synthesis

Today's AI-ATP systems (★-Hammers)

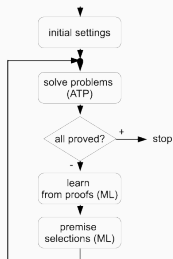


How much can it do?

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

High-level feedback loops – MALARea, ATPBoost

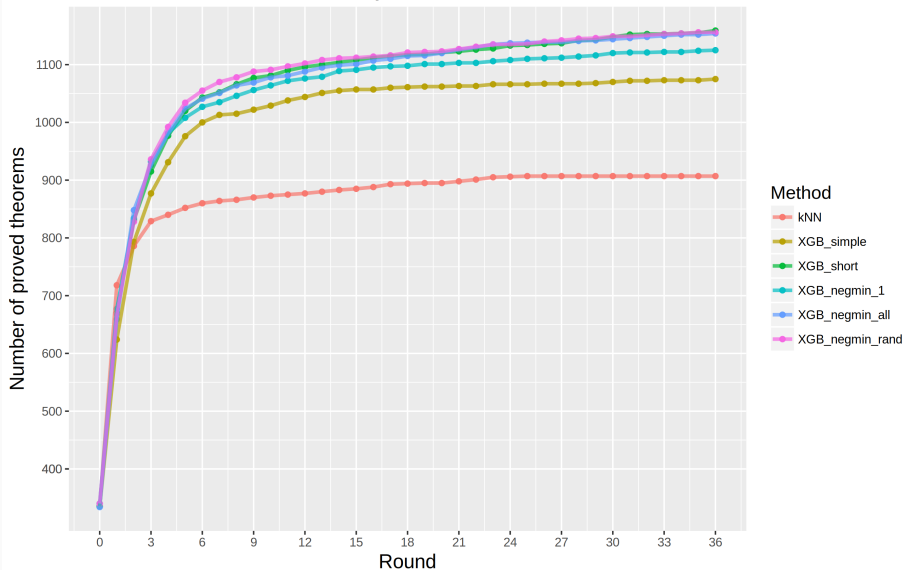
- Machine Learner for Autom. Reasoning (2006) – infinite hammering
- feedback loop interleaving **ATP** with **learning premise selection**
- both syntactic and **semantic** features for characterizing formulas:
- evolving set of finite (counter)models in which formulas evaluated
- winning AI/ATP benchmarks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



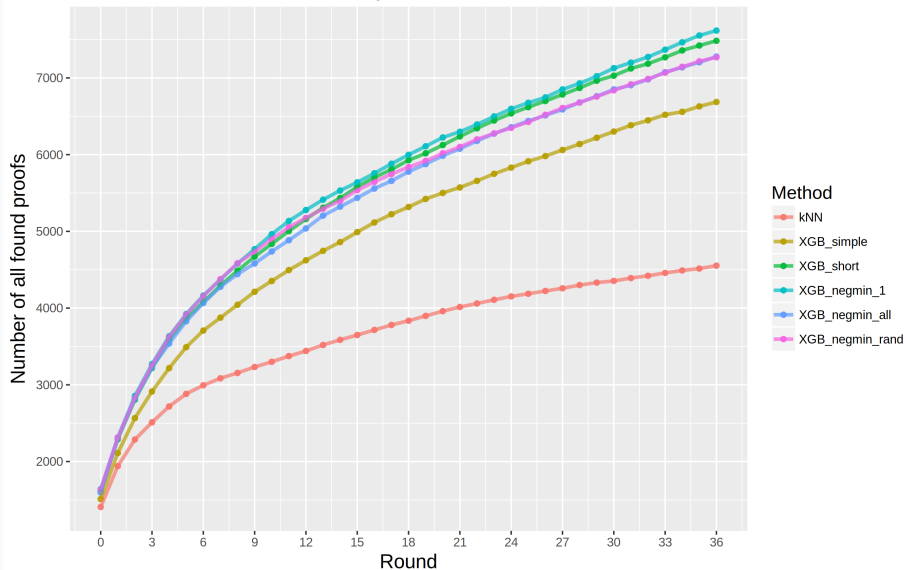
The screenshot shows a browser window with a table titled 'Results - Chromium'. The table compares the performance of various ATP solvers on the 'Large Theory Batch Problems' (LTB) benchmark. The solvers listed are MaLARE, E, IPraver, Zipperpit, Leo-III, ATPBoost, GKC, and Leo-III. The table shows the number of problems solved out of a total of 10,000, along with the percentage of solutions found.

| Large Theory Batch Problems | MaLARE | E | IPraver | Zipperpit | Leo-III | ATPBoost | GKC | Leo-III |
|-----------------------------|-----------------------|-----------------------|-----------------------|-----------------------|-----------------------|-----------------------|----------------------|----------------------|
| Solved ₁₀₀₀₀ | 7054 ₁₀₀₀₀ | 3393 ₁₀₀₀₀ | 3164 ₁₀₀₀₀ | 1699 ₁₀₀₀₀ | 1413 ₁₀₀₀₀ | 1237 ₁₀₀₀₀ | 493 ₁₀₀₀₀ | 134 ₁₀₀₀₀ |
| Solutions | 7054 70% | 3393 33% | 3163 31% | 1699 16% | 1413 14% | 1237 12% | 493 4% | 134 1% |

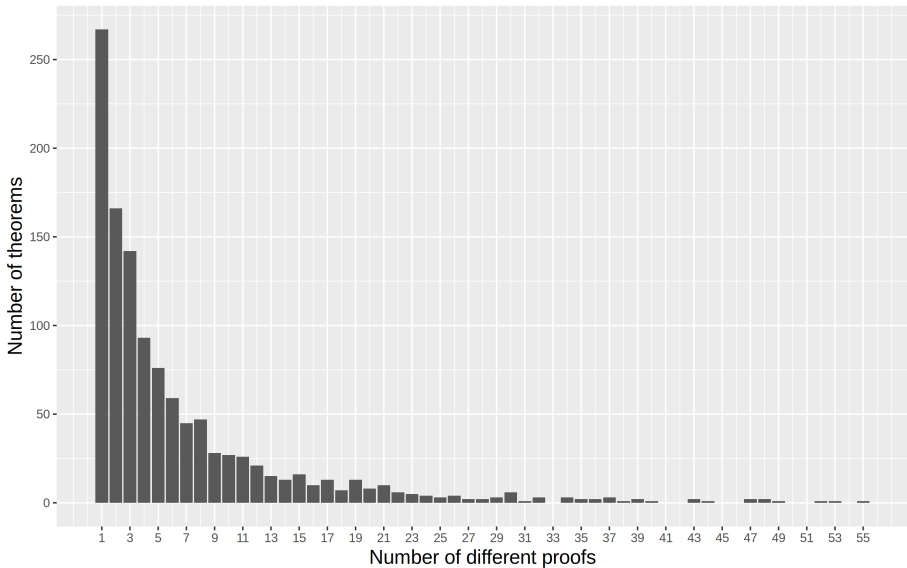
Prove-and-learn loop on MPTP2078 data set



Prove-and-learn loop on MPTP2078 data set



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



Outline

Quick Intro

Motivation, Learning vs. Reasoning

Learning of Theorem Proving - Overview

Demos

High-level Reasoning Guidance: Premise Selection

Low Level Guidance of Theorem Provers

Mid-level Reasoning Guidance

Synthesis

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:

$$c_1 : P(x)$$

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

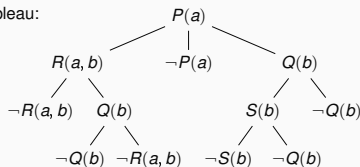
$$c_3 : S(x) \vee \neg Q(b)$$

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

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

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

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 – rICoP

- 2018: strong 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:

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

- learning both **policy** (p) (clause selection) and **value** (w) (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 also with GNNs (Olsak, Rawson, Zombori, ...)

Statistical Guidance of Connection Tableau – rICoP

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

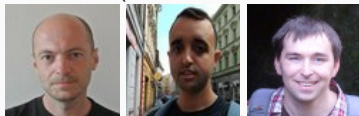
| System | leanCoP | bare prover | rICoP no policy/value (UCT only) |
|--------------------------|-------------|-------------|----------------------------------|
| Training problems proved | 10438 | 4184 | 7348 |
| Testing problems proved | 1143 | 431 | 804 |
| Total problems proved | 11581 | 4615 |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

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

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

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

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



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

Feedback prove/learn loop for ENIGMA on Mizar data

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

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

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

ENIGMA Anonymous: Learning from patterns only

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

More Low-Level Guidance of Various Creatures

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

Outline

Quick Intro

Motivation, Learning vs. Reasoning

Learning of Theorem Proving - Overview

Demos

High-level Reasoning Guidance: Premise Selection

Low Level Guidance of Theorem Provers

Mid-level Reasoning Guidance

Synthesis

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
- Demo: http://grid01.ciirc.cvut.cz/~mptp/tactictoe_demo.ogv
- 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 followup work for HOL Light (Google), Coq, Lean, ...

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/maintenance friendly**
- **Demo:** <https://blaauwbroek.eu/papers/cicm2020/demo.mp4>,
<https://coq-tactician.github.io/demo.html>
- Took several years, but could become a common tool for Coq formalizers
- Recently GNNs added, a major comparison of k-NN, GNN and LMs (Graph2Tac - <https://arxiv.org/abs/2401.02949>)

Outline

Quick Intro

Motivation, Learning vs. Reasoning

Learning of Theorem Proving - Overview

Demos

High-level Reasoning Guidance: Premise Selection

Low Level Guidance of Theorem Provers

Mid-level Reasoning Guidance

Synthesis

More on Conjecturing in Mathematics

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

Conjecturing and Proof Synthesis by Neural 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?

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

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

Mizar autocompletion server in action

Applications Places

GPT-2 generator trained on Mizar - Chromium

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

number of samples (fewer is raster)

Temperature (lower is less chaotic)

Length of output (shorter is faster)

Generate

Sample 1

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

Sample 2

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

Sample 3

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

[github]

Figure: MGG - Mizar Gibberish Generator.

Proving the conditioned completions - MizAR hammer

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

A correct conjecture that was too hard to prove

Kinyon and Stanovsky (algebraists) confirmed that this conjecture 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 :: SIN COS 10:17

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>
- Non-neural (Turing complete) symbolic computing and **semantics** collaborate with the statistical/neural learning
- Btw., we **experimentally verify Occam's Razor**
- **Evolving (self-improving) population of 4.5M matching explanations**
- Connections to Solomonoff Induction, AIXI, Gödel Machine?
- **50+ different characterizations of primes**, evolution governed by high-level criteria (Occam, efficiency)



OEIS: \geq 350000 finite sequences

The OEIS is supported by [the many generous donors to the OEIS Foundation](#).

0 1 3 6 2 7
: 13
: OE 20
23 IS 12
10 22 11 21

THE ON-LINE ENCYCLOPEDIA OF INTEGER SEQUENCES[®]

founded in 1964 by N. J. A. Sloane

 [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.

(Formerly M0652 N0241)

+30
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, ...

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: $+, -, \times, \text{div}, \text{mod}$
- Condition : if $\dots \leq 0$ then \dots else \dots
- $\text{loop}(f, a, b) := u_a$ where $u_0 = b$,

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

- Two other loop constructs: loop2 , a while loop

Example:

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

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

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:

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

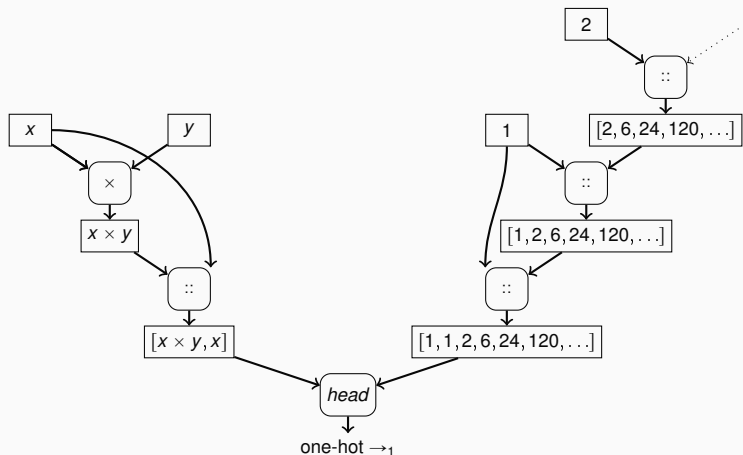
$$\lambda(x, y).a \in F, \text{loop}(f, a, b), \text{loop2}(f, g, a, b, c), \text{compr}(f, a) \in P$$

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

$$\begin{aligned} & [] \rightarrow_x [x] \rightarrow_y [x, y] \rightarrow_{\times} [x \times y] \rightarrow_x [x \times y, x] \\ & \rightarrow_1 [x \times y, x, 1] \rightarrow_{\text{loop}} [\text{loop}(\lambda(x, y). x \times y, x, 1)] \end{aligned}$$

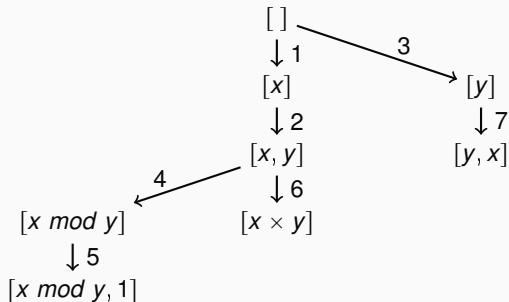
QSynt: Training of the Neural Net Guiding the Search

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



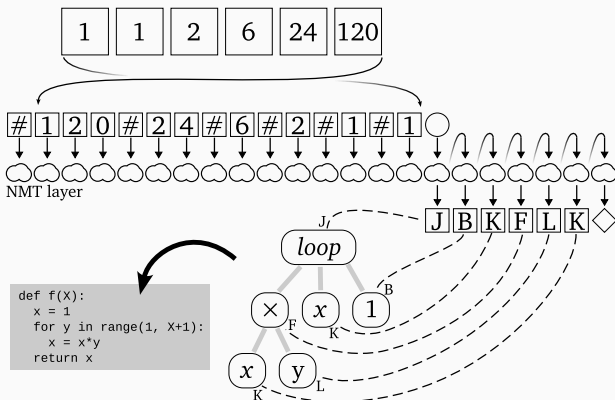
QSynt program search - Monte Carlo search tree

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



Encoding OEIS for Language Models

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



Search-Verify-Train Feedback Loop for OEIS

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

Search-Verify-Train Feedback Loop

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

QSynt web interface for program invention

Applications Places 896MHz Mon 11:40 Mon

grid01.ciirc.cvut.cz/~thibault/qsynt.html - Chromium

QSynt: AI rediscovers Fermat's Last Theorem x grid01.ciirc.cvut.cz/~thibault/qsynt.html x +

Not secure | grid01.ciirc.cvut.cz/~thibault/qsynt.html Incognito (2)

QSynt: Program Synthesis for Integer Sequences

Propose a sequence of integers:

Timeout (maximum 300s)

Generated integers (maximum 100)

A few sequences you can try:

```
0 1 1 0 1 0 1 0 0 0 1 0 1 0 0 0 1 0 1
0 1 4 9 16 21 25 28 36 37 49
0 1 3 6 10 15
2 3 5 7 11 13 17 19 23 29 31 37 41 43
1 1 2 6 24 120
2 4 16 256
```

QSynt inventing Fermat pseudoprimes

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

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

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

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

Program found in 5.81 seconds

f(x) := 2 + 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:



The screenshot shows the Brython web interface. At the top, the Brython logo is displayed. Below it are navigation links: Tutorial, Demo, Documentation, Console, Editor, Gallery, and Resources. On the right side, there is a language selector set to English. The main content area displays the Brython version (3.10.6) and a Python code editor. The code defines three functions: f2(X), f1(X), and f0(X). f2(X) is a simple linear function. f1(X) is a loop that iterates until a condition is met. f0(X) calls f1(X) and returns a value. The code is executed, and the output is shown in a terminal window on the right, displaying the first 16 numbers of the sequence: 2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53.

```
Brython version: 3.10.6
```

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

run Python Javascript Share code

```
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.: $(n - 1)! \equiv -1 \pmod n$)

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

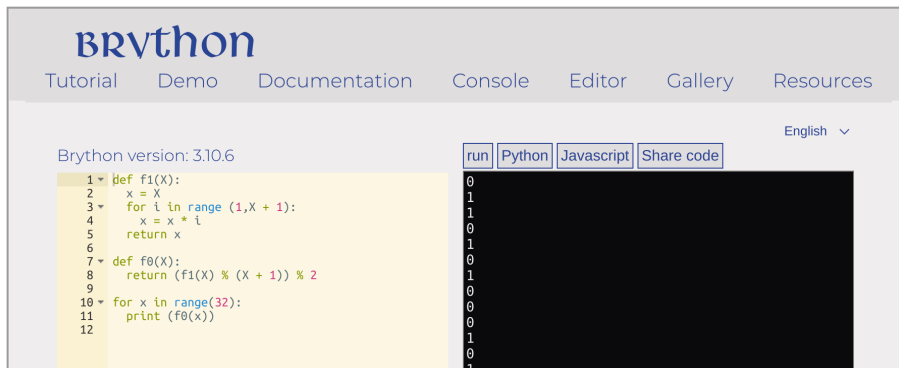
0 1 1 0 1 0 1 0 0 0 1 0 1 0 0 0 1 0 1 0 0 0 1 0 1 0 0 0 0 0 1 0 1 0

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

Program found in 5.17 seconds

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

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



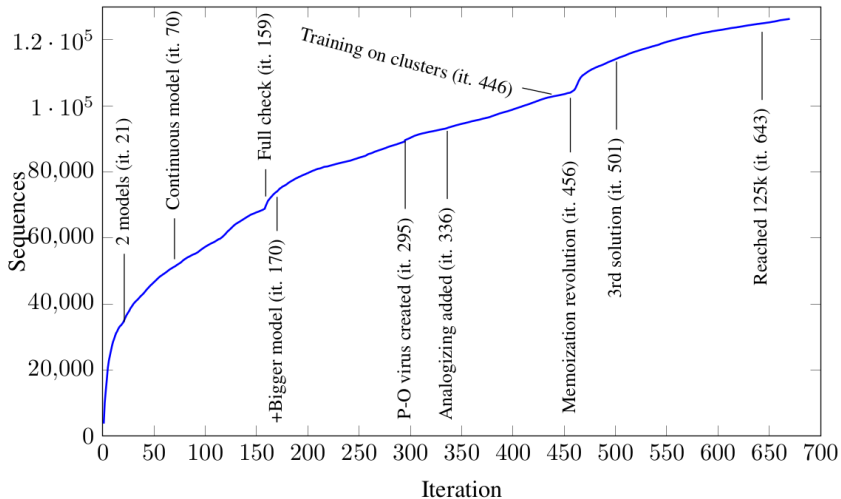
The screenshot shows the Brython web interface. At the top, the logo "Brython" is displayed in blue. Below it, there are navigation links: "Tutorial", "Demo", "Documentation", "Console", "Editor", "Gallery", and "Resources". On the right side, there is a language selector set to "English".

The main content area displays the Brython version: "Brython version: 3.10.6". Below this, there is a code editor with the following Python code:

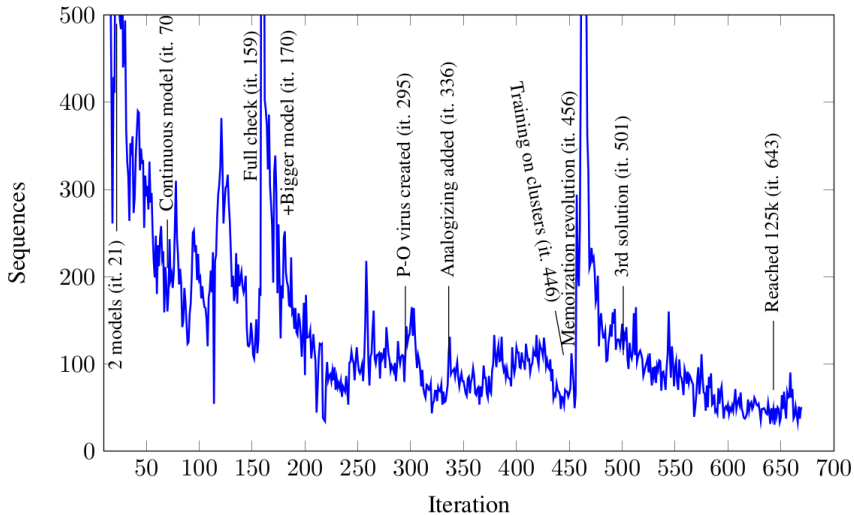
```
1 def f1(X):
2     x = X
3     for i in range(1, X + 1):
4         x = x * i
5     return x
6
7 def f0(X):
8     return (f1(X) % (X + 1)) % 2
9
10 for x in range(32):
11     print (f0(x))
12
```

To the right of the code editor, there are four buttons: "run", "Python", "Javascript", and "Share code". Below these buttons, the output of the program is displayed in a black terminal window, showing the sequence of 32 generated numbers: 0, 1, 1, 0, 1, 0, 1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 1, 0.

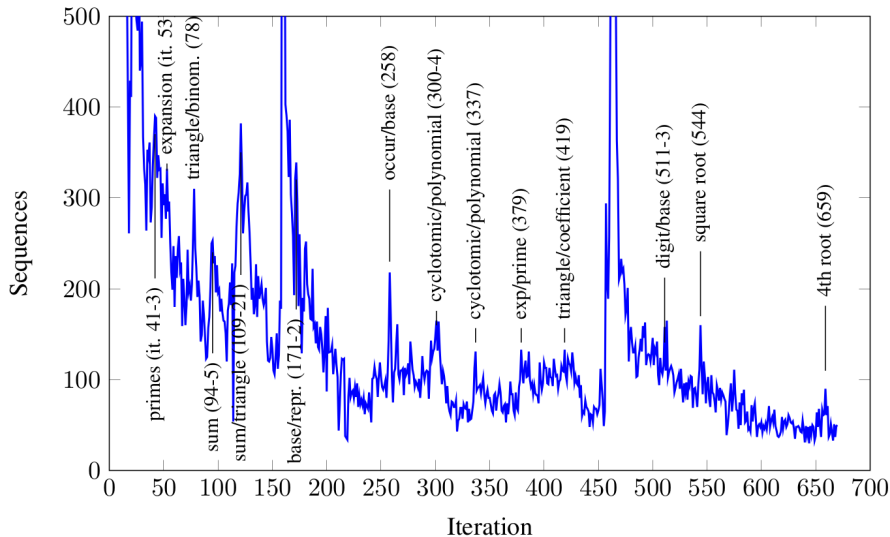
Human Made Technology Jumps



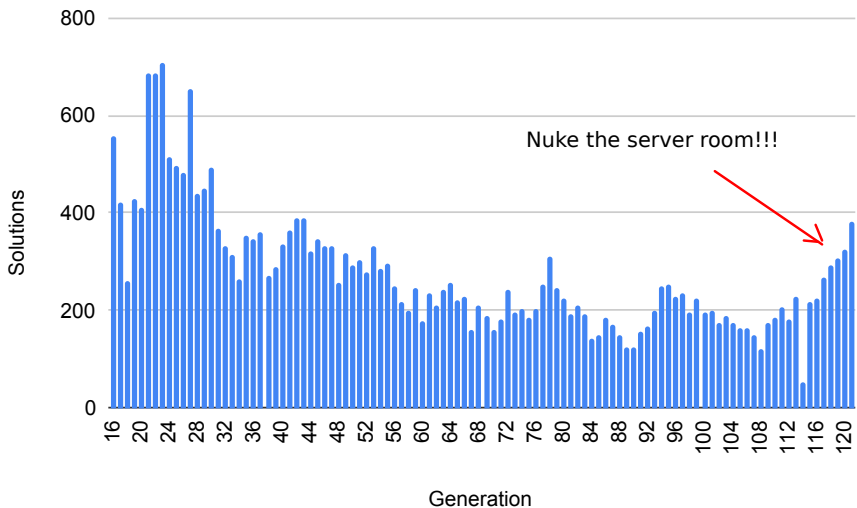
Human Made Technology Jumps



Some Automatic Technology Jumps



Singularity Take-Off X-mas Card



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 $\text{binomial}(i, j) * 10^{i-j} * 11^j$
- iter 94-5: sum: A100192 $a(n) = \text{Sum}_{k=0..n} \text{binomial}(2n, n+k) * 2^k$
- 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.

Generalization of the Solutions to Larger Indices

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

Are two QSynt programs equivalent?

- As with primes, we often find **many programs** for one OEIS sequence
- Currently we have almost 4.5M programs for the 126k sequences
- It may be quite hard to see that the programs **are equivalent**
- Extend to Schmidhuber's **Gödel Machine?**
- A simple example for 0, 2, 4, 6, 8, ... with two programs f and g :
 - $f(0) = 0, f(n) = 2 + f(n - 1)$ if $n > 0$
 - $g(n) = 2 * n$
 - conjecture: $\forall n \in \mathbb{N}. g(n) = f(n)$
- We can ask mathematicians, but we have **thousands of such problems**
- Or we can try to **ask our ATPs** (and thus create a large ATP benchmark)!
- Here is one SMT encoding by Janota & Gauthier:

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

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

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

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

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. |

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.
- $\text{loop2}(((y * y) \text{ div } (x + y)) + y, y, x + x, 2, \text{loop}((1 + 2) * x, x, 2)) \text{ mod } (1 + 2)$
- **MO**: *Not a proof, just a rough idea: The program iterates the function $q \mapsto 2+q / 1+q$, 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, $a = \text{floor}(3^x * \sqrt{8})$, so $a \text{ mod } 3$ 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 $q * \text{Product}_{k \geq 1} (1 - q^k)^2 * (1 - q^{11*k})^2$

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)
```

Minsky 2014

*It seems to me that the **most important discovery since Gödel** was the discovery by Chaitin, Solomonoff and Kolmogorov of the concept called **Algorithmic Probability** which is a fundamental new theory of how to make predictions given a collection of experiences and this is a beautiful theory, everybody should learn it,*

but it's got one problem, that is, that you cannot actually calculate what this theory predicts because it is too hard, it requires an infinite amount of work.

*However, it should be possible to **make practical approximations** to the Chaitin, Kolmogorov, Solomonoff theory that would make better predictions than anything we have today. Everybody should learn all about that and **spend the rest of their lives working on it.***

– M. Minsky, Panel discussion on The Limits of Understanding, 2014

References to our relevant work

- L. Blaauwbroek, D. Cerna, T. Gauthier, J. Jakubuv, C. Kaliszyk, M. Suda, J. Urban: Learning Guided Automated Reasoning: A Brief Survey. *Logics and Type Systems in Theory and Practice 2024*: 54-83
- Thibault Gauthier, Josef Urban: Learning Program Synthesis for Integer Sequences from Scratch. *AAAI 2023*: 7670-7677
- Thibault Gauthier, Miroslav Olsák, Josef Urban: Alien Coding. *CoRR abs/2301.11479 (2023)*
- Thibault Gauthier, Chad E. Brown, Mikolas Janota, Josef Urban: A Mathematical Benchmark for Inductive Theorem Provers. *LPAR 2023*: 224-237
- Thibault Gauthier, Cezary Kaliszyk, Josef Urban: TacticToe: Learning to Reason with HOL4 Tactics. *LPAR 2017*: 125-143
- Thibault Gauthier, Cezary Kaliszyk, Josef Urban, Ramana Kumar, Michael Norrish: Learning to Prove with Tactics. *CoRR abs/1804.00596 (2018)*
- 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
- Thibault Gauthier: Deep Reinforcement Learning for Synthesizing Functions in Higher-Order Logic. *LPAR 2020*: 230-248
- Thibault Gauthier: Tree Neural Networks in HOL4. *CICM 2020*: 278-283
- Qingxiang Wang, Cezary Kaliszyk, Josef Urban: First Experiments with Neural Translation of Informal to Formal Mathematics. *CICM 2018*: 255-270
- Bartosz Piotrowski, Josef Urban: Stateful Premise Selection by Recurrent Neural Networks. *LPAR 2020*: 409-422
- 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
- Bartosz Piotrowski, Josef Urban, Chad E. Brown, Cezary Kaliszyk: Can Neural Networks Learn Symbolic Rewriting? *CoRR abs/1911.04873 (2019)*

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