Machine Learning in Automated and Interactive Theorem Proving

Josef Urban

Czech Technical University in Prague

SAT, July 11th, 2019

European Research Council

Established by the European Commission

Outline

Motivation, Learning vs. Reasoning

Computer Understandable (Formal) Math

Learning of Theorem Proving

Demos

High-level Reasoning Guidance: Premise Selection

Low Level Guidance of Theorem Provers

Mid-level Reasoning Guidance

Autoformalization

Motivation: Learning vs. Reasoning

“C’est par la logique qu’on démontre, c’est par l’intuition qu’on invente.”
(It is by logic that we prove, but by intuition that we discover.)
Henri Poincaré, Mathematical Definitions and Education.

“Hypothesen sind Netze; nur der fängt, wer auswirft.”
(Hypotheses are nets: only he who casts will catch.)
Novalis, quoted by Popper – The Logic of Scientific Discovery

How Do We Automate Math and Science?

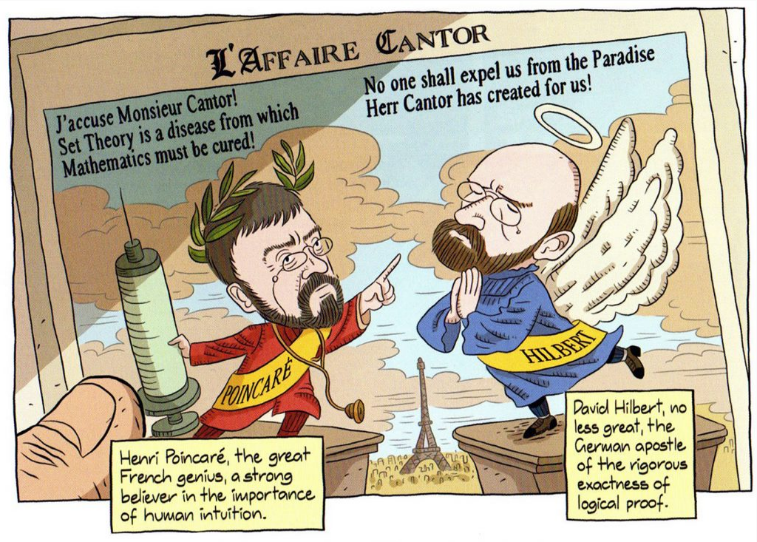
- What is mathematical and scientific thinking?
- Pattern-matching, analogy, induction from examples
- Deductive reasoning
- Complicated feedback loops between induction and deduction
- Using a lot of previous knowledge - both for induction and deduction

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

History, Motivation, AI/TP/ML/DL

- Intuition vs Formal Reasoning – Poincaré vs Hilbert, Science & Method
- Turing's 1950 paper: Learning Machines, learn Chess?, undecidability??
- Lenat, Langley, etc: manually-written heuristics, learn Kepler laws,...
- Denzinger, Schulz, Goller, Fuchs – late 90's, ATP-focused:
- *Learning from Previous Proof Experience*
- My MSc (1998): Try ILP to learn rules and heuristics from IMPS/Mizar
- Since: Use large formal math (Big Proof) corpora: Mizar, Isabelle, HOL
- ... to combine/develop symbolic/statistical deductive/inductive ML/TP/AI
- ... hammer-style methods, feedback loops, internal guidance, ...
- AI vs DL?: Ben Goertzel's 2018 Prague talk:
  `https://youtu.be/Zt2HSTuGBn8`

Intuition vs Formal Reasoning – Poincaré vs Hilbert

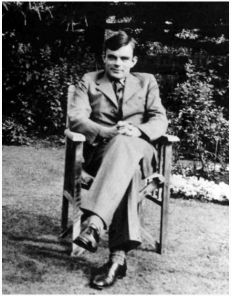


[Adapted from: *Logicomix: An Epic Search for Truth* by A. Doxiadis]

Induction/Learning vs Reasoning – Henri Poincaré

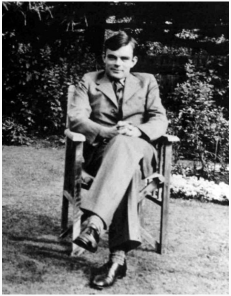
- Science and Method: Ideas about the interplay between correct deduction and induction/intuition
- *"And in demonstration itself logic is not all. The true mathematical reasoning is a real induction [...]"*
- I believe he was right: strong general reasoning engines have to combine deduction and induction (learning patterns from data, making conjectures, etc.)

Learning vs Reasoning – Alan Turing 1950 – AI



- 1950: *Computing machinery and intelligence* – AI, Turing test
- *"We may hope that machines will eventually compete with men in all purely intellectual fields."* (regardless of his 1936 undecidability result!)
- last section on Learning Machines:
- *"But which are the best ones [fields] to start [learning on] with?"*
- *"... Even this is a difficult decision. Many people think that a very abstract activity, like the playing of chess, would be best."*
- Why not try with math? It is much more (universally?) expressive ...

Induction/Learning vs Reasoning – Turing 1950 – AI



- 1950: *Computing machinery and intelligence* – AI, Turing test
- On pure deduction: *“For at each stage when one is using a logical system, there is a very large number of alternative steps, any of which one is permitted to apply, so far as obedience to the rules of the logical system is concerned. These choices make the difference between a brilliant and a footling reasoner, not the difference between a sound and a fallacious one.”*

Why Combine Learning and Reasoning Today?

1. It practically helps!
   - Automated theorem proving for large formal verification is useful:
     - Formal Proof of the Kepler Conjecture (2014 – Hales – 20k lemmas)
     - Formal Proof of the Feit-Thompson Theorem (2012 – Gonthier)
     - Verification of compilers (CompCert) and microkernels (seL4)
     - ...
   - But good learning/AI methods needed to cope with large theories!
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:
     - What are the components (inductive/deductive thinking)?
     - How to combine them together?

The Plan

1. Make large “formal thought” (Mizar/MML, Isabelle/HOL/AFP, HOL/Flyspeck ...) accessible to strong reasoning and learning AI tools – DONE (or well under way)
2. Test/Use/Evolve existing AI and ATP tools on such large corpora
3. Build custom/combined inductive/deductive tools/metasystems
4. Continuously test performance, define harder AI tasks as the performance grows

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
- 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, compiled by F. Wiedijk:

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 $(a, b) = 1$. $\square$

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
```

Irrationality of $\sqrt{2}$ in Otter

Problem

```
set(auto).
set(ur_res).
assign(max_distinct_vars, 1).
list(usable).
x = x.
m(1,x) = x.  %identity
m(x,1) = x.
m(x,m(y,z)) = m(m(x,y),z).   %assoc
m(x,y) = m(y,x).             %comm
m(x,y) != m(x,z) | y = z.   %cancel
-d(x,y) | m(x,f(x,y)) = y.  %divides
m(x,z) != y | d(x,y).
-d(2,m(x,y)) | d(2,x) | d(2,y). %2 prime
m(a,a) = m(2,m(b,b)).  % a/b=sqrt(2)
-d(x,a) | -d(x,b) | x = 1. % a/b lowest
2 != 1.
end_of_list.
```

Proof

```
1 [] m(x,y)!=m(x,z)|y=z.
2 [] -d(x,y)|m(x,f(x,y))=y.
3 [] m(x,y)!=z|d(x,z).
4 [] -d(2,m(x,y))|d(2,x)|d(2,y).
5 [] -d(x,a)| -d(x,b)|x=1.
6 [] 2!=1.
7 [factor,4.2.3] -d(2,m(x,x))|d(2,x).
13 [] m(x,m(y,z))=m(m(x,y),z).
14 [copy,13,flip.1] m(m(x,y),z)=m(x,m(y,
16 [] m(x,y)=m(y,x).
17 [] m(a,a)=m(2,m(b,b)).
18 [copy,17,flip.1] m(2,m(b,b))=m(a,a).
30 [hyper,18,3] d(2,m(a,a)).
39 [para_from,18.1.1,1.1.1] m(a,a)!=m(2,
42 [hyper,30,7] d(2,a).
46 [hyper,42,2] m(2,f(2,a))=a.
48 [ur,42,5,6] -d(2,b).
50 [ur,48,7] -d(2,m(b,b)).
59 [ur,50,3] m(2,x)!=m(b,b).
60 [copy,59,flip.1] m(b,b)!=m(2,x).
145 [para_from,46.1.1,14.1.1.1,flip.1] m
189 [ur,60,39] m(a,a)!=m(2,m(2,x)).
190 [copy,189,flip.1] m(2,m(2,x))!=m(a,a
1261 [para_into,145.1.1.2,16.1.1] m(2,m(
1272 [para_from,145.1.1,190.1.1.2] m(2,m
1273 [binary,1272.1,1261.1] $F.
```

Today: Computers Checking Large Math Proofs

SCI NEWS

Top Headlines:
Researchers Find 40,000-Year-Old Figurative Paintings in Bornean Cave

HOME ASTRONOMY SPACE EXPLORATION ARCHAEOLOGY PALEONTOLOGY BIOLOGY PHYSICS MEDICINE GENETICS GEOLOGY MORE

Scientists Deliver Formal Proof of Famous Kepler Conjecture

Jun 16, 2017 by News Staff / Source « Previous | Next »

Published in
Mathematics

Tagged as
Johannes Kepler
Kepler conjecture

Follow
You Might Like

Researchers Develop First-Ever 3D Numerical Model of Melting Snowflake

Researchers Develop Mathematical Model for How Innovations Emerge

An international team of mathematicians led by University of Pittsburgh Professor Thomas Hales has delivered a formal proof of the Kepler conjecture, a famous problem in discrete geometry. The team's paper is published in the journal *Forum of Mathematics, Pi*.

LATEST NEWS

SPHERE Captures Young Exoplanet Beta Pictoris b Orbiting around Its Star
Nov 13, 2018 | Astronomy

Mirarce eatoni: Newly-Discovered Cretaceous Bird Lived Among Dinosaurs, Was Strong Flier
Nov 13, 2018 | Paleontology

Juno Takes Closer Look at Jupiter's Magnificent, Swirling Clouds
Nov 13, 2018 | Space Exploration

Physicists Solve Structure of Unusually Complex Form of Nitrogen
Nov 13, 2018 | Physical Chemistry

Natural Compound Protects Hypertensive Rats against Heart Disease
Nov 13, 2018 | Medicine

Inventive Orangutans Make Hook Tools to Retrieve Food
Nov 12, 2018 | Biology

Researchers Find 40,000-Year-Old Figurative Paintings in Bornean Cave
Nov 12, 2018 | Archaeology

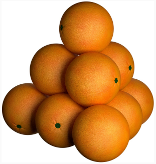
Hubble Sees Lensing Galaxy Cluster,

cdn.sci-news.com/images/enlarge3/image_4960e-Kepler-Conjecture.jpg

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)
- Consistency of the Negation of CH – Jesse Han and Floris van Doorn (Lean)

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

What Are Automated Theorem Provers?

- Computer programs that (try to) determine if
  - A conjecture C is a logical consequence of a set of axioms Ax
  - The derivation of conclusions that follow inevitably from facts.
- Systems: Vampire, E, SPASS, Prover9, Z3, CVC4, Satallax, iProver, ...
- Brute-force search calculi (resolution, superposition, tableaux, SMT, ...)
- Human-designed heuristics for pruning of the search space
- Fast combinatorial explosion on large knowledge bases like Flyspeck and Mizar
- Need to be equipped with good domain-specific inference guidance ...
- ... and that is what we try to do ...
- ... typically by learning in various ways from over knowledge bases ...

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
- ...

Large Datasets

- Mizar / MML / MPTP – since 2003
- MPTP Challenge (2006), MPTP2078 (2011), Mizar40 (2013)
- Isabelle (and AFP) – since 2005
- Flyspeck (including core HOL Light and Multivariate) – since 2012
- HOL4 – since 2014, CakeML – 2017, GRUNGE – 2019
- Coq – since 2013/2016
- ACL2 – 2014?
- Lean?, Stacks?, Arxiv?, ProofWiki?, ...

Demos

- Hammering Mizar: `http://grid01.ciirc.cvut.cz/~mptp/out4.ogv`
- TacticToe on HOL4: `http://grid01.ciirc.cvut.cz/~mptp/tactictoe_demo.ogv`
- Inf2formal over HOL Light: `http://grid01.ciirc.cvut.cz/~mptp/demo.ogv`
- TacticToe longer: `https://www.youtube.com/watch?v=BO4Y8ynwT6Y`

High-level ATP guidance: Premise Selection

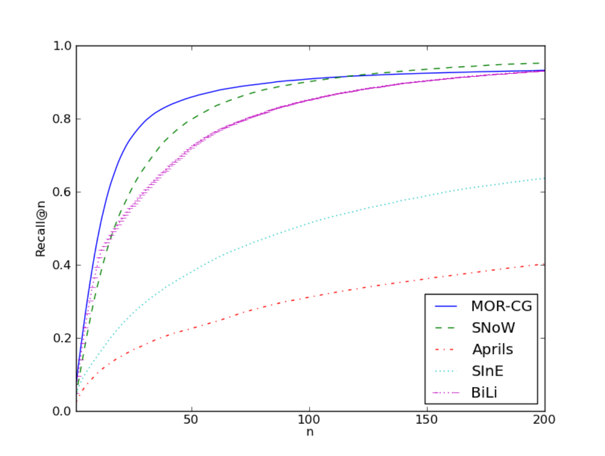
- Early 2003: Can existing ATPs be used over the freshly translated Mizar library?
- About 80000 nontrivial math facts at that time – impossible to use them all
- Is good premise selection for proving a new conjecture possible at all?
- Or is it a mysterious power of mathematicians? (Penrose)
- Today: Premise selection is not a mysterious property of mathematicians!
- Reasonably good algorithms started to appear (more below).
- Will extensive human (math) knowledge get obsolete?? (cf. Watson, Debater, etc)

Example system: Mizar Proof Advisor (2003)

- train naive-Bayes fact selection on all previous Mizar/MML proofs (50k)
- input features: conjecture symbols; output labels: names of facts
- recommend relevant facts when proving new conjectures
- give them to unmodified FOL ATPs
- possibly reconstruct inside the ITP afterwards (lots of work)
- First results over the whole Mizar library in 2003:
  - about 70% coverage in the first 100 recommended premises
  - chain the recommendations with strong ATPs to get full proofs
  - about 14% of the Mizar theorems were then automatically provable (SPASS)
- Today's methods: about 45-50% (and we are still just beginning!)

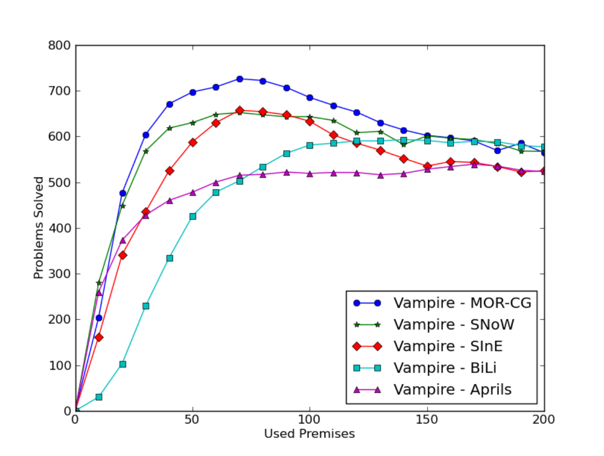
ML Evaluation of methods on MPTP2078 – recall

- Coverage (recall) of facts needed for the Mizar proof in first *n* predictions
- MOR-CG – kernel-based, SNoW - naive Bayes, BiLi - bilinear ranker
- SINe, Aprils - heuristic (non-learning) fact selectors

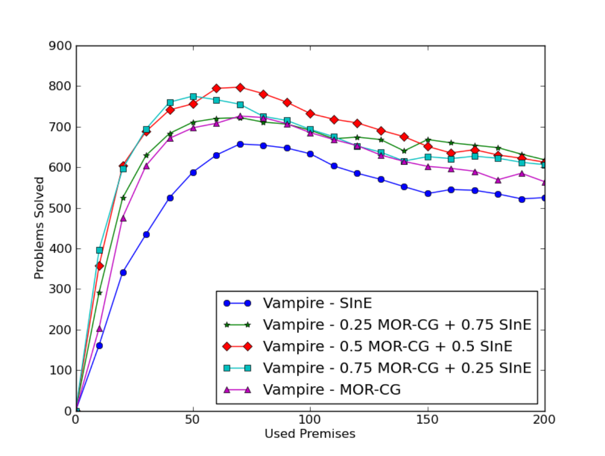


ATP Evaluation of methods on MPTP2078

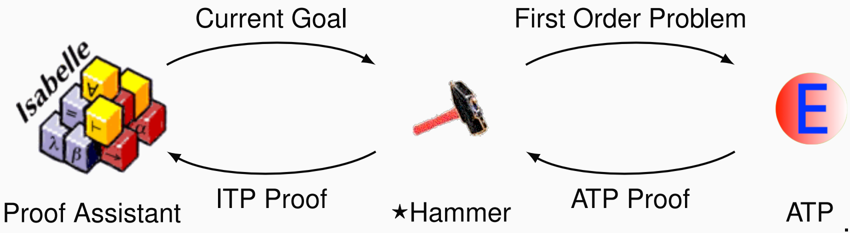
- Number of the problems proved by ATP when given *n* best-ranked facts
- Good machine learning on previous proofs really matters for ATP!



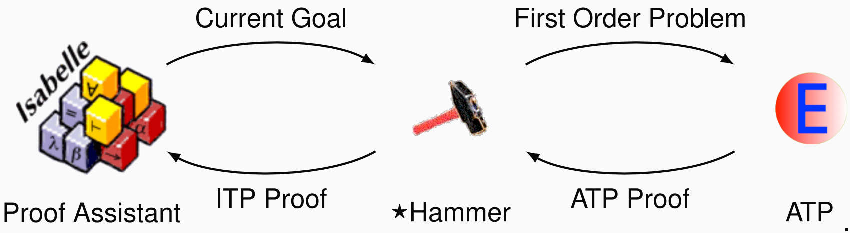
Combined (ensemble) methods on MPTP2078



Today's AI-ATP systems (⋆-Hammers)

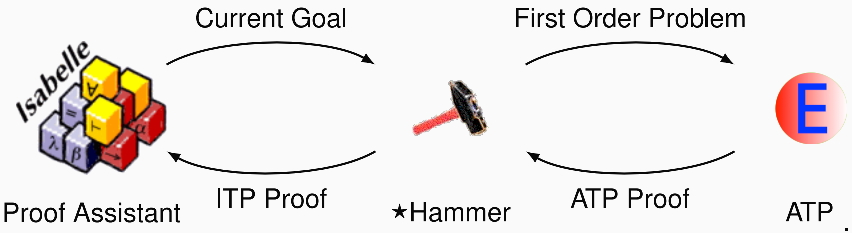


Today's AI-ATP systems (⋆-Hammers)



How much can it do?

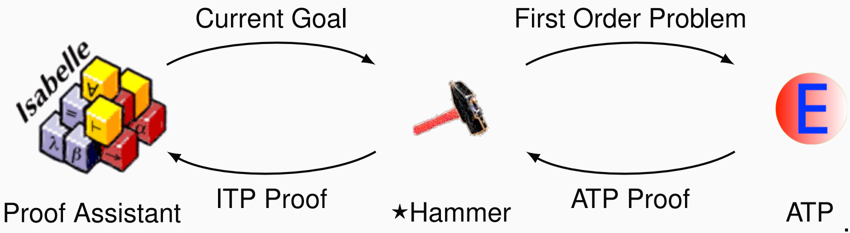
Today's AI-ATP systems (⋆-Hammers)



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

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

$\approx$ 45% success rate

Various Improvements and Additions

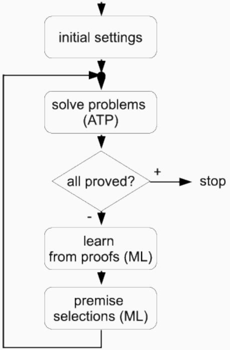
- Semantic features encoding term matching/unification [IJCAI'15]
- Distance-weighted k-nearest neighbor, LSI, boosted trees (XGBoost)
- Matching and transferring concepts and theorems between libraries (Gauthier & Kaliszyk) – allows “superhammers”, conjecturing, and more
- Lemmatization – extracting and considering millions of low-level lemmas
- First useful CoqHammer (Czajka & Kaliszyk 2016), 40%–50% reconstruction/ATP success on the Coq standard library
- Neural models, definitional embeddings (with Google, more groups today)
- Hammers combined with Monte-Carlo tactical search: TacticToe (Gauthier 2017)
- Learning in binary setting from many alternative proofs
- Negative/positive mining - ATPBoost (Piotrowski & JU, IJCAR'18)

Summary of Features Used

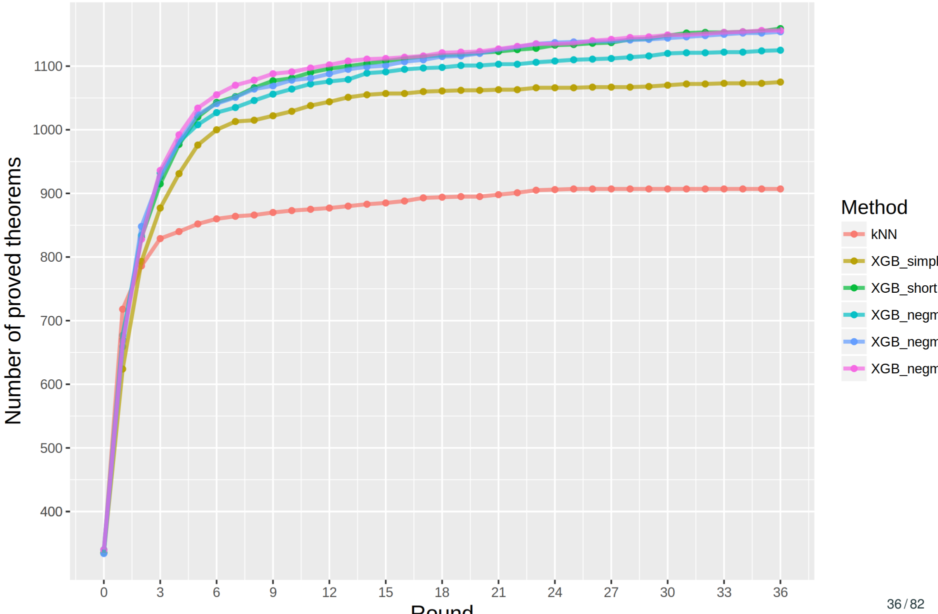
- From syntactic to more semantic:
- Constant and function symbols
- Walks in the term/formula graph
- Walks in clauses with polarity and variables/skolems unified
- Subterms, de Bruijn normalized
- Subterms, all variables unified
- Matching terms, no generalizations
- terms and (some of) their generalizations
- Substitution tree nodes
- All unifying terms
- LSI/PCA, word2vec, fasttext, etc.
- Neural embeddings: CNN, RNN, Tree NN, Graph CNN, ...
- Evaluation in a large set of (finite) models
- Vectors of proof similarities (proof search hidden states)
- Vectors of problems solved (for ATP strategies)

High-level feedback loops – MaLARea

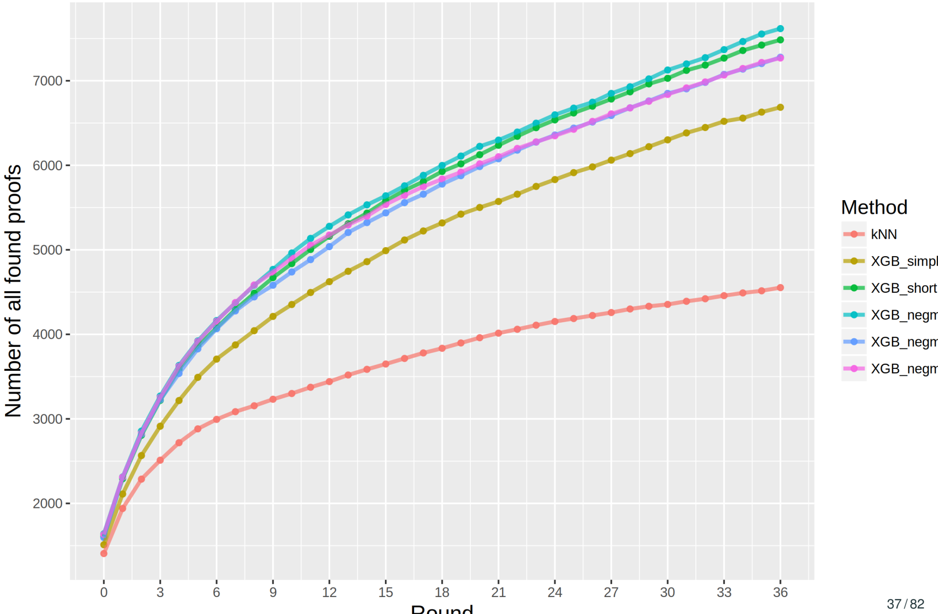
- Machine Learner for Autom. Reasoning (2006) – *infinite hammering*
- feedback loop interleaving ATP with learning premise selection
- both syntactic and semantic features for characterizing formulas:
- evolving set of finite (counter)models in which formulas evaluated
- winning AI/ATP benchmarks (MPTPChallenge,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

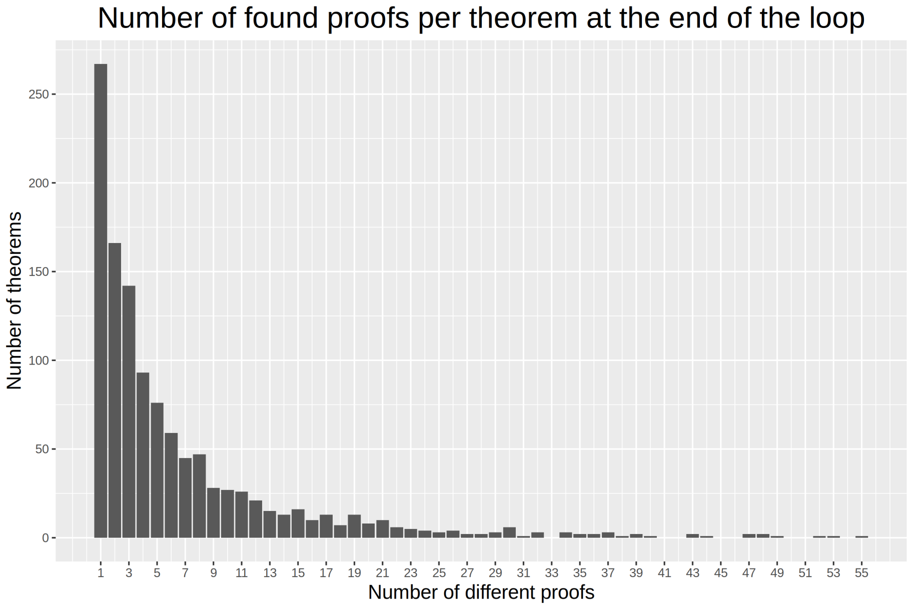


Prove-and-learn loop on MPTP2078 data set



Prove-and-learn loop on MPTP2078 data set




Number of found proofs per theorem at the end of the loop
Number of theorems
Number of different proofs
0
50
100
150
200
250
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

Low-level: Statistical Guidance of Connection Tableau

- learn guidance of every clausal inference in connection tableau (leanCoP)
- *tabula rasa* ATP: 20 lines of Prolog, but complete for FOL/math!
- set of first-order clauses, *extension* and *reduction* steps
- proof finished when all branches are closed
- a lot of nondeterminism, requires backtracking
- *Iterative deepening* used in leanCoP to ensure completeness
- good for learning – the tableau compactly represents the proof state

Clauses:

$c_1 : P(x)$

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

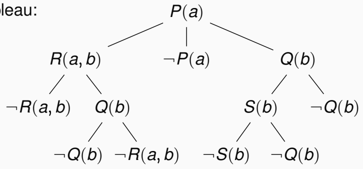
$c_3 : S(x) \vee \neg Q(b)$

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

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

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

Closed Connection Tableau:



Statistical Guidance of Connection Tableau

- **MaLeCoP** (2011): first prototype Machine Learning Connection Prover
- extension rules chosen by naive Bayes trained on good decisions
- training examples: tableau features plus the name of the chosen clause
- initially slow: off-the-shelf learner 1000 times slower than raw leanCoP
- 20-time search shortening on the MPTP Challenge
- second version: 2015, with C. Kaliszyk
- both prover and naive Bayes in OCAML, fast indexing
- Fairly Efficient MaLeCoP = **FEMaLeCoP**
- 15% improvement over untrained leanCoP on the MPTP2078 problems
- using iterative deepening - enumerate shorter proofs before longer ones

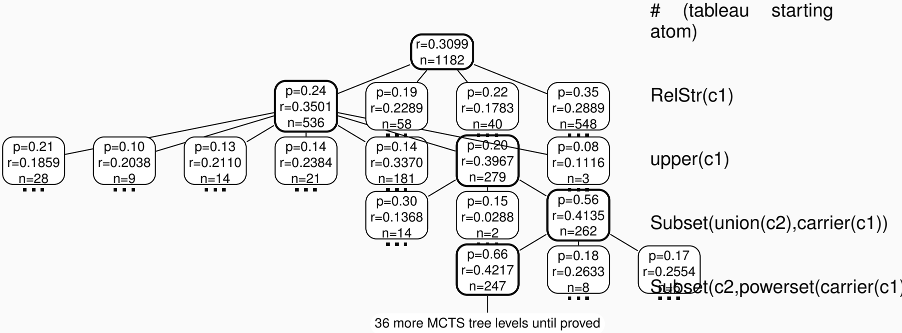
Statistical Guidance of Connection Tableau – rlCoP

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

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

- learning both *policy* (clause selection) and *value* (state evaluation)
- 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 (XGBoost)

MCTS search tree example



Statistical Guidance of Connection Tableau – rlCoP

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

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

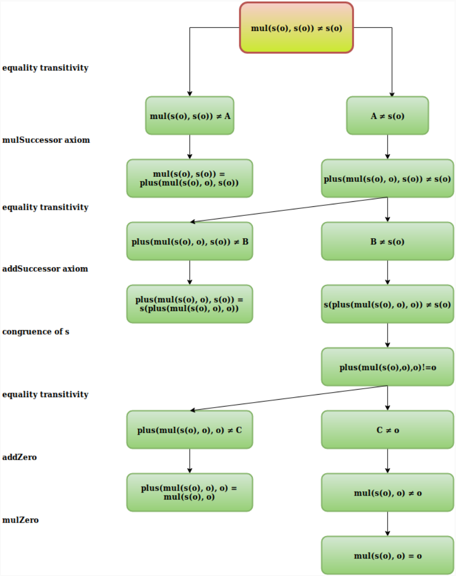
- rlCoP with policy/value after 5 proving/learning iters on the training data
- $1624/1143 = 42.1$% improvement over leanCoP on the testing problems

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

Recent Variations – FLoP, RNN, GCN

- FLoP – Finding Longer Proofs (Zsombori et al, 2019)
- Curriculum Learning used for connection tableau over Robinson Arithmetic
- addition and multiplication learned perfectly from $1 * 1 = 1$
- headed towards learning algorithms/decision procedures from math data
- currently black-box, combinations with symbolic methods (ILP) our next target
- Using RNNs for better tableau encoding, prediction of actions ...
- ... even guessing (decoding) next tableau literals (Piotrowski & JU, 2019)
- The same with graph convolutional nets (GCN) - encouraging preliminary results

FLoP Training Proof



Side Note on Symbolic Learning with NNs

- Recurrent NNs with attention recently very good at the inf2formal task
- Experiments with using them for symbolic rewriting (Piotrowski et. al)
- We can learn rewrite rules from sufficiently many data
- 80-90% on algebra datasets, 70-99% on normalizing polynomials
- again, complements symbolic methods like ILP that suffer if too much data
- Similar use for conjecturing (Chvalovsky et al):
- Learn *consistent translations* between different math contexts:
- additive groups → multiplicative groups

Side Note on Symbolic Learning with NNs

Table: Examples in the AIM data set.

Rewrite rule:	Before rewriting:	After rewriting:
`b(s(e,v1),e)=v1`	`k(b(s(e,v1),e),v0)`	`k(v1,v0)`
`o(V0,e)=V0`	`t(v0,o(v1,o(v2,e)))`	`t(v0,o(v1,v2))`

Table: Examples in the polynomial data set.

Before rewriting:	After rewriting:
`(x * (x + 1)) + 1`	`x ^ 2 + x + 1`
`(2 * y) + 1 + (y * y)`	`y ^ 2 + 2 * y + 1`
`(x + 2) * ((2 * x) + 1) + (y + 1)`	`2 * x ^ 2 + 5 * x + y + 3`

Side Note on Conjecturing with RNNs

We can obtain a new valid automatically provable lemma

$(X \cap Y) \setminus Z = (X \setminus Z) \cap (Y \setminus Z)$

from

$(X \cup Y) \setminus Z = (X \setminus Z) \cup (Y \setminus Z)$

Examples of false but syntactically consistent conjectures:

```
for n, m being natural numbers holds n gcd m = n div m;

for R being Relation holds
with_suprema(A) <=> with_suprema(inverse_relation(A));
```

Side Note on Model Learning with NNs

- Smolik 2019 (MSc thesis): modelling mathematical structures with NNs
- NNs reasonably learn cyclic groups and their extensions
- ... so far struggle in learning bigger permutation groups
- Plan: learn composition/variation of complicated math structures
- Use for neural model-style evaluation of formulas, conjectures, etc. ...
- ... similarly to the finite models in MaLARea, etc.

Guiding Saturation-Based Theorem Proving

Basic Saturation Loop – Given Clause Loop

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

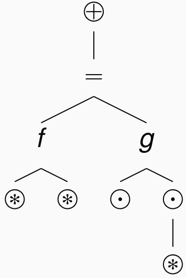
Typically, U grows quadratically wrt. P
1M clauses in U in 10s common – choosing good g gets harder

Statistical Guidance the Given Clause in E Prover

- harder for learning than tableau
- the proof state are two large heaps of clauses *processed*/*unprocessed*
- 2017: ENIGMA - fast exhaustive features (Jakubuv & JU 2017)
- 2017: Deep guidance (neural nets) (Loos et al. 2017)
- both learn on E's proof search traces, put classifier in E
- positive examples: given clauses used in the proof
- negative examples: given clauses not used in the proof
- ENIGMA: fast feature extraction followed by fast/sparse linear classifier
- about 80% improvement on an AIM benchmark
- Deep guidance: convolutional nets - very slow
- ENIGMA-NG: better features and ML, gradient-boosted trees, tree NNs
- NNs made competitive in real-time, boosted trees still best

Clauses as Feature Vectors for ENIGMA

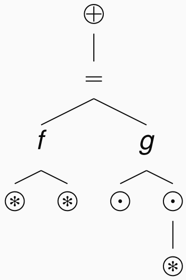
Collect and enumerate all the features. Count the clause features.



#	feature	count
1	(⊕,=,a)	0
⋮	⋮	⋮
11	(⊕,=,f)	1
12	(⊕,=,g)	1
13	(=,f,⊛)	2
14	(=,g,⊙)	2
15	(g,⊙,⊛)	1
⋮	⋮	⋮

Clauses as Feature Vectors for ENIGMA

Take the counts as a feature vector.



#	feature	count
1	$(\oplus,=,a)$	0
⋮	⋮	⋮
11	$(\oplus,=,f)$	1
12	$(\oplus,=,g)$	1
13	$(=,f,\circledast)$	2
14	$(=,g,\odot)$	2
15	$(g,\odot,\circledast)$	1
⋮	⋮	⋮

Feedback loop for ENIGMA on Mizar data

- Similar to rlCoP - interleave proving and learning of ENIGMA guidance
- Done on 57880 Mizar problems very recently
- Ultimately a 70% improvement over the original strategy
- Example Mizar proof found by ENIGMA: `http://grid01.ciirc.cvut.cz/~mptp/7.13.01_4.181.1147/html/knaster#T21`
- Its E-ENIGMA proof:
  `http://grid01.ciirc.cvut.cz/~mptp/t21_knaster`

	$\mathcal{S}$	$\mathcal{S}\odot\mathcal{M}_9^0$	$\mathcal{S}\oplus\mathcal{M}_9^0$	$\mathcal{S}\odot\mathcal{M}_9^1$	$\mathcal{S}\oplus\mathcal{M}_9^1$	$\mathcal{S}\odot\mathcal{M}_9^2$	$\mathcal{S}\oplus\mathcal{M}_9^2$	$\mathcal{S}\odot\mathcal{M}_9^3$	$\mathcal{S}\oplus\mathcal{M}_9^3$
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 Proof Example – Knaster

```
theorem Th21:
  ex a st a is_a_fixpoint_of f
proof
  set H = {h where h is Element of L: h [= f.h};
  set fH = {f.h where h is Element of L: h [= f.h};
  set uH = "\/"(H, L);
  set fuH = "\/"(fH, L);
  take uH;
  now
    let fh be Element of L;
    assume fh in fH;
    then consider h being Element of L such that
A1: fh = f.h and
A2: h [= f.h;
    h in H by A2;
    then h [= uH by LATTICE3:38;
    hence fh [= f.uH by A1,QUANTAL1:def 12;
  end;
  then fH is_less_than f.uH by LATTICE3:def 17;
  then
A3: fuH [= f.uH by LATTICE3:def 21;
  now
    let a be Element of L;
    assume a in H;
    then consider h being Element of L such that
A4: a = h & h [= f.h;
    reconsider fh = f.h as Element of L;
    take fh;
    thus a [= fh & fh in fH by A4;
  end;
  then uH [= fuH by LATTICE3:47;
  then
A5: uH [= f.uH by A3,LATTICES:7;
  then f.uH [= f.(f.uH) by QUANTAL1:def 12;
  then f.uH in H;
  then f.uH [= uH by LATTICE3:38;
  hence uH = f.uH by A5,LATTICES:8;
end;
```

ProofWatch: Statistical/Semantic Guidance of E (Goertzel et al. 2018)

- Bob Veroff's *hints* method used for Prover9/AIM
- solve many easier/related problems
- load their useful lemmas on the *watchlist* (kind of conjecturing)
- boost inferences on clauses that subsume a watchlist clause
- watchlist parts are fast thinking, bridged by standard (slow) search
- ProofWatch (2018): load many proofs separately
- dynamically boost those that have been covered more
- needed for heterogeneous ITP libraries
- statistical: watchlists chosen using similarity and usefulness
- semantic/deductive: dynamic guidance based on exact proof matching
- results in better vectorial characterization of saturation proof searches

ProofWatch: Statistical/Symbolic Guidance of E

```
theorem Th36: :: YELLOW_5:36
for L being non empty Boolean RelStr for a, b being Element of L
holds ( 'not' (a "\/" b) = ('not' a) "/\" ('not' b)
         & 'not' (a "/\" b) = ('not' a) "\/" ('not' b) )
```

- De Morgan's laws for Boolean lattices
- guided by 32 related proofs resulting in 2220 watchlist clauses
- 5218 given clause loops, resulting ATP proof is 436 clauses
- 194 given clauses match the watchlist and 120 (61.8%) used in the proof
- most helped by the proof of WAYBEL_1:85 – done for lower-bounded Heyting

```
theorem :: WAYBEL_1:85
for H being non empty lower-bounded RelStr st H is Heyting holds
for a, b being Element of H holds
  'not' (a "/\" b) >= ('not' a) "\/" ('not' b)
```

ProofWatch: Vectorial Proof State

Final state of the proof progress for the 32 proofs guiding `YELLOW_5:36`

0	0.438	42/96	1	0.727	56/77	2	0.865	45/52	3	0.360	9/25
4	0.750	51/68	5	0.259	7/27	6	0.805	62/77	7	0.302	73/242
8	0.652	15/23	9	0.286	8/28	10	0.259	7/27	11	0.338	24/71
12	0.680	17/25	13	0.509	27/53	14	0.357	10/28	15	0.568	25/44
16	0.703	52/74	17	0.029	8/272	18	0.379	33/87	19	0.424	14/33
20	0.471	16/34	21	0.323	20/62	22	0.333	7/21	23	0.520	26/50
24	0.524	22/42	25	0.523	45/86	26	0.462	6/13	27	0.370	20/54
28	0.411	30/73	29	0.364	20/55	30	0.571	16/28	31	0.357	10/28

EnigmaWatch: ProofWatch used with ENIGMA

- Use the proof completion ratios as features for characterizing proof state
- Instead of just static conjecture features - the vectors evolve
- Feed them to ML systems along with other features
- Relatively good improvement
- To be extended in various ways

EnigmaWatch: ProofWatch used with ENIGMA

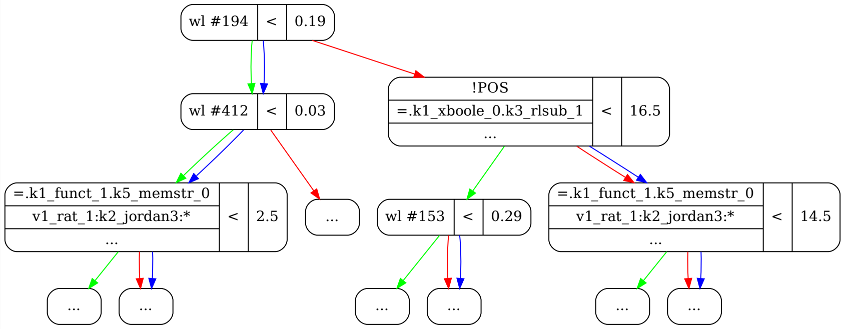
Baseline	Mean	Var	Corr	Rand	Baseline ∪ Mean	Total
1140	1357	1345	1337	1352	1416	1483

Table: ProofWatch evaluation: Problems solved by different versions.

loop	ENIGMA	Mean	Var	Corr	Rand	ENIGMA ∪ Mean	Total
0	1557	1694	1674	1665	1690	1830	1974
1	1776	1815	1812	1812	1847	1983	2131
2	1871	1902	1912	1882	1915	2058	2200
3	1931	1954	1946	1920	1926	2110	2227

Table: ENIGMAWatch evaluation: Problems solved and the effect of looping.

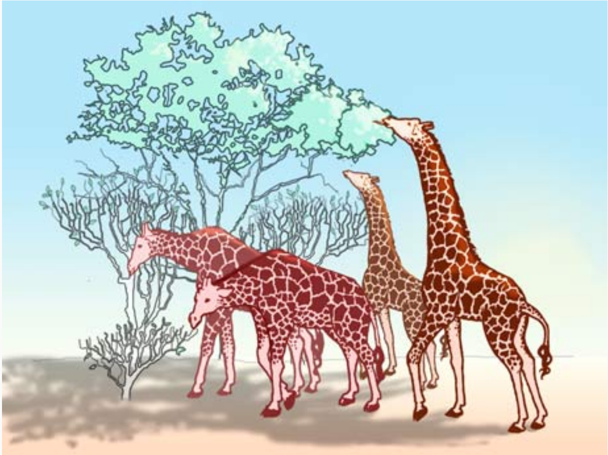
Example of an XGBoost decision tree



TacticToe: mid-level ITP Guidance (Gauthier et al.'18)

- learns from human tactical HOL4 proofs to solve new goals
- no translation or reconstruction needed
- similar to rlCoP: policy/value learning
- however much more technically challenging:
  - tactic and goal state recording
  - tactic argument abstraction
  - absolutization of tactic names
  - nontrivial evaluation issues
- 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)
- work in progress for Coq (us, OpenAI) and HOL Light (us, Google)

More Mid-level guidance: BliStr: Blind Strategymaker



- The ATP strategies are like giraffes, the problems are their food
- The better the giraffe specializes for eating problems unsolvable by others, the more it gets fed and further evolved

The E strategy with longest specification in Jan 2012

```
G-E--_029_K18_F1_PI_AE_SU_R4_CS_SP_S0Y:

4 * ConjectureGeneralSymbolWeight(
       SimulateSOS,100,100,100,50,50,10,50,1.5,1.5,1),
3 * ConjectureGeneralSymbolWeight(
       PreferNonGoals,200,100,200,50,50,1,100,1.5,1.5,1),
1 * Clauseweight(PreferProcessed,1,1,1),
1 * FIFOWeight(PreferProcessed)
```

BliStr: Blind Strategymaker

- Use clusters of similar solvable problems to train for unsolved problems
- Interleave low-time training on easy problems with high-time evaluation
- Single strategy evolution done by ParamILS - Iterated Local Search (genetic methods tried recently)
- Thus co-evolve the strategies and their training problems
- The hard problems gradually become easier and turn into training data (the trees get lower for a taller giraffe)
- In the end, learn which strategy to use on which problem

BliStr on 1000 Mizar@Turing problems

- original E coverage: 597 problems
- after 30 hours of strategy growing: 22 strategies covering 670 problems
- The best strategy solves 598 problems (1 more than all original strategies)
- A selection of 14 strategies improves E auto-mode by 25% on unseen problems
- Similar results for the Flyspeck problems
- Be lazy, don't do "hard" theory-driven ATP research (a.k.a: thinking)
- Larry Wall (Programming Perl): *"We will encourage you to develop the three great virtues of a programmer: laziness, impatience, and hubris"*

The E strategy with longest specification in May 2014

```
atpstr_my_c7bb78cc4c665670e6b866a847165cb4bf997f8a:

6 * ConjectureGeneralSymbolWeight(PreferNonGoals,100,100,100,50,50,1000,100,1.5,1.5,1)
8 * ConjectureGeneralSymbolWeight(PreferNonGoals,200,100,200,50,50,1,100,1.5,1.5,1)
8 * ConjectureGeneralSymbolWeight(SimulateSOS,100,100,100,50,50,50,50,1.5,1.5,1)
4 * ConjectureRelativeSymbolWeight(ConstPrio,0.1, 100, 100, 100, 100, 1.5, 1.5, 1.5)
10 * ConjectureRelativeSymbolWeight(PreferNonGoals,0.5, 100, 100, 100, 100, 1.5, 1.5, 1)
2 * ConjectureRelativeSymbolWeight(SimulateSOS,0.5, 100, 100, 100, 100, 1.5, 1.5, 1)
10 * ConjectureSymbolWeight(ConstPrio,10,10,5,5,5,1.5,1.5,1.5)
1 * Clauseweight(ByCreationDate,2,1,0.8)
1 * Clauseweight(ConstPrio,3,1,1)
6 * Clauseweight(ConstPrio,1,1,1)
2 * Clauseweight(PreferProcessed,1,1,1)
6 * FIFOWeight(ByNegLitDist)
1 * FIFOWeight(ConstPrio)
2 * FIFOWeight(SimulateSOS)
8 * OrientLMaxWeight(ConstPrio,2,1,2,1,1)
2 * PNRefinedweight(PreferGoals,1,1,1,2,2,2,0.5)
10 * RelevanceLevelWeight(ConstPrio,2,2,0,2,100,100,100,100,1.5,1.5,1)
8 * RelevanceLevelWeight2(PreferNonGoals,0,2,1,2,100,100,100,400,1.5,1.5,1)
2 * RelevanceLevelWeight2(PreferGoals,1,2,1,2,100,100,100,400,1.5,1.5,1)
6 * RelevanceLevelWeight2(SimulateSOS,0,2,1,2,100,100,100,400,1.5,1.5,1)
8 * RelevanceLevelWeight2(SimulateSOS,1,2,0,2,100,100,100,400,1.5,1.5,1)
5 * rweight21_g
3 * Refinedweight(PreferNonGoals,1,1,2,1.5,1.5)
1 * Refinedweight(PreferNonGoals,2,1,2,2,2)
2 * Refinedweight(PreferNonGoals,2,1,2,3,0.8)
8 * Refinedweight(PreferGoals,1,2,2,1,0.8)
10 * Refinedweight(PreferGroundGoals,2,1,2,1.0,1)
20 * Refinedweight(SimulateSOS,1,1,2,1.5,2)
1 * Refinedweight(SimulateSOS,3,2,2,1.5,2)
```

Autoformalization

- Goal: Learn understanding of informal math formulas and reasoning
- Experiments with the CYK chart parser linked to semantic methods
- Experiments with neural methods
- Semantic methods: Type checking, theorem proving
- Corpora: Flyspeck, Mizar, Proofwiki, etc.

Statistical/Semantic Parsing of Informalized HOL

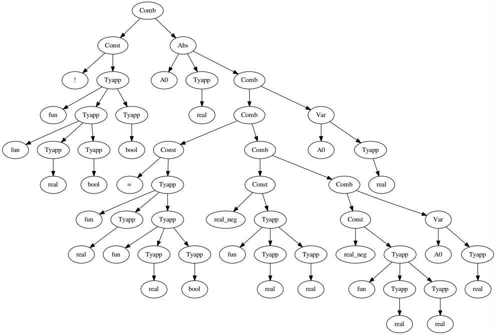
- Training and testing examples exported form Flyspeck formulas
  - Along with their informalized versions
- Grammar parse trees
  - Annotate each (nonterminal) symbol with its HOL type
  - Also “semantic (formal)” nonterminals annotate overloaded terminals
  - guiding analogy: word-sense disambiguation using CYK is common
- Terminals exactly compose the textual form, for example:
- REAL_NEGNEG: $\forall x. - -x = x$

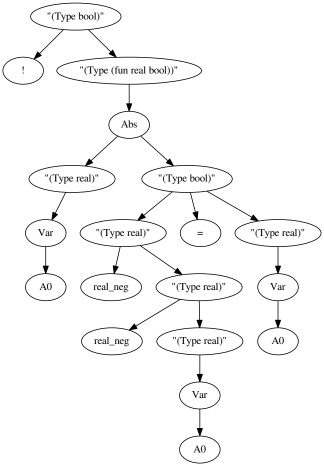
```
(Comb (Const "!" (Tyapp "fun" (Tyapp "fun" (Tyapp "real") (Tyapp "bool"))
(Tyapp "bool"))) (Abs "A0" (Tyapp "real") (Comb (Comb (Const "=" (Tyapp "fun"
(Tyapp "real") (Tyapp "fun" (Tyapp "real") (Tyapp "bool")))) (Comb (Const
"real_neg" (Tyapp "fun" (Tyapp "real") (Tyapp "real"))) (Comb (Const
"real_neg" (Tyapp "fun" (Tyapp "real") (Tyapp "real"))) (Var "A0" (Tyapp
"real"))))) (Var "A0" (Tyapp "real")))))
```

- becomes

```
("¨(Type bool)"¨ ! ("¨(Type (fun real bool))"¨ (Abs ("¨(Type real)"¨
(Var A0)) ("¨(Type bool)"¨ ("¨(Type real)"¨ real_neg ("¨(Type real)"¨
real_neg ("¨(Type real)"¨ (Var A0)))) = ("¨(Type real)"¨ (Var A0))))))
```

Example grammars





CYK Learning and Parsing (KUV, ITP 17)

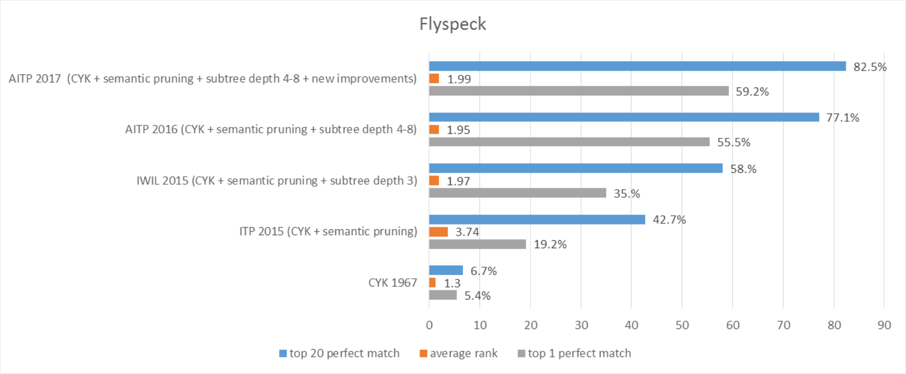
- Induce PCFG (probabilistic context-free grammar) from the trees
  - Grammar rules obtained from the inner nodes of each grammar tree
  - Probabilities are computed from the frequencies
- The PCFG grammar is binarized for efficiency
  - New nonterminals as shortcuts for multiple nonterminals
- CYK: dynamic-programming algorithm for parsing ambiguous sentences
  - input: sentence – a sequence of words and a binarized PCFG
  - output: N most probable parse trees
- Additional semantic pruning
  - Compatible types for free variables in subtrees
- Allow small probability for each symbol to be a variable
- Top parse trees are de-binarized to the original CFG
  - Transformed to HOL parse trees (preterms, Hindley-Milner)
  - typed checked in HOL and then given to an ATP (hammer)

Autoformalization based on PCFG and semantics

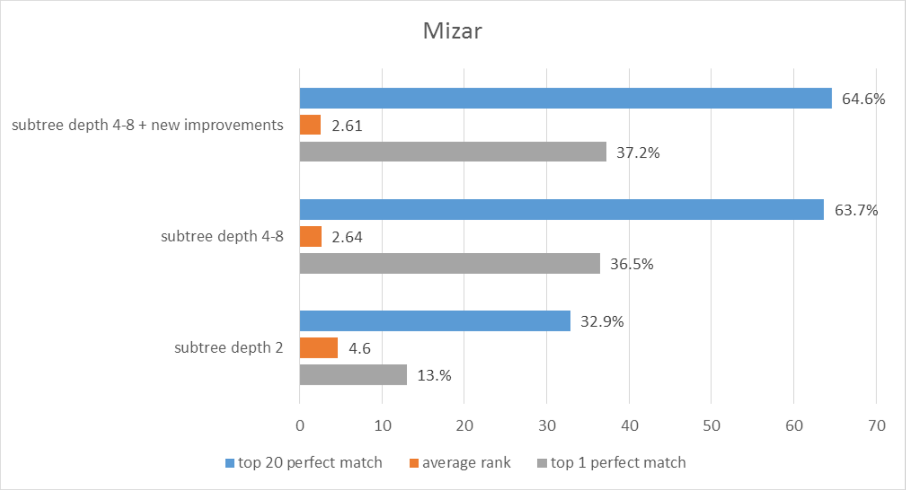
- “`sin ( 0 * x ) = cos pi / 2`”
- produces 16 parses
- of which 11 get type-checked by HOL Light as follows
- with all but three being proved by HOL(y)Hammer
- demo: `http://grid01.ciirc.cvut.cz/~mptp/demo.ogv`

```
sin (&0 * A0) = cos (pi / &2) where A0:real
sin (&0 * A0) = cos pi / &2 where A0:real
sin (&0 * &A0) = cos (pi / &2) where A0:num
sin (&0 * &A0) = cos pi / &2 where A0:num
sin (&(0 * A0)) = cos (pi / &2) where A0:num
sin (&(0 * A0)) = cos pi / &2 where A0:num
csin (Cx (&0 * A0)) = ccos (Cx (pi / &2)) where A0:real
csin (Cx (&0) * A0) = ccos (Cx (pi / &2)) where A0:real^2
Cx (sin (&0 * A0)) = ccos (Cx (pi / &2)) where A0:real
csin (Cx (&0 * A0)) = Cx (cos (pi / &2)) where A0:real
csin (Cx (&0) * A0) = Cx (cos (pi / &2)) where A0:real^2
```

Flyspeck Progress



First Mizar Results (100-fold Cross-validation)



Neural Autoformalization (Wang et al., 2018)

- generate about 1M Latex - Mizar pairs based on Bancerek's work
- 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 (you can help!)
- Recent addition: unsupervised methods (Lample et all 2018) – no need for aligned data!
- Supervised NMT BLEU: best in our paper: 61.6, later: 70.9
- Unsupervised NMT BLEU: fasttext embeddings: 28.56, better embeddings 65.52!

Neural Autoformalization data

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

Neural Autoformalization results

Parameter	Final Test Perplexity	Final Test BLEU	Identical Statements (%)	Identical No-overlap (%)
128 Units	3.06	41.1	40121 (38.12%)	6458 (13.43%)
256 Units	1.59	64.2	63433 (60.27%)	19685 (40.92%)
512 Units	1.6	**67.9**	66361 (63.05%)	21506 (44.71%)
1024 Units	**1.51**	61.6	**69179 (65.73%)**	**22978 (47.77%)**
2048 Units	2.02	60	59637 (56.66%)	16284 (33.85%)

Neural Fun – Performance after Some Training

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

Unsupervised NMT Fun on Short Formulas

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

Acknowledgments

- Prague Automated Reasoning Group `http://arg.ciirc.cvut.cz/`:
  - Jan Jakubuv, 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 References

- ARG ML&R course: `http://arg.ciirc.cvut.cz/teaching/mlr19/index.html`
- C. Kaliszyk: `http://cl-informatik.uibk.ac.at/teaching/ss18/mltp/content.php`
- C. Kaliszyk, J. Urban, H. Michalewski, M. Olsak: Reinforcement Learning of Theorem Proving. CoRR abs/1805.07563 (2018)
- Z. Goertzel, J. Jakubuv, S. Schulz, J. Urban: ProofWatch: Watchlist Guidance for Large Theories in E. CoRR abs/1802.04007 (2018)
- T. Gauthier, C. Kaliszyk, J. Urban, R. Kumar, M. Norrish: Learning to Prove with Tactics. CoRR abs/1804.00596 (2018).
- J. Jakubuv, J. Urban: ENIGMA: Efficient Learning-Based Inference Guiding Machine. CICM 2017: 292-302
- S. M. Loos, G. Irving, C. Szegedy, C. Kaliszyk: Deep Network Guided Proof Search. LPAR 2017: 85-105
- L. Czajka, C. Kaliszyk: Hammer for Coq: Automation for Dependent Type Theory. J. Autom. Reasoning 61(1-4): 423-453 (2018)
- J. C. Blanchette, C. Kaliszyk, L. C. Paulson, J. Urban: Hammering towards QED. J. Formalized Reasoning 9(1): 101-148 (2016)
- 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
- C. Kaliszyk, J. Urban, J. Vyskocil: Automating Formalization by Statistical and Semantic Parsing of Mathematics. ITP 2017: 12-27
- Q. Wang, C. Kaliszyk, J. Urban: First Experiments with Neural Translation of Informal to Formal Mathematics. CoRR abs/1805.06502 (2018)
- J. Urban, J. Vyskocil: Theorem Proving in Large Formal Mathematics as an Emerging AI Field. LNCS 7788, 240-257, 2013.

Thanks and Advertisement

- Thanks for your attention!
- AITP – Artificial Intelligence and Theorem Proving
- March 22–27, 2020, Aussois, France, `aitp-conference.org`
- ATP/ITP/Math vs AI/Machine-Learning people, Computational linguists
- Discussion-oriented and experimental - submit a talk abstract!
- Grown to 80 people in 2019