Machine Learning for Proof Automation and Formalization

Josef Urban

Czech Technical University in Prague

Outline

Machine Learning (a.k.a. Function Approximation)

Learning vs. Reasoning

Learning of Theorem Proving

Autoformalization

Data vs Theory-driven Approach to Problem Solving

- John Shawe-Taylor and Nello Cristianini – Kernel Methods for Pattern Analysis (2004):
- *Many of the most interesting problems in AI and computer science in general are extremely complex often making it difficult or even impossible to specify an explicitly programmed solution.*
- *As an example consider the problem of recognising genes in a DNA sequence. We do not know how to specify a program to pick out the subsequences of, say, human DNA that represent genes.*
- *Similarly we are not able directly to program a computer to recognise a face in a photo.*
- me: or to prove an arbitrary reasonably easy lemma

Data vs Theory-driven Approach to Problem Solving

- *Learning systems offer an alternative methodology for tackling these problems.*
- *By exploiting the knowledge extracted from a sample of data, they are often capable of adapting themselves to infer a solution to such tasks.*
- *We will call this alternative approach to software design the learning methodology.*
- *It is also referred to as the data driven or data based approach, in contrast to the theory driven approach that gives rise to precise specifications of the required algorithms.*

Sample of Learning Approaches

- **neural networks** (statistical ML) – backpropagation, deep learning, convolutional, recurrent, etc.
- **decision trees, random forests** – find good classifying attributes (and/or their values); more explainable
- **support vector machines** – find a good classifying hyperplane, possibly after non-linear transformation of the data (*kernel methods*)
- **k-nearest neighbor** – find the k nearest neighbors to the query, combine their solutions
- **naive Bayes** – compute probabilities of outcomes assuming complete (naive) independence of characterizing features (just multiplying probabilities)
- **inductive logic programming** (symbolic ML) – generate logical explanation (program) from a set of ground clauses by generalization
- **genetic algorithms** – evolve large population by crossover and mutation
- various combinations of statistical and symbolic approaches
- supervised, unsupervised, reinforcement learning (actions, explore/exploit, cumulative reward)

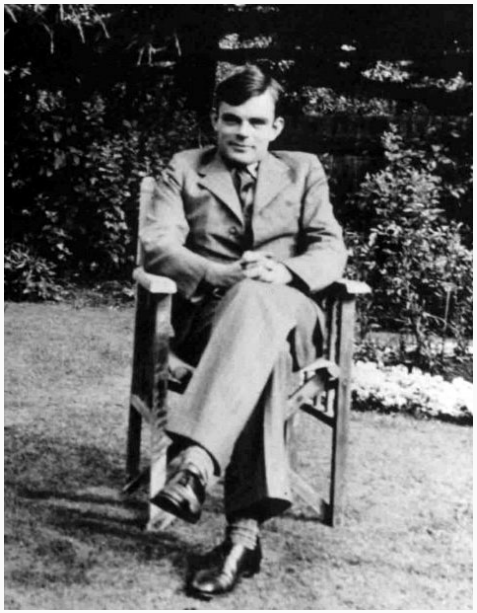
Learning – Features and Data Preprocessing

- Extremely important - if irrelevant, there is no use to learn the function from input to output (“garbage in garbage out”)
- Feature discovery – a big field
- Deep Learning – design neural architectures that automatically find important high-level features for a task
- Latent Semantics, dimensionality reduction: use linear algebra (eigenvector decomposition) to discover the most similar features, make approximate equivalence classes from them
- word2vec and related methods: represent words/sentences by *embeddings* (in a high-dimensional real vector space) learned by predicting the next word on a large corpus like Wikipedia
- math and theorem proving: syntactic/semantic patterns/abstractions
- how do we represent math objects (formulas, proofs, ideas) in our mind?

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 large computer-understandable math corpora?

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, ...
- ...

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
- HOLStep – 2016, kernel inferences
- Coq – since 2013/2016
- HOL4 – since 2014
- ACL2 – 2014?
- Lean? – 2017?

Statistical Guidance of Connection Tableau

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

Clauses:

$c_1 : P(x)$

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

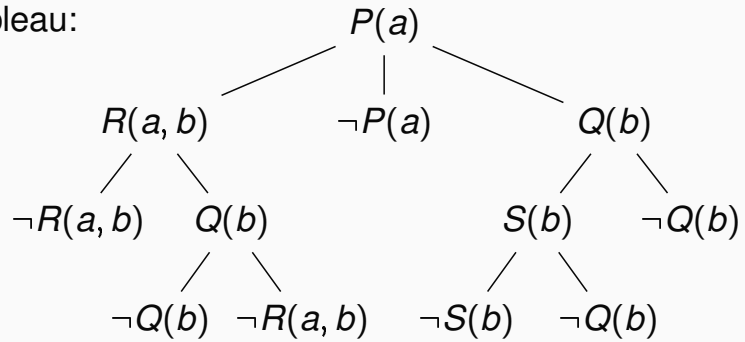
$c_3 : S(x) \vee \neg Q(b)$

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

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

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

Closed Connection Tableau:



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
- added Monte-Carlo Tree Search (MCTS)
- MCTS search nodes are sequences of clause application
- a good heuristic to explore new vs exploit good nodes:

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

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

Statistical Guidance of Connection Tableau – rlCoP

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

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

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

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

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 (features engineering), Deep guidance (neural nets)
- 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 the AIM benchmark
- Deep guidance: convolutional nets - no feature engineering but slow

ProofWatch: Statistical/Semantic Guidance of E

- Bob Veroff's *hints* method used for Prover9/AIM
- solve many easier/related problems
- load their useful lemmas on the *watchlist*
- boost inferences on clauses that subsume a watchlist clause
- watchlist parts are fast thinking, bridged by standard 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

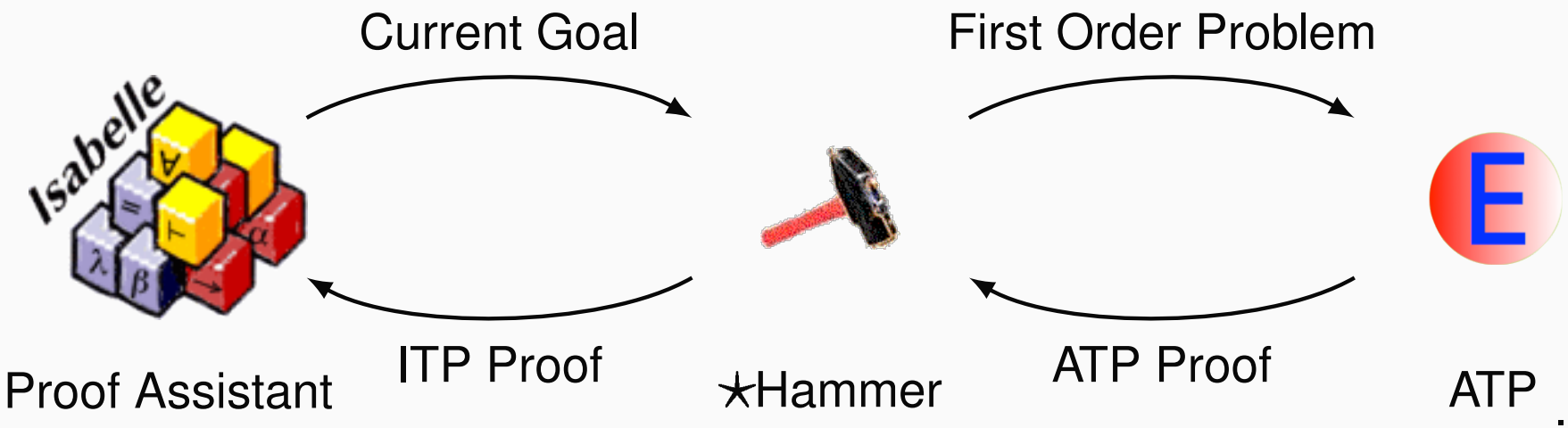
Final state of the proof progress for the 32 proofs guiding `YELLOW_5:36`

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

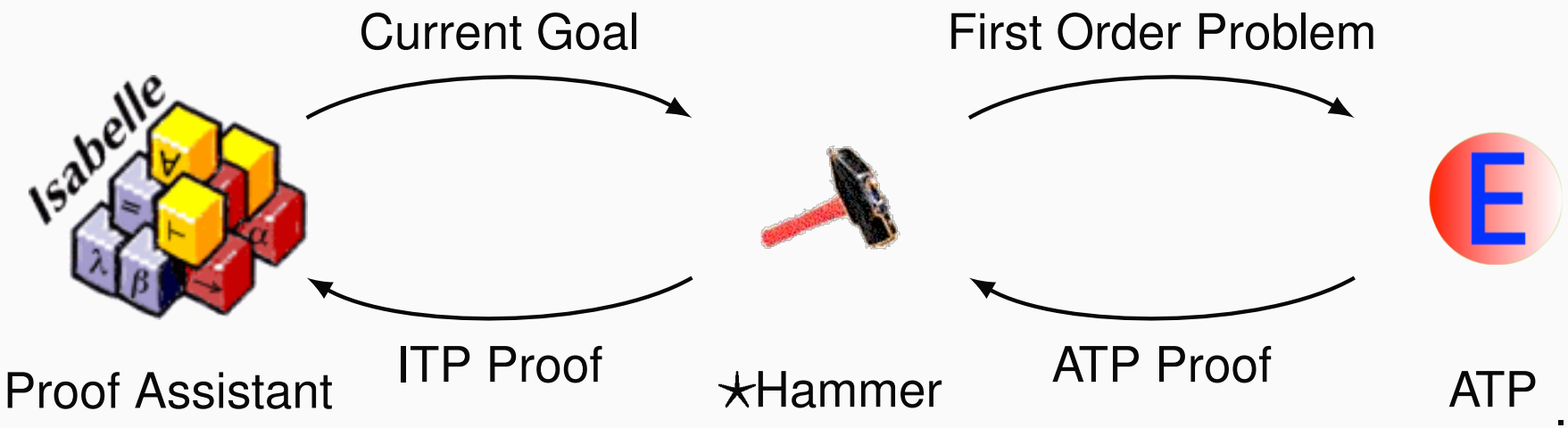
High-level ATP guidance: Premise Selection/Hammers

- 2003: Can existing ATPs be used on the freshly translated Mizar library?
- About 80000 nontrivial math facts at that time – impossible to use them all
- Mizar Proof Advisor (2003):
- train naive-Bayes fact selection on previous Mizar/MML
- recommend relevant premises 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 AI-ATP systems ($\star$-Hammers)

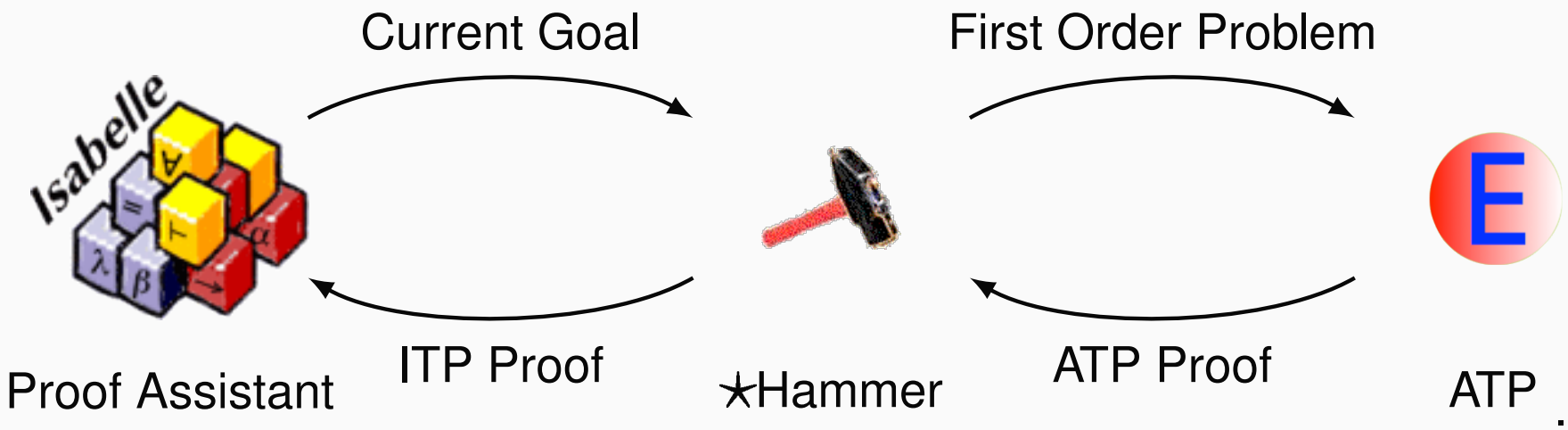


Today's AI-ATP systems ($\star$-Hammers)



How much can it do?

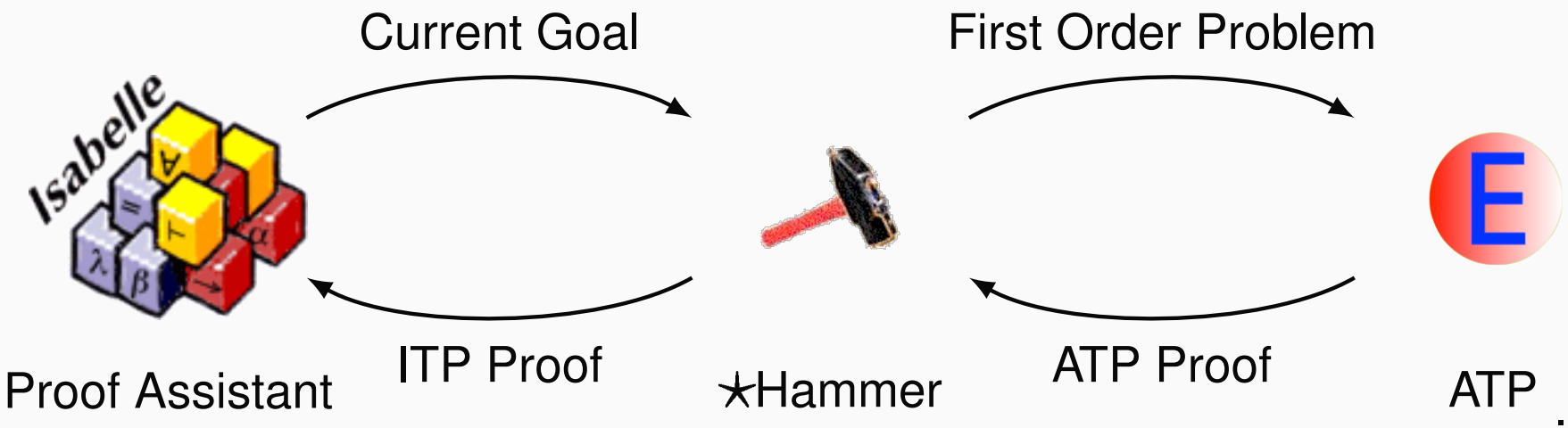
Today's AI-ATP systems ($\star$-Hammers)



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

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



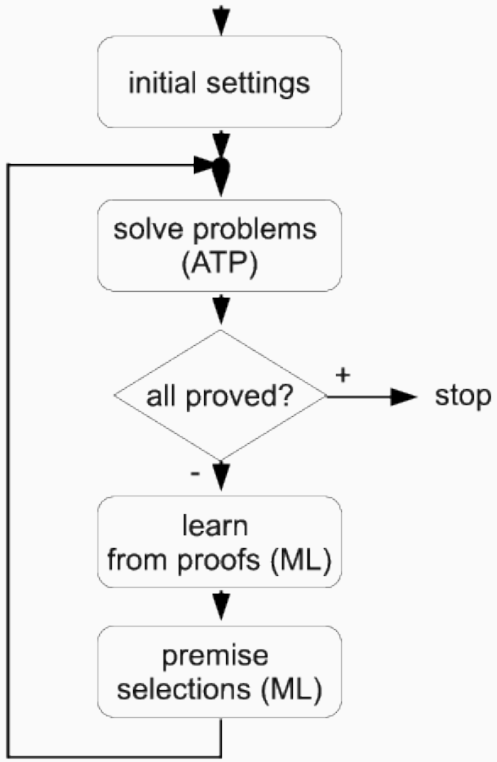
How much can it do?

- 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

$\approx$ 45% success rate

Machine Learner for Automated Reasoning

- MaLARea (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



Recent 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 sequence models, definitional embeddings (Google Research)
- Hammers combined with statistical tactical search: TacticToe (HOL4)
- Learning in binary setting from many alternative proofs
- Negative/positive mining (ATPBoost)

Summary of Features Used

- From syntactic to more semantic:
- Constant and function symbols
- Walks in the term 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
- Evaluation in a large set of (finite) models
- LSI/PCA combinations of above
- Neural embeddings of above

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

- 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!)
- work in progress for Coq
- earlier Coq work: SEPIA (Gransden et al, 2015) - inferred automata

Statistical/Semantic Parsing of Informalized HOL

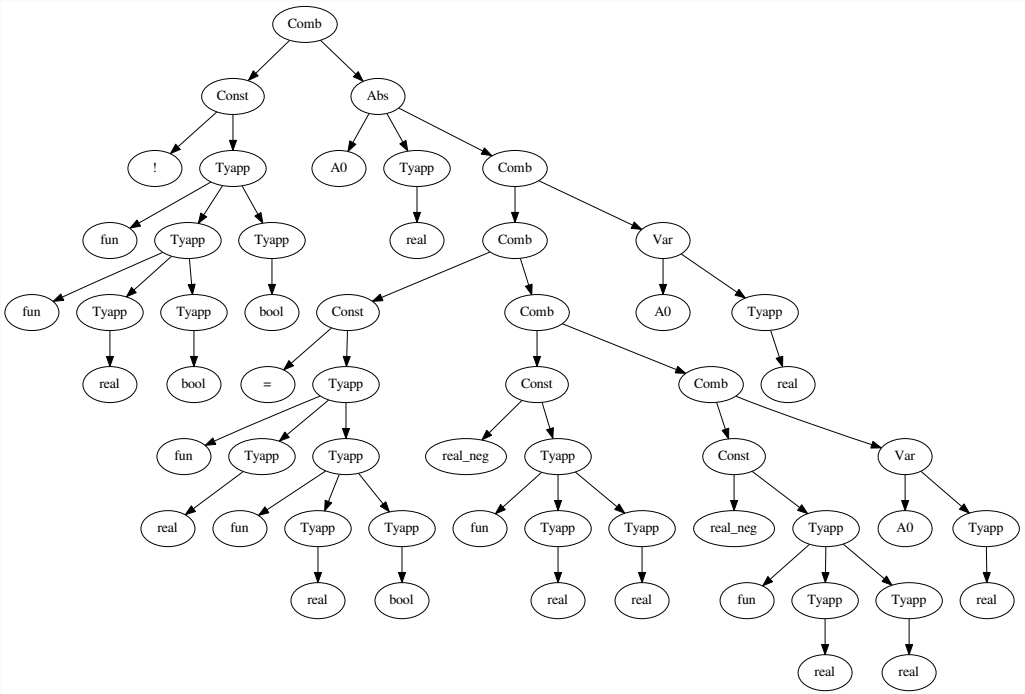
- Goal: Learn understanding of informal math formulas and reasoning
- Experiments with the CYK chart parser linked to semantic methods
- Training and testing examples exported form Flyspeck formulas
  - Along with their informalized versions
- Grammar parse trees
  - Annotate each (nonterminal) symbol 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: $\forall x.\ --x = x$

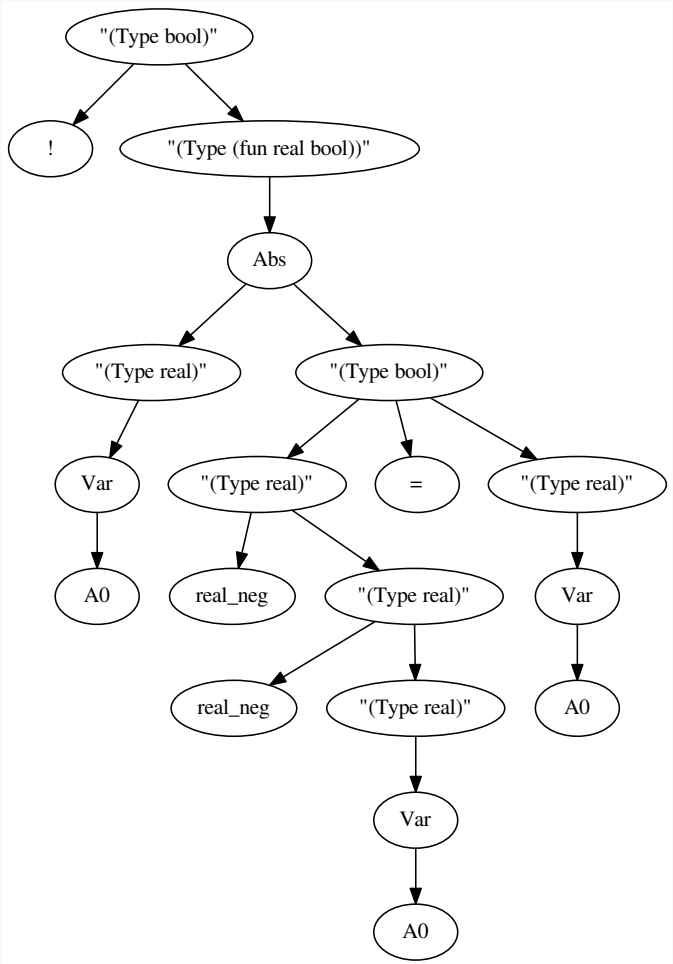
```
(Comb (Const "!" (Tyapp "fun" (Tyapp "fun" (Tyapp "real") (Tyapp "bool"))
(Tyapp "bool"))) (Abs "A0" (Tyapp "real") (Comb (Comb (Const "=" (Tyapp "fun"
(Tyapp "real") (Tyapp "fun" (Tyapp "real") (Tyapp "bool")))) (Comb (Const
"real_neg" (Tyapp "fun" (Tyapp "real") (Tyapp "real"))) (Comb (Const
"real_neg"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

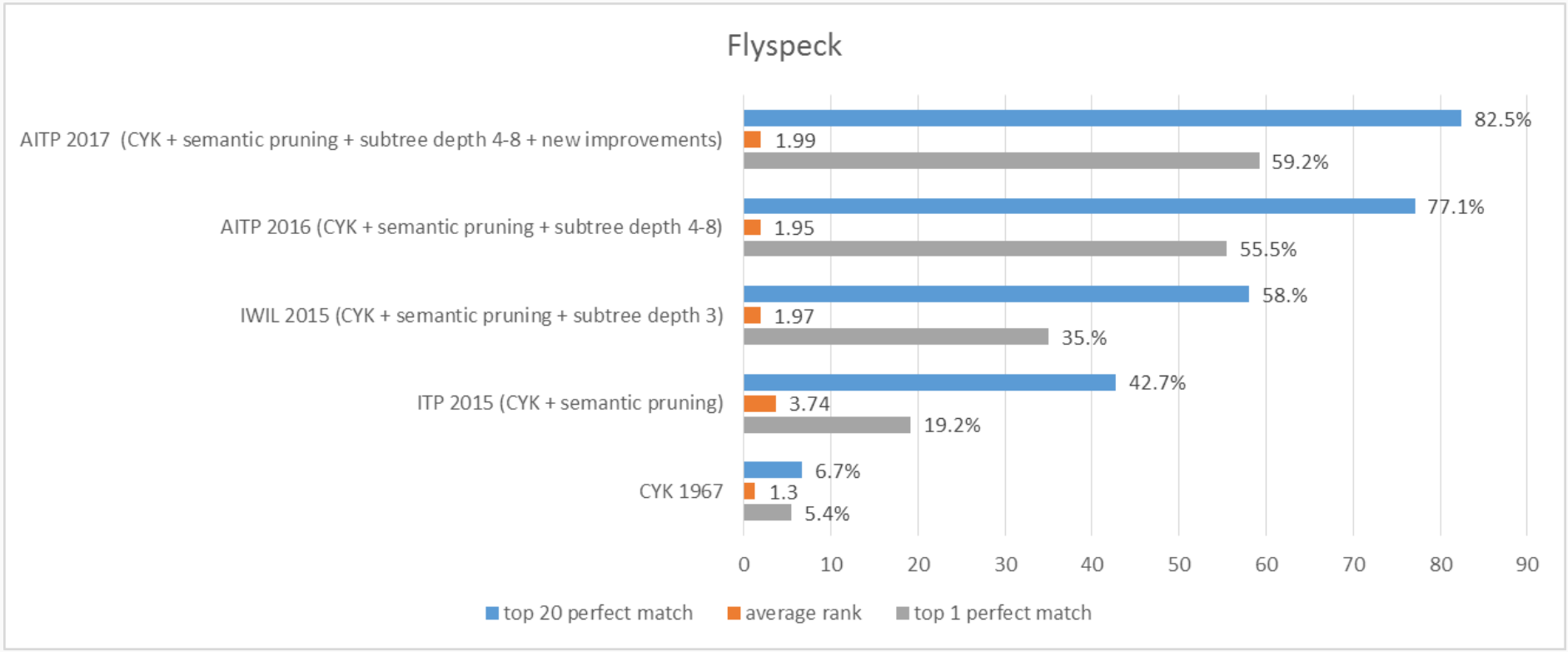
- Induce PCFG (probabilistic context-free grammar) from the trees
  - Grammar rules obtained from the inner nodes of each grammar tree
  - Probabilities are computed from the frequencies
- The PCFG grammar is binarized for efficiency
  - New nonterminals as shortcuts for multiple nonterminals
- CYK: 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)

Online parsing system

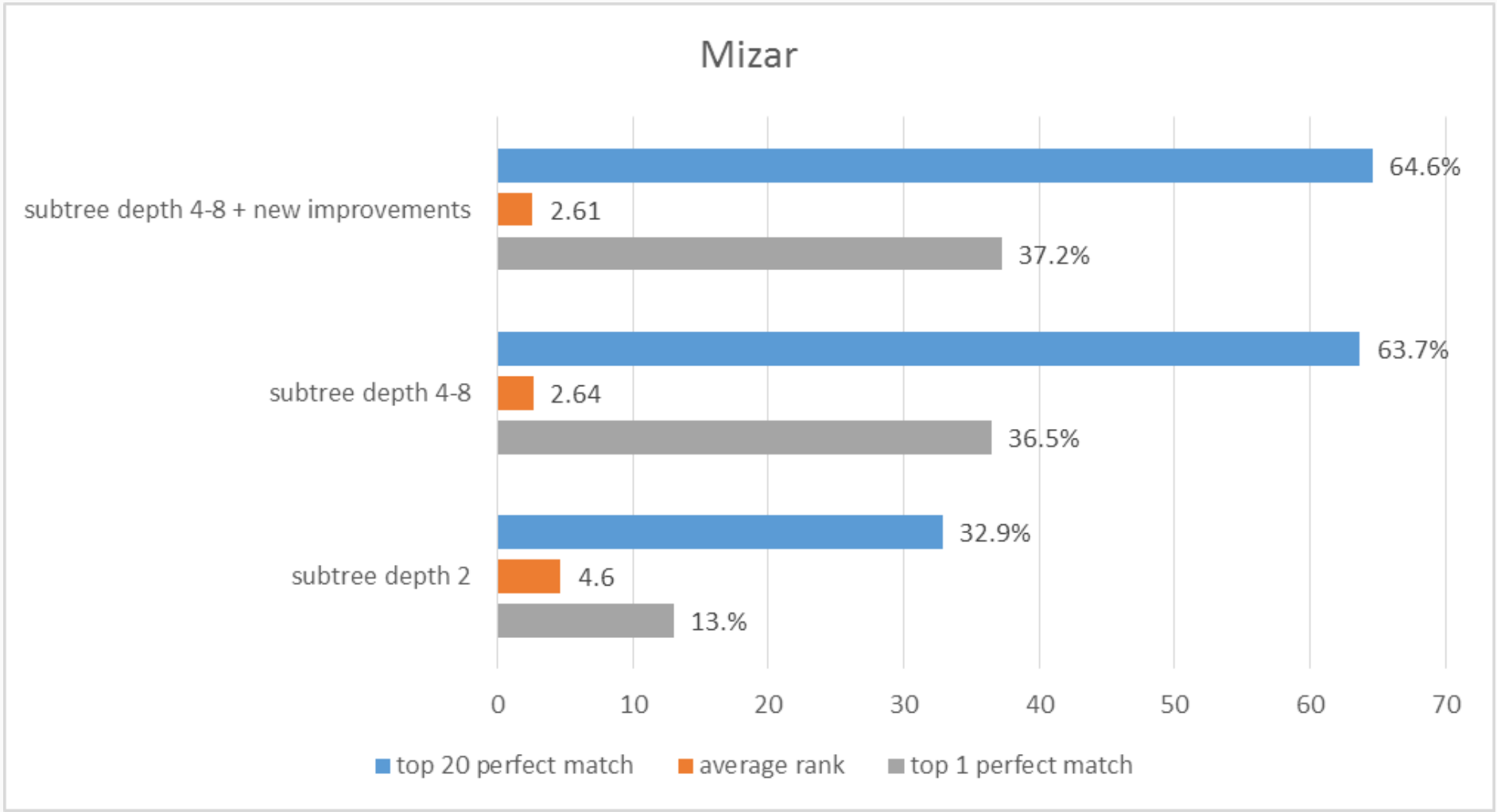
- “`sin ( 0 * x ) = cos pi / 2`”
- produces 16 parses
- of which 11 get type-checked by HOL Light as follows
- with all but three being proved by HOL(y)Hammer

```
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!)

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 ) ;`

Acknowledgments

- Prague Automated Reasoning Group `http://arg.ciirc.cvut.cz/`:
  - Petr Stepanek, Jiri Vyskocil, Petr Pudlak, David Stanovsky, Krystof Hoder, Jan Jakubuv, Ondrej Kuncar, Martin Suda, Zar Goertzel, Bartosz Piotrowski, Lasse Blaauwbroek, ...
- 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

- 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
- April 8–12, 2019, Obergurgl, Austria, `aitp-conference.org`
- ATP/ITP/Math vs AI/Machine-Learning people, Computational linguists
- Discussion-oriented and experimental
- Grown to 60 people in 2018