AI FOR AUTOMATED AND INTERACTIVE THEOREM PROVING

Josef Urban

Czech Technical University in Prague

Novos Talentos 2022
Coimbra, July 24, 2022

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

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

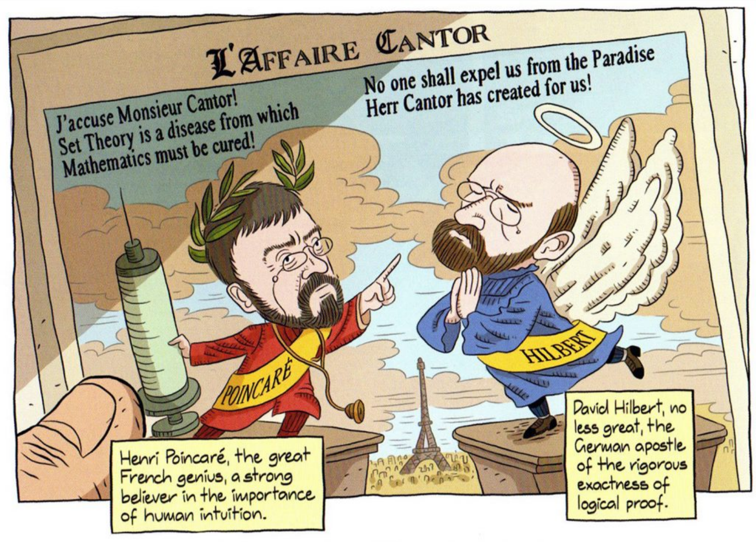
[Adapted from: *Logicomix: An Epic Search for Truth* by A. Doxiadis]

How Do We Automate Math, Science, Programming?

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

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

Intuition vs Formal Reasoning – Poincaré vs Hilbert



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

What is Formal Mathematics and Theorem Proving?

- 1900s: Mathematics put on formal logic foundations – symbolic logic
- Culmination of a program by Leibniz/Frege/Russell/Hilbert/Church/...
- ... led also to the rise of computers (Turing/Church, 1930s)
- ... and rise of AI - Turing's 1950 paper: Learning Machines, Chess, etc.
- 1950s: First AI program: Logic Theorist by Newell & Simon
- Formalization of math (60s): combine formal foundations and computers
- Proof assistants/Interactive theorem provers and their large libraries:
- Automath (1967), LCF, Mizar, NQTHM, HOL, Coq, Isabelle, ACL2, Lean
- Automated theorem provers - search for proofs automatically:
- Otter, Vampire, E, SPASS, Prover9, CVC4, Z3, Satallax, ...
- more limited logics: SAT, QBF, SMT, UEQ, ... (DPLL, CDCL, ...)
- TP-motivated PLs: ML, Prolog, (*logic programming* - Hayes, Kowalski)
- My MSc (1998): Try ILP to learn explainable rules/heuristics from Mizar
- Since: Do AI/TP over (in)formal math corpora: Mizar, Isabelle, HOL, ...

Why Do This Today?

1. Practically Useful for Verification of Complex HW/SW and Math
   - Formal Proof of the Kepler Conjecture (2014 – Hales – 20k lemmas)
   - Formal Proof of the Feit-Thompson Theorem (2 books, 2012 – Gonthier)
   - Verification of several math textbooks and CS algorithms
   - Verification of compilers (CompCert)
   - Verification of OS microkernels (seL4), HW chips (Intel), transport, finance,
   - Verification of cryptographic protocols (Amazon), etc.
2. Blue Sky AI Visions:
   - Get strong AI by learning/reasoning over large KBs of human thought?
   - Big formal theories: good semantic approximation of such thinking KBs?
   - Deep non-contradictory semantics – better than scanning books?
   - Gradually try learning math/science
   - automate/verify them, include law, etc. (Leibniz, McCarthy, ..)
     - What are the components (inductive/deductive thinking)?
     - How to combine them together?

What is Formal Mathematics?

- Developed thanks to the Leibniz/Russell/Frege/Hilbert/... program
- Mathematics put on formal logic foundations (*symbolic computation*)
- ... which btw. led also to the rise of computers (Turing/Church, 1930s)
- Formal math (1950/60s): combine formal foundations and the newly available computers
- For AGI/Singularity people: Formal proof is the *Secure Hardware Environment* from Vinge's Rainbows End
- Conceptually very simple:
- Write all your axioms and theorems so that computer understands them
- Write all your inference rules so that computer understands them
- Use the computer to check that your proofs follow the rules
- But in practice, it turns out not to be so simple
- Many approaches, still not mainstream, but big breakthroughs recently

Irrationality of $\sqrt{2}$ (informal text)

tiny proof from Hardy & Wright:

Theorem 43 (Pythagoras' theorem). $\sqrt{2}$ is irrational.
The traditional proof ascribed to Pythagoras runs as follows. If $\sqrt{2}$ is rational, then the equation

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

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

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

exactly the same text in Mizar syntax:

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

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

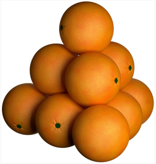
```
let SQRT_2_IRRATIONAL = prove
 (`~rational(sqrt(&2))`,
  SIMP_TAC[rational; real_abs; SQRT_POS_LE; REAL_POS] THEN
  REWRITE_TAC[NOT_EXISTS_THM] THEN REPEAT GEN_TAC THEN
  DISCH_THEN(CONJUNCTS_THEN2 ASSUME_TAC MP_TAC) THEN
  SUBGOAL_THEN `~((&p / &q) pow 2 = sqrt(&2) pow 2)`
    (fun th -> MESON_TAC[th]) THEN
  SIMP_TAC[SQRT_POW_2; REAL_POS; REAL_POW_DIV] THEN
  ASM_SIMP_TAC[REAL_EQ_LDIV_EQ; REAL_OF_NUM_LT; REAL_POW_LT;
               ARITH_RULE `0 < q <=> ~(q = 0)`] THEN
  ASM_MESON_TAC[NSQRT_2; REAL_OF_NUM_POW; REAL_OF_NUM_MUL; REAL_OF_NUM_EQ]);;
```

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

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

Big Example: The Flyspeck project

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



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

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

Big Formalizations

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

Mid-size Formalizations

- Gödel's First Incompleteness Theorem — Natarajan Shankar (NQTHM), Russell O'Connor (Coq)
- Brouwer Fixed Point Theorem — Karol Pak (Mizar), John Harrison (HOL Light)
- Jordan Curve Theorem — Tom Hales (HOL Light), Artur Kornilowicz et al. (Mizar)
- Prime Number Theorem — Jeremy Avigad et al (Isabelle/HOL), John Harrison (HOL Light)
- Gödel's Second incompleteness Theorem — Larry Paulson (Isabelle/HOL)
- Central Limit Theorem – Jeremy Avigad (Isabelle/HOL)

Large Software Verifications

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

Central Limit Theorem in Isabelle/HOL

The Top 100 Theorems in Isabelle - Chromium

The Top 100 Theo ×

www.cse.unsw.edu.au/~kleing/top100/#47

```
theorem (in prob_space) central_limit_theorem:
  fixes
    X :: "nat ⇒ 'a ⇒ real" and
    μ :: "real measure" and
    σ :: real and
    S :: "nat ⇒ 'a ⇒ real"
  assumes
    X_indep: "indep_vars (λi. borel) X UNIV" and
    X_integrable: "⋀n. integrable M (X n)" and
    X_mean_0: "⋀n. expectation (X n) = 0" and
    σ_pos: "σ > 0" and
    X_square_integrable: "⋀n. integrable M (λx. (X n x)²)" and
    X_variance: "⋀n. variance (X n) = σ²" and
    X_distrib: "⋀n. distr M borel (X n) = μ"
  defines
    "S n ≡ λx. ∑i<n. X i x"
  shows
    "weak_conv_m (λn. distr M borel (λx. S n x / sqrt (n * σ²)))
        (density lborel std_normal_density)"
```

Sylow's Theorems in Mizar

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

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

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

Gödel Theorems in Isabelle

The Top 100 Theorems in Isabelle - Chromium

The Top 100 Theo

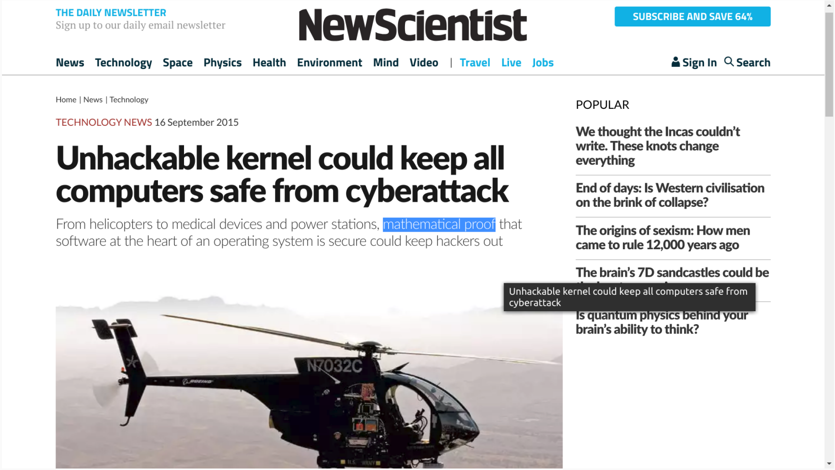
www.cse.unsw.edu.au/~kleing/top100/#6

```
theorem Goedel_I:
  assumes "¬ {} ⊢ Fls"
  obtains δ where
      "{} ⊢ δ IFF Neg (PfP ⌈δ⌉)"
      "¬ {} ⊢ δ"
      "¬ {} ⊢ Neg δ"
      "eval_fm e δ"
      "ground_fm δ"

theorem Goedel_II:
  assumes "¬ {} ⊢ Fls"
    shows "¬ {} ⊢ Neg (PfP ⌈Fls⌉)"
```

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

Today's Applications



THE DAILY NEWSLETTER
Sign up to our daily email newsletter

NewScientist

SUBSCRIBE AND SAVE 64%

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

Home | News | Technology

TECHNOLOGY NEWS 16 September 2015

Unhackable kernel could keep all computers safe from cyberattack

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

POPULAR

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

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

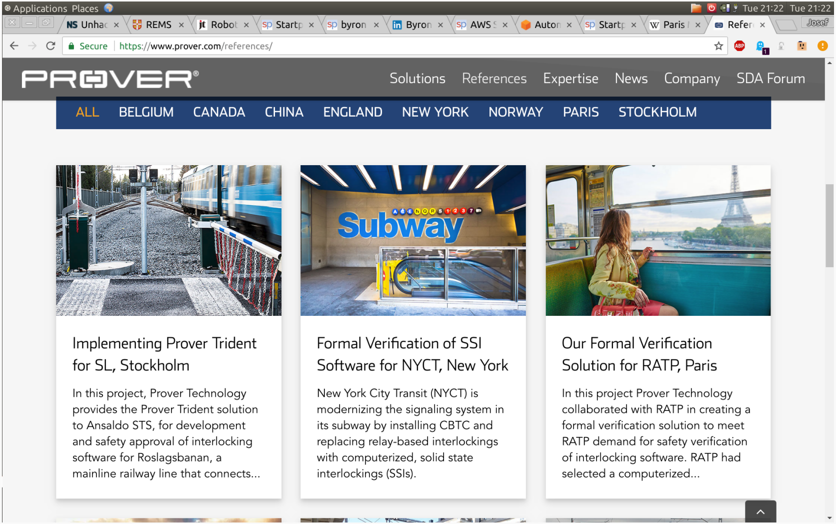
The origins of sexism: How men came to rule 12,000 years ago

The brain's 7D sandcastles could be

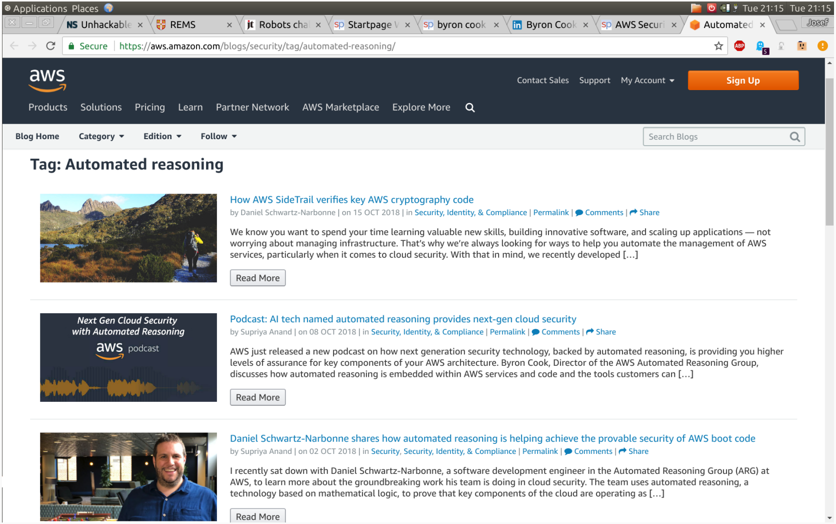
Unhackable kernel could keep all computers safe from cyberattack

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

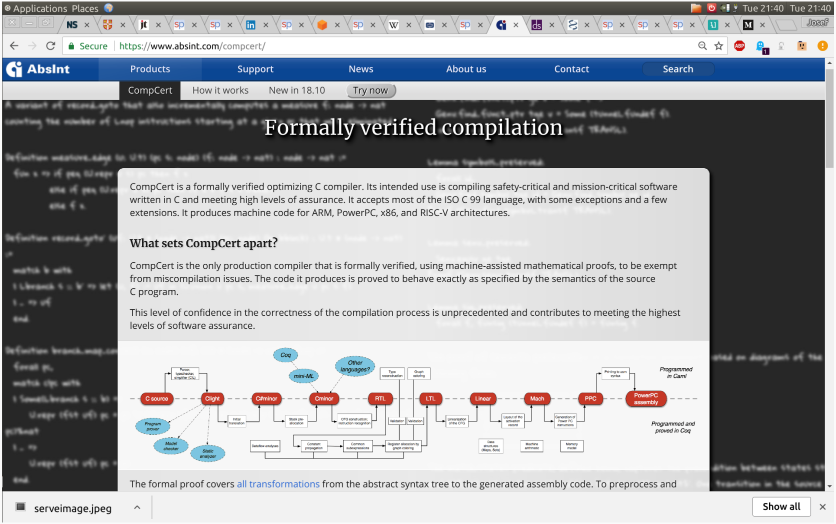
Today’s Applications



Today's Applications



Today’s Applications



Today's Applications

Overview About Research Education Industry People Projects Events Publications Institutions Jobs Visitors Student Positions Login

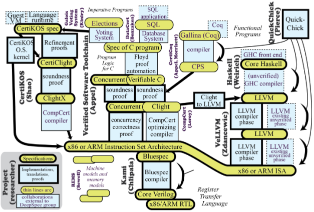
deep spec **the science of deep specification**

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

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

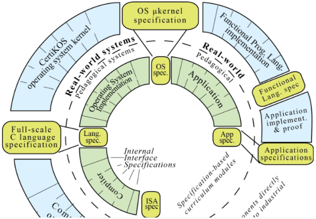
Research

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

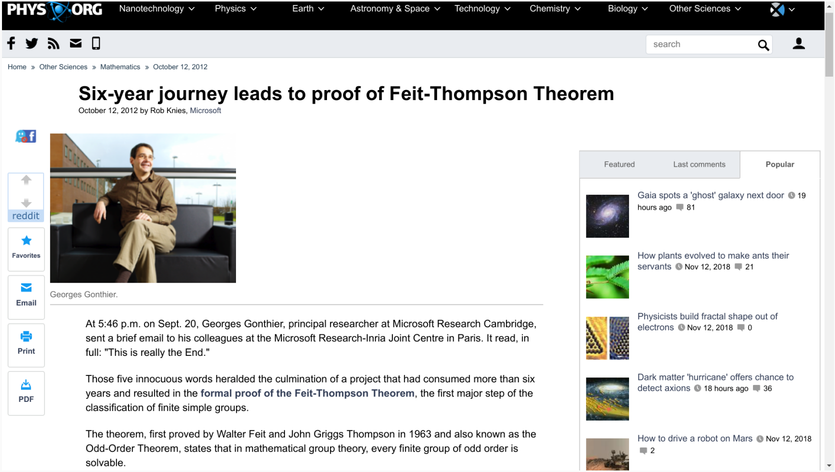


Education

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



Today’s Applications



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

search

Home » Other Sciences » Mathematics » October 12, 2012

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

October 12, 2012 by Rob Knies, Microsoft

reddit

Favorites

Email

Print

PDF

Georges Gonthier.

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

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

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

Featured | Last comments | **Popular**

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

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

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

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

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

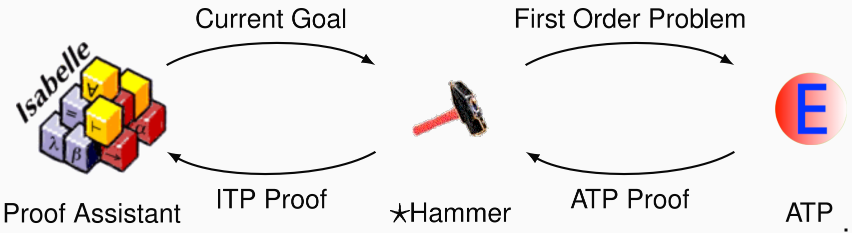
Using Learning to Guide Theorem Proving

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

AI/TP Examples and Demos

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

Today's AI-ATP systems (⋆-Hammers)



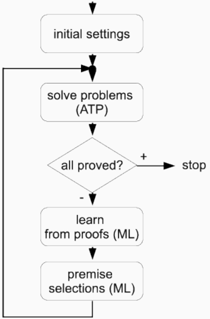
How much can it do?

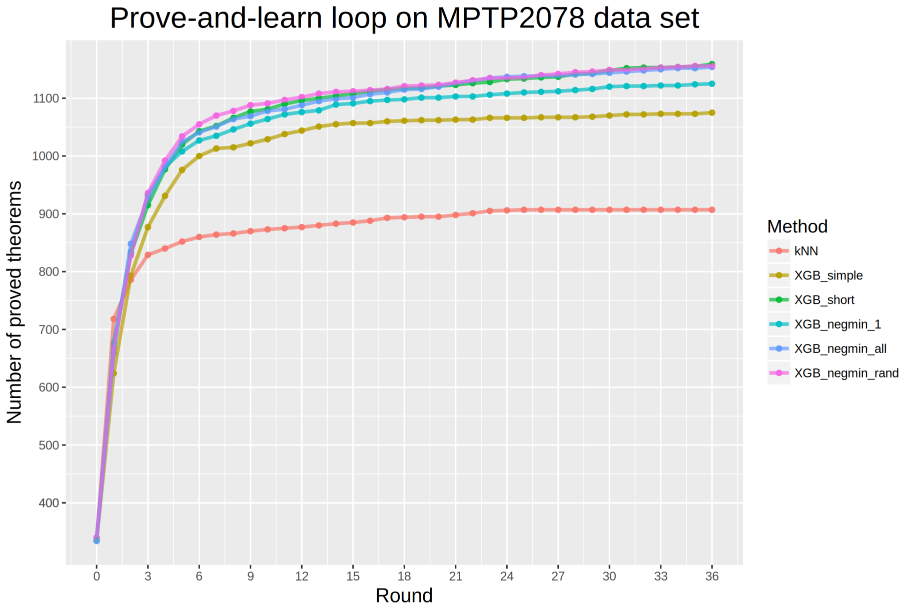
- Mizar / MML – MizAR
- Isabelle (Auth, Jinja) – Sledgehammer
- Flyspeck (including core HOL Light and Multivariate) – HOL(y)Hammer
- HOL4 (Gauthier and Kaliszyk)
- CoqHammer (Czajka and Kaliszyk) - about 40% on Coq standard library

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

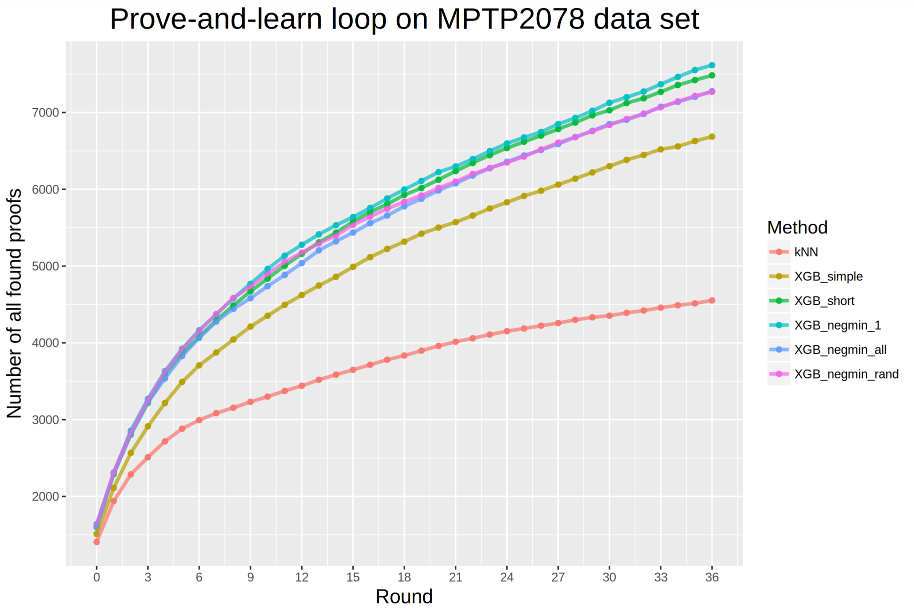
High-level feedback loops – MALARea, ATPBoost

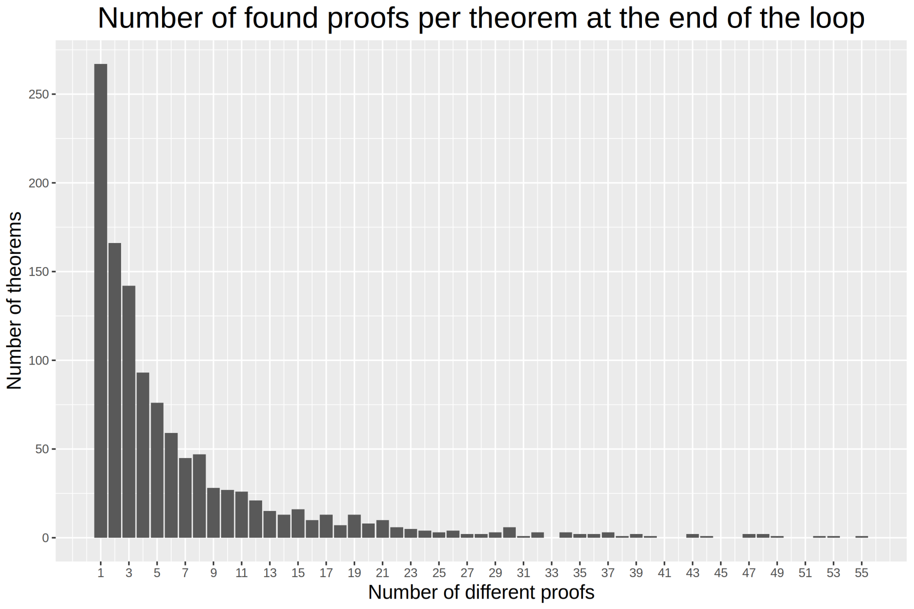
- Machine Learner for Autom. Reasoning (2006) – infinite hammering
- feedback loop interleaving ATP with learning premise selection
- both syntactic and semantic features for characterizing formulas:
- evolving set of finite (counter)models in which formulas evaluated
- winning AI/ATP benchmarks (MPTPChallenge, CASC 2008/12/13/18)
- ATPBoost (Piotrowski) - recent incarnation focusing on multiple proofs





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

Low-level: Statistical Guidance of Connection Tableau

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

Clauses:

$c_1 : P(x)$

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

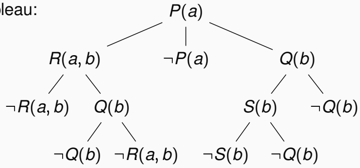
$c_3 : S(x) \lor \neg Q(b)$

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

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

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

Closed Connection Tableau:



Statistical Guidance of Connection Tableau – rlCoP

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

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

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

Statistical Guidance of Connection Tableau – rlCoP

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

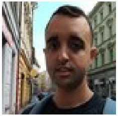
System	leanCoP	bare prover	rlCoP no policy/value (UCT only)
Training problems proved	10438	4184	7348
Testing problems proved	**1143**	431	804
Total problems proved	11581	4615	8152

- 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

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

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


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

Feedback prove/learn loop for ENIGMA on Mizar data

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

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

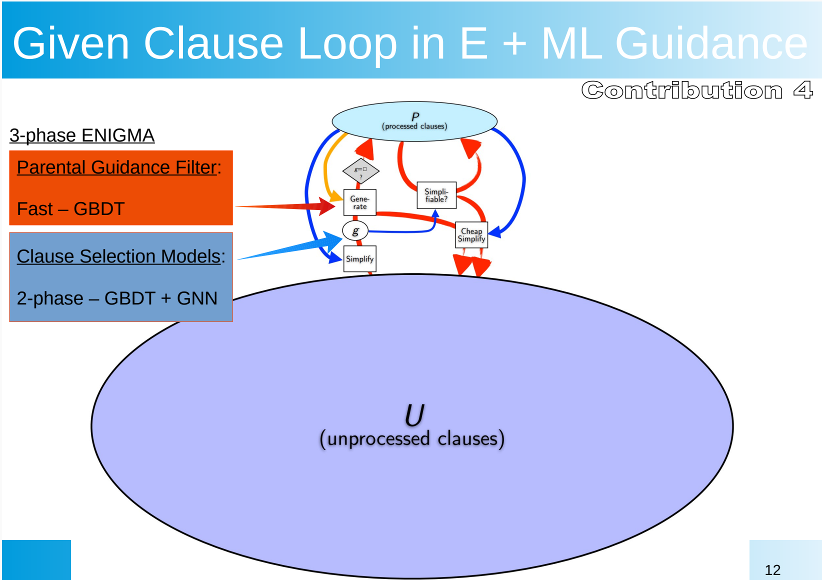
	$\mathcal{S} \odot \mathcal{M}_{12}^3$	$\mathcal{S} \oplus \mathcal{M}_{12}^3$	$\mathcal{S} \odot \mathcal{M}_{16}^3$	$\mathcal{S} \oplus \mathcal{M}_{16}^3$
solved	24159	24701	25100	**25397**
$\mathcal{S}\%$	+61.1%	+64.8%	+68.0%	**+70.0%**
$\mathcal{S}+$	+9761	+10063	+10476	+10647
$\mathcal{S}-$	-535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

ENIGMA Anonymous: Learning from patterns only

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

3-phase Anonymous ENIGMA

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



Neural Clause Selection in Vampire (M. Suda)

Deepire: Similar to ENIGMA:

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

Deepire's contributions:

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

Preliminary Evaluation on Mizar "57880"

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

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
- Speed more important than better learners
- Fully integrated with Coq, should work for any development
- User friendly, installation friendly, integration friendly and maintenance friendly
- Took several years, but could become a very common tool for Coq formalizers

More on Conjecturing in Mathematics

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

Conjecturing and Proof Synthesis by Neural Language models

- Karpathy’15 - RNN experiments with generating fake Math over Stacks
- I have tried to use that for formal math in 2016 but it looked weak
- GPT (-2,3) looks stronger
- Renewed experiments in 2020 on:
- All Mizar articles, stripped of comments and concatenated together (78M)
- Articles with added context/disambiguation (156M) (types, names, thesis)
- TPTP proofs of 28271 Mizar/MPTP theorems by E/ENIGMA (658M)
- Just the conjecture and premises needed for the 28271 proofs printed in prefix notation
- Quite interesting results, server for Mizar authors
- Quickly taken up by others on HOL, Isabelle, MetaMath ...

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

```
:: 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 “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

- Gauthier’19-22
- Synthesize math expressions based on semantic characterizations
- i.e., not just on the syntactic descriptions (e.g. proof situations)
- Tree Neural Nets and RL (MCTS, policy/value), used for:
- Guiding synthesis of a diophantine equation characterizing a given set
- Guiding synthesis of combinators for a given lambda expression
- 2022: invention of programs for OEIS sequences from scratch
- 50k sequences discovered so far:

  `https://www.youtube.com/watch?v=24oejR9wsXs`,

  `http://grid01.ciirc.cvut.cz/~thibault/qsynt.html`
- Many conjectures invented: 4 different characterizations of primes
- Non-neural (Turing complete) computing and semantics collaborates with the statistical learning

QSynt: synthesizing the programs/expressions

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

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

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

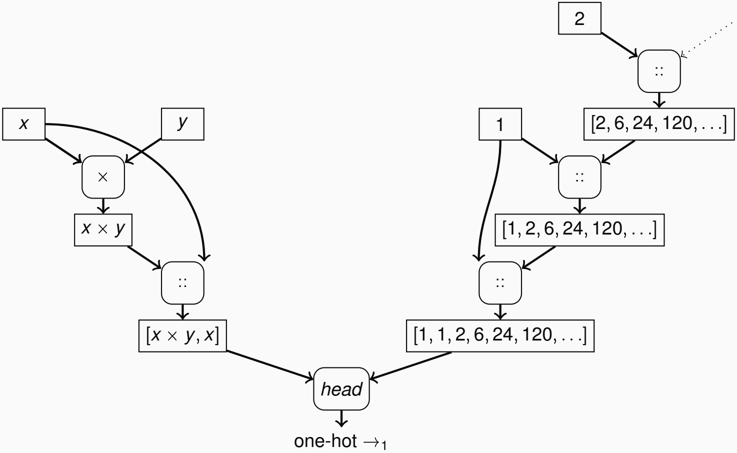
- Programs are built in reverse polish notation
- Start from an empty stack
- Use ML to repeatedly choose the next operator to push on top of a stack
- Example: Factorial is $loop(\lambda(x, y).\ x \times y, x, 1)$, built by:

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

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

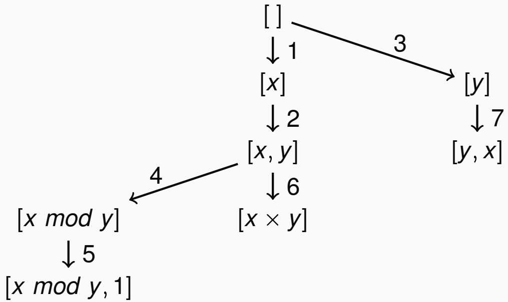
QSynt: Training of the Neural Net Guiding the Search

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

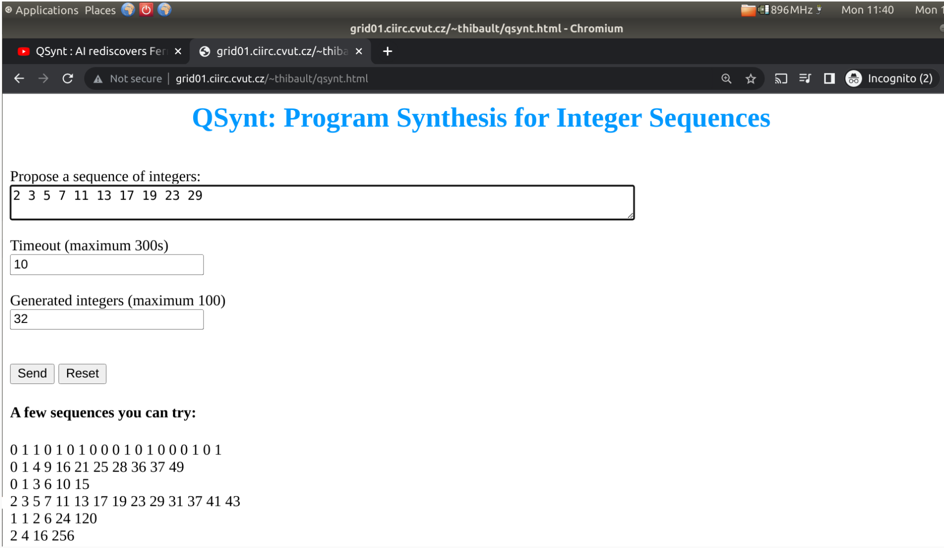


QSynt program search - Monte Carlo search tree

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



QSynt web interface for program invention



QSynt inventing Fermat pseudoprimes

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

```
First 16 generated numbers (f(0),f(1),f(2),...):
2 3 5 7 11 13 17 19 23 29 31 37 41 43 47 53
Generated sequence matches best with: A15919(1-75), A100726(0-59), A40(0-58)

Program found in 5.81 seconds
f(x) := 2 + compr(\x.loop(\(x,i).2*x + 2, x, 2) mod (x + 2), x)
Run the equivalent Python program here or in the window below:
```

Brython

Tutorial Demo Documentation Console Editor Gallery Resources

English

Brython version: 3.10.6

run Python Javascript Share code

```
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

```
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
```

Are two QSynt programs equivalent?

- As with primes, we often find many programs for one OEIS sequence
- It may be quite hard to see that the programs are equivalent
- A simple example for $0, 2, 4, 6, 8, \ldots$ 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
```

Future: AITP Challenges/Bets from 2014

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

Acknowledgments

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

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

Hausdorff trimester on formal math in 2024