AI AND REASONING

Josef Urban

Czech Technical University in Prague

AI Symposium @ CIIRC CTU
May 31, 2023

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]

Our project: Combine modern AI and Reasoning methods; connect/apply them to research in related fields.

People

- Seniors: C. Brown (US), Z. Hanzalek, M. Janota, V. Kucera, V. Marik, P. Sucha, M. Svitek
- Juniors/Phd: L. Blaauwbroek (NL), K. Chvalovsky, Z. Goertzel (US), M. Hasal, B. Hudcova, J. Jakubuv, J. Kanis, S. Kozhevnikov (RU), Y. Nagashima (JP), J. Piepenbrock (NL), M. Postranecky, J. Zelinka
- Admin team: M. Adler, V. Fencl, H. Puklicka
- Affiliated researchers: R. Veroff (US), A. Pease (US)
- Partner institutions: Radboud U. Nijmegen, U. of Innsbruck
- Contacts there: H. Geuvers (NL), T. Heskes (NL), C. Kaliszyk (PL/AT)
- some 40 visitors over 6 years

Faces

Visitors (who is who in automated reasoning?)

John Harrison, Robert Veroff, Stephan Schulz, Cezary Kaliszyk, Tanel Tammet, Michael Beeson, Deepak Kapur, Geoff Sutcliffe, Moa Johansson, Markus Wenzel, Hendrik Prakken, Mario Carneiro, Adam Pease, Edward Zalta, Peter Koepke, Mark Adams, Ramana Kumar, Thibault Gauthier, Zsolt Zombori, Bartosz Piotrowski, Qingxiang Wang, John Hester, Yuli Daune Funato, Minchao Wu, Miroslav Olšák, Tobias Grosser, Michael Färber, Liao Zhang, Mathieu Chanavat, Nikolai Antonov, Jens Otten, Jelena Mitrović, Jan Heemstra, Choiwah Chow, Michal Buráň, Jenmifer Pease, Petra Hozzová, Michael Rawson , Goncalo Araujo, Adrian De Lon

Topics

- Automated and Interactive Theorem Proving, Automated Formalization
- Combinations with Machine Learning, Ontologies
- Satisfiability Modulo Theories, Complex systems and Artificial Life
- Theory of Systems and Automatic Control, Data Science
- Operations research, Optimization, Scheduling and real-time systems
- Human-Computer Interaction, Natural language processing, Sign language processing
- Altogether over 100 papers, more than 50% with international co-authors

Results Highlights

- WP1: First convincing systems that combine learning and proving
- WP1: First systems for automated formalization, neural conjecturing and synthesis
- WP1: Multiple victories in theorem proving competitions
- WP1: Open conjectures solved in algebraic loop theory
- WP2: Mixed criticality scheduling, optimization of robotic cells
- WP2: Victory in Formula 1/10 Autonomous Racing Competition
- WP2: AI for Smart Cities
- WP3: Long standing problem solved in control theory (Czech Mind for V. Kucera)
- WP3: Victory in a Hand Pose Estimation competition
- WP3: Started research in classification of complex systems

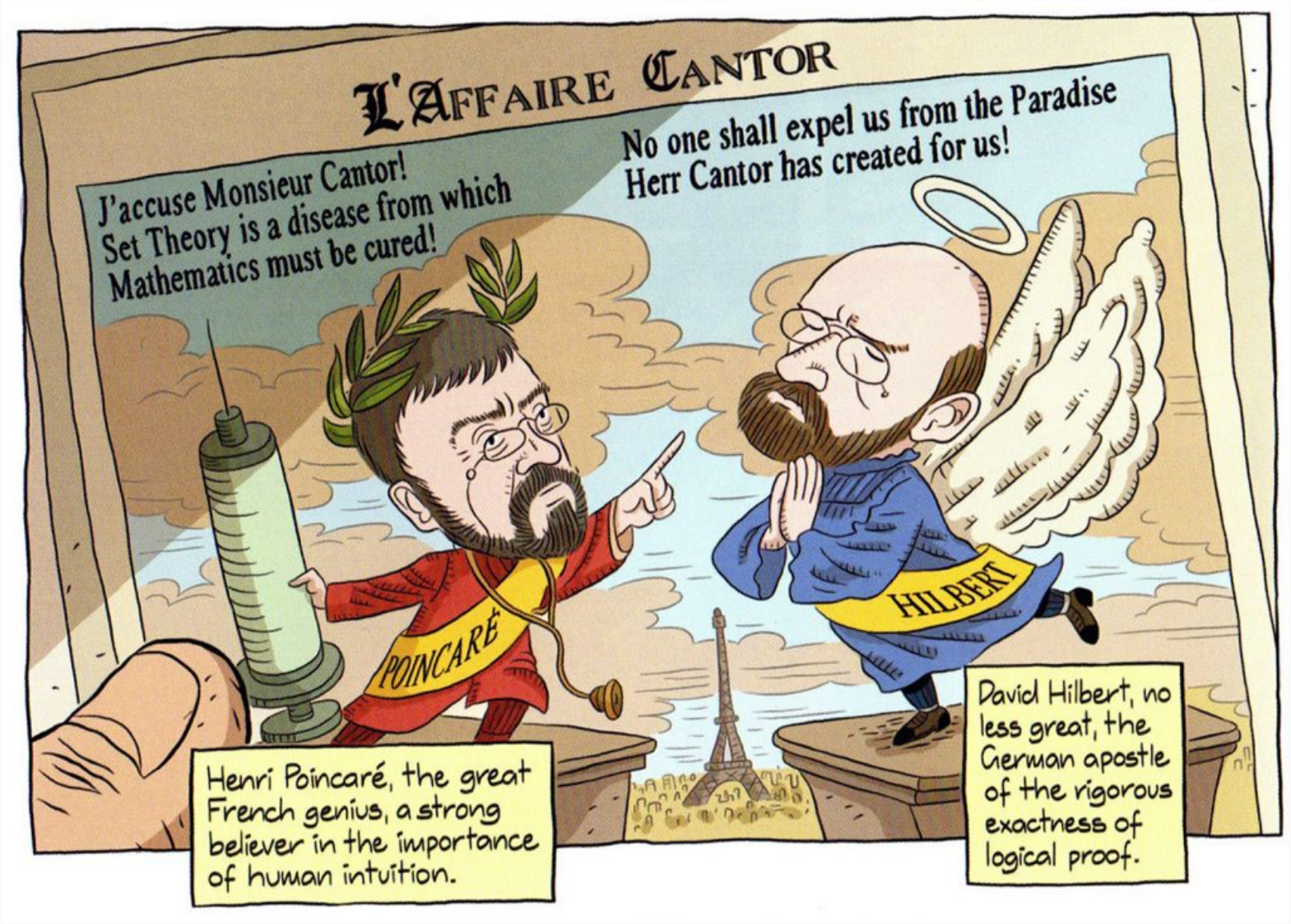
How Do We Automate Math, Science, Programming?

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 try to 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/Zuse, 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)
- Our work: 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?

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)

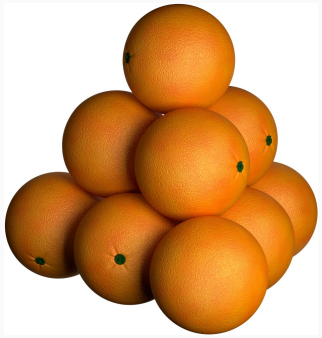
exactly the same text in Mizar syntax:

```
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;
```

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!
- In 2014, C. Kaliszyk and I automatically proved 40% of the 20k lemmas

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`

ENIGMA (2017-23): Guiding the Best ATPs like E

- ENIGMA (J. Jakubuv, Z. Goertzel, K. Chvalovsky, cf. M. Suda's talk)



- 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)
- Learning from structure only (unusual analogizing capability)
- 2021: leapfrogging and Split&Merge:
- aiming at learning reasoning/algo components

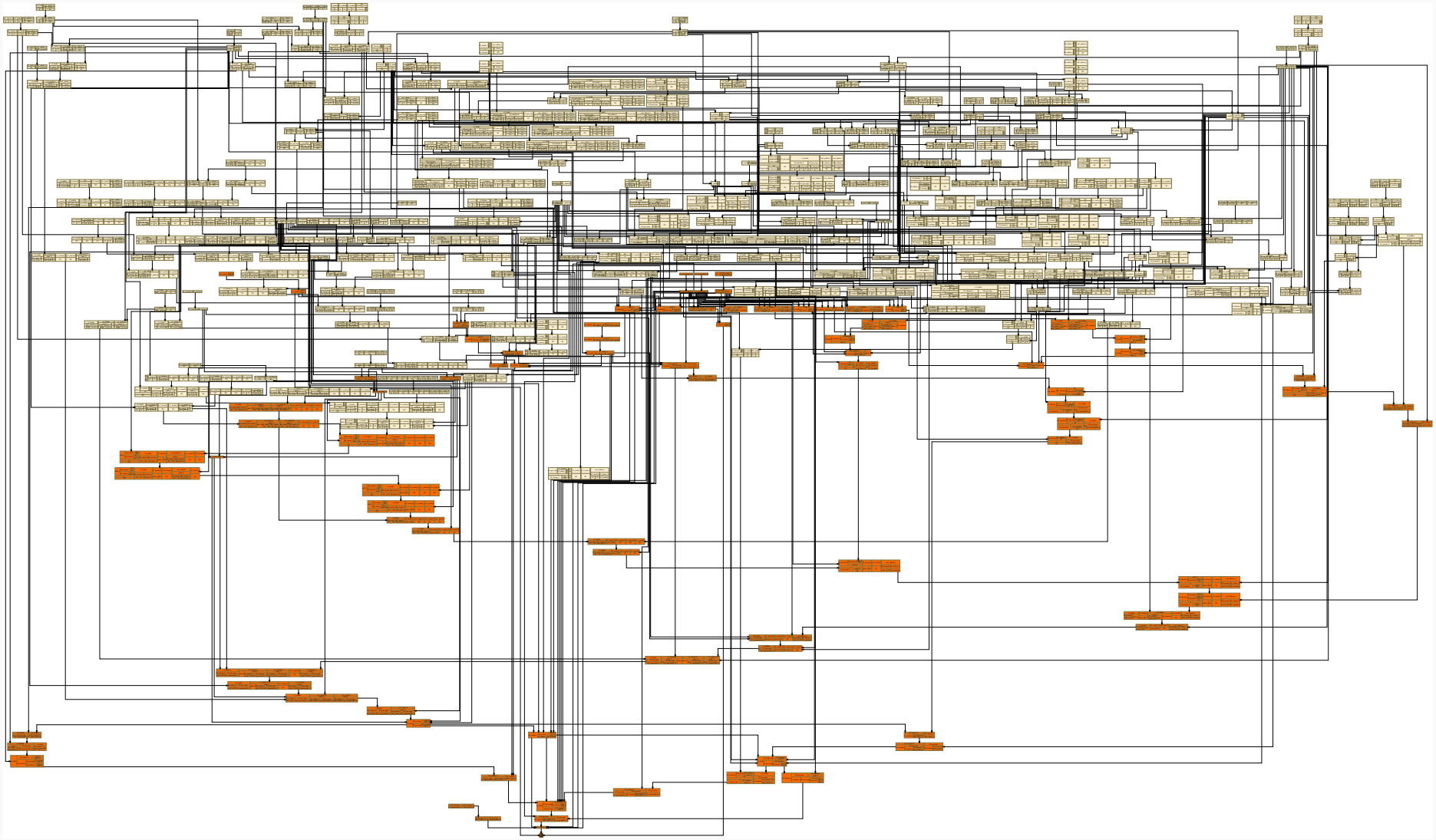
Feedback prove/learn loop for ENIGMA on Mizar data

- 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

Can you do this in 4 minutes?



Can you do this in 4 minutes?

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

Can you do this in 4 minutes?

README.md

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

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

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

The Mizar proof takes 80 lines:

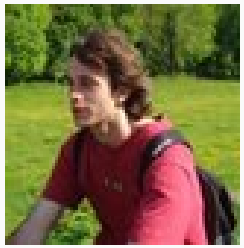
http://grid01.ciirc.cvut.cz/~mptp/7.13.01_4.181.1147/html/borsuk_5.html#T31

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

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

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

QSynt: Semantics-Aware Synthesis of Math Objects



- Gauthier'19-23
- 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:
- 2022: invention of programs for OEIS sequences from scratch
- 98k sequences (out of 350k) discovered so far:
  `https://www.youtube.com/watch?v=24oejR9wsXs`,
  `http://grid01.ciirc.cvut.cz/~thibault/qsynt.html`
- Many conjectures invented: 20+ different characterizations of primes
- Non-neural (Turing complete) computing and semantics collaborates with the statistical/neural 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:

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

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

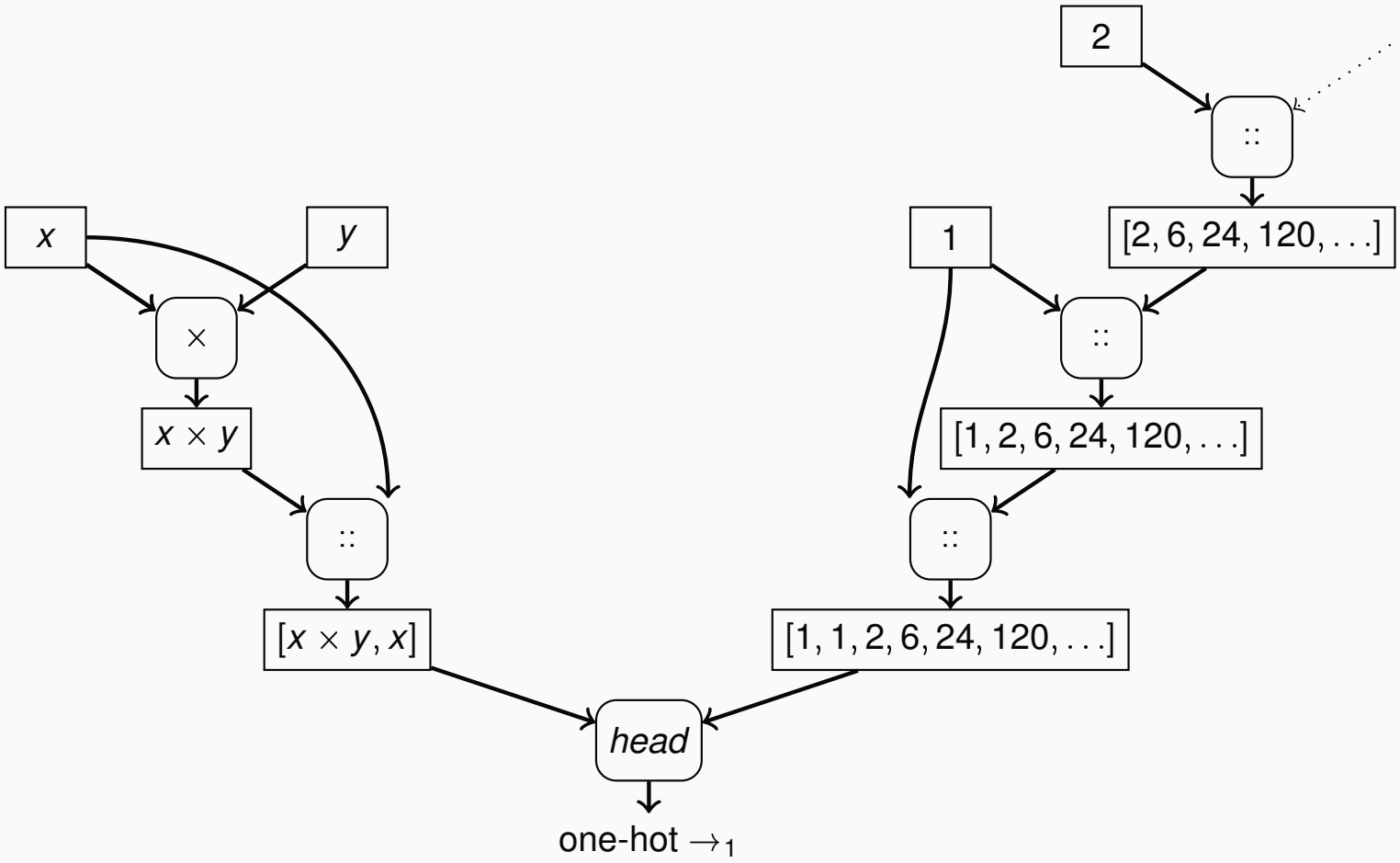
- Programs are built in reverse polish notation
- Start from an empty stack
- Use ML to repeatedly choose the next operator to push on top of a stack
- Example: Factorial is $loop(\lambda(x, y).\ x \times y, x, 1)$, built by:

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

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

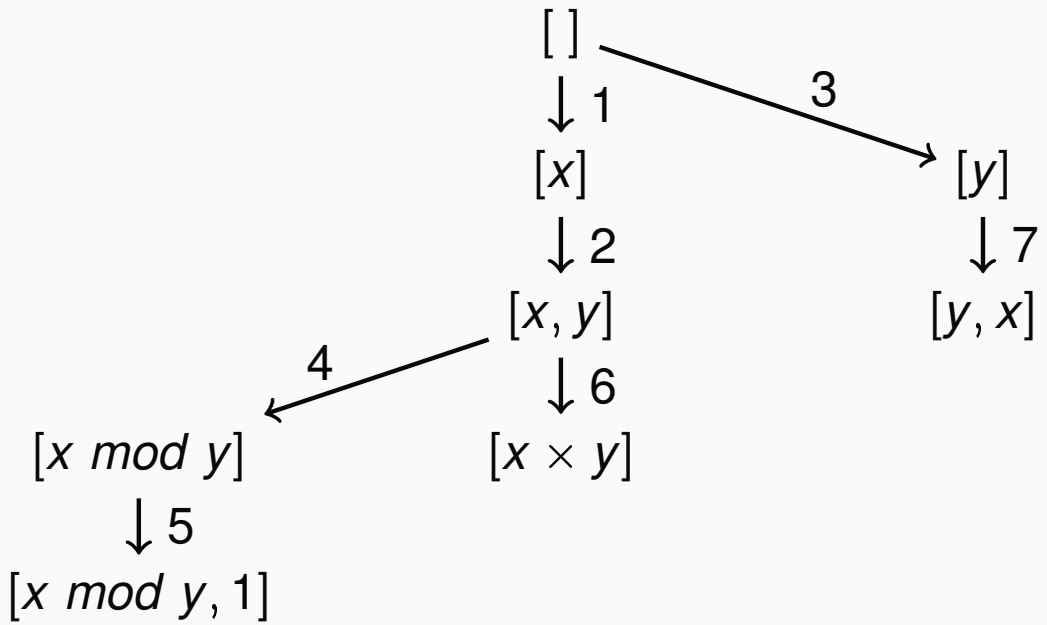
QSynt: Training of the Neural Net Guiding the Search

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

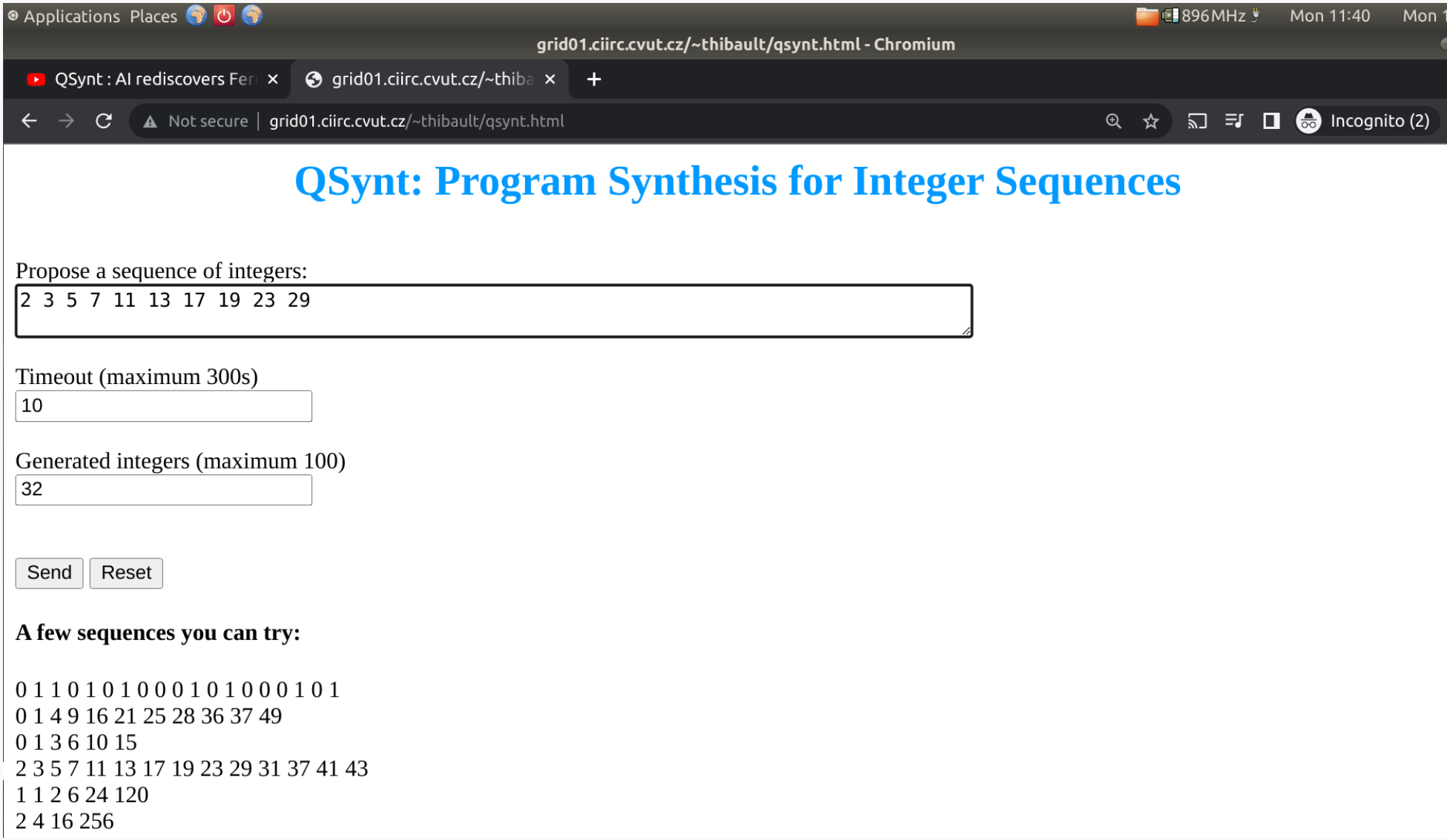


QSynt program search - Monte Carlo search tree

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



QSynt web interface for program invention



QSynt 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
```

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

```
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
```

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

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 (cf Mikolas Janota's talk):

```
(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)

Thanks and Acknowledgments

- CIIRC (V. Marik, O. Velek, many others) for welcoming us here
- EU, MSMT and OPVVV for funding us
- Our team members for great work and collaboration
- Our international partners
- Our great visitors, who typically like to return
- Very special thanks to people (both Czech and non-Czech) who came here from abroad and survived
- Further funding: Marie-Curie, NWO, ERC, Horizon, Amazon Research Awards