Learning to Reason (and Compute)

Josef Urban

Czech Technical University in Prague

December 11, 2021

European Research Council
Established by the European Commission

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!

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)

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?

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

small 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]);;
```

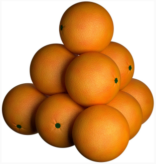
Irrationality of $\sqrt{2}$ in Isabelle/HOL

```
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.

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

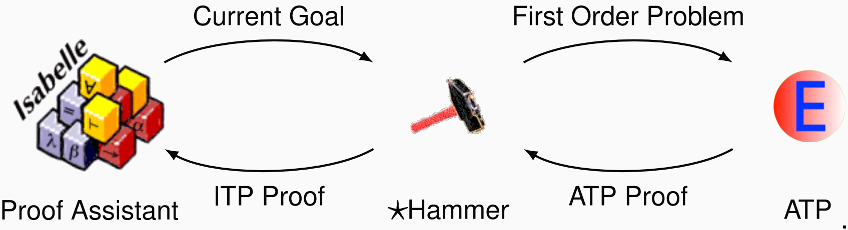


- Proved by Hales in 1998, 300-page proof + computations
- Big: Annals of Mathematics gave up reviewing after 4 years
- Formal proof finished in 2014
- 20000 lemmas in geometry, analysis, graph theory
- All of it at `https://code.google.com/p/flyspeck/`
- All of it computer-understandable and verified in HOL Light:
- `polyhedron s /\ c face_of s ==> polyhedron c`
- However, this took 20 – 30 person-years!
- our 2014 work: AI/TP combinations can *hammer* 40% of the 20k lemmas

AI and ML Combinations with 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
-

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



How much can it do?

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

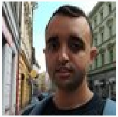
$\approx$ 40-45% success by 2016, 60% on Mizar as of 2021

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`
- 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`

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)
  - logic-aware graph neural network (GNN) run on a GPU server
  - logic-based subsumption using fast indexing (discrimination trees)
- 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

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

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!)
- also for Isabelle (Nagashima), HOL Light (Google), Coq (Blaauwbroek)

More on Conjecturing and Synthesis in Math

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

A bit of history

- The topic goes back at least to Lenat (AM) and Fajtlowicz (Graffiti)
- Combined with automated theorem proving by Colton et al. in early 2000s (HR)
- Theory exploration for Isabelle by Johansson et al (Hipster)
- Several learning-based/neural approaches by our groups since 2015:
- Based mainly on learning analogies and informalization followed by probabilistic/neural disambiguation
- Gauthier's deep RL-based synthesis toolkit in HOL:
  - Guiding synthesis of combinators for a given lambda expression
  - Guiding synthesis of a diophantine equation characterizing a given set
  - Guiding synthesis of programs describing integer sequences (OEIS)

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

```
Applications  Places                                   4,71 GHz   Wed 15:02   Wed 15:02
emacs@dell
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 “proof” - typechecks!

A correct GPT conjecture that was too hard to prove

Original Mizar theorem stated for finite groups:

```
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
```

Kinyon and Stanovsky (algebraists) confirmed that this GPT generalization that avoids finiteness is valid:

```
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
```

Prover9 - Research-Level Open Conjectures

- Michal Kinyon, Bob Veroff and Prover9: quasigroup and loop theory
- the Abelian Inner Mappinngs (AIM) Conjecture (>10 year program)
- Strong AIM: Q is AIM implies Q/Nuc(Q) is abelian and Q/Z(Q) is a group
- The Weak AIM Conjecture positively resolved in August 2021
- Q is AIM implies Q is nilpotent of class at most 3.
- 20-200k long proofs by Prover9 assisting the humans
- Prover9 hints strategy (Bob Veroff): extract hints from easier proofs to guide more difficult proofs
- Human-guided exploration to get good hints (not really automated yet)
- Millions of hints collected, various algorithms for their selection for a particular conjecture
- Symbolic machine learning?

Neural Autoformalization (Wang et al., 2018)

- generate ca 1M Latex/Mizar (informal/formal) pairs
- train neural seq-to-seq translation models (Luong – NMT)
- evaluate on about 100k examples
- many architectures tested, some work much better than others
- very important latest invention: *attention* in the seq-to-seq models
- more data very important for neural training – our biggest bottleneck
- Recent addition: unsupervised methods (Lample et all 2018) – no need for aligned data!

Neural Autoformalization data

Rendered $\LaTeX$	If $X \subseteq Y \subseteq Z$, then $X \subseteq Z$.
Mizar	`X c= Y & Y c= Z implies X c= Z;`
Tokenized Mizar	`X c= Y & Y c= Z implies X c= Z ;`
$\LaTeX$	`If $X \subseteq Y \subseteq Z$, then $X \subseteq Z$.`
Tokenized $\LaTeX$	`If $ X \subseteq Y \subseteq Z $ , then $ X \subseteq Z $ .`

Neural Fun – Performance after Some Training

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

Future: AITP Challenges/Bets

- Big challenge: Learn complicated symbolic algorithms (not black box)
- 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)
  - 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`

Computing and Logic: Curry Howard vs Others

- Disclaimer: you may want to discuss this with more formalizers
- But my impression is that Curry-Howard is a nice analogy ...
- ... which is however not much used in ITP practice
- Because extraction of *efficient* programs from proofs is hard
- Eg: constructive proof of the fundamental theorem of algebra in Coq
- Verified code is typically produced by other mechanisms:
- Extraction of definitions/lemmas as Haskell/ML from Isabelle - Flyspeck
- Verified machine code in HOL Light, CakeML, CompCert, seL4, ...
- Coq: division into efficient automated term normalization (*"computing"*) vs slow/manual general *"reasoning"* (Barendregt?)

Computing and Logic: Logic Programming

- Kowalski (2014): *"The driving force behind logic programming is the idea that a single formalism suffices for both logic and computation, and that logic subsumes computation."*
- *"Logic programs [..] combine logic and control, but make it possible to read the same program both logically and procedurally."*
- *"I later expressed this as Algorithm = Logic + Control (A = L + C) [Kowalski, 1979a], influenced by Pat Hayes' [1973] Computation = Controlled Deduction."*

Computing and Logic: Logic Programming

- Kowalski (2014):
- *“logic programming can also be understood more generally, for example, to include negation by failure , set construction, or goal-directed reasoning with equations”*
- Hayes: inference with equations imitates computation in Lisp
- *“LP excludes, for example, systems of constructive logic in which proofs are interpreted as programs,...”*
- Today: Clausal ATPs over Mizar seem to begin to learn some computational tasks
- Numerical calculations, boolean algebra, differentiation/integration, matrix operations, algebraic rewriting, etc

Automated Large-Theory Logic Programming?

1. a growing *computational/reasoning universe* of millions of verified mathematical (Prolog-style?) concepts/facts/rules
2. many queries: easier/harder mathematical problems including computation and reasoning
3. queries/conjectures are continuously asked and attempted by (continuously) trained AI/TP algorithms
4. The AI/TP systems select and combine the facts and rules automatically - a.k.a learning-guided theorem provers
5. The systems are continuously learning from their successes and failures, resulting in self-improvement and capability to attack more efficiently/deterministically harder computational problems
6. I.e., neither the manual programming of control as in Prolog, nor the unrestricted brute force search as in unguided ATPs
7. Instead: A framework for self-learned automated program control

Let’s Do Generalized (Fuzzy) Logic Programming

- Winograd [1971] *“Our heads don’t contain neat sets of logical axioms from which we can deduce everything through a ‘proof procedure’. Instead we have a large set of heuristics and procedures for solving problems at different levels of generality.”*
- Our 2021 Learning of (Fuzzy) Reasoning Components (Split & Merge):
  1. use a GNN to learn to identify interacting reasoning components based on many proofs
  2. use graph-based and clustering-based algorithms to split the sets of clauses into components,
  3. run saturation ATPs on the components,
  4. use premise selection to merge the component results, and
  5. iterate the procedure.

Further Notes For Discussion

- ML and ILP are further examples of (non-human) algorithm construction
- Learning-based symbolic program synthesis - various ways - ILP, transformers, symbolic regression, combined methods
- Similar to our learning-based proof construction methods (overlapping in the case of logic programming)
- Cf. Turing's last chapter on learning machines
- Algorithms are formally defined and often even executable in various ways in today's large formalizations: Isabelle, HOL, Coq, Mizar.
- Example: John Harrison's formalization and machine code for elliptic curve cryptography.
- Ownership of mathematical statements - the Proofgold blockchain (Chad Brown - Kuratowski, bounties). Can be similarly done for proofs and algorithms (smart contracts?) in blockchains. Alternative to law?
- Our 2011 MML licencing paper and its connection to today's systems like Copilot: What if you train reasoning/computing systems on large math/code corpora? How much is even Google search legal (it extracts knowledge/algorithms from Wikipedia, etc.)