Artificial Intelligence and Theorem Proving

Josef Urban

Czech Technical University in Prague

European Research Council
Established by the European Commission

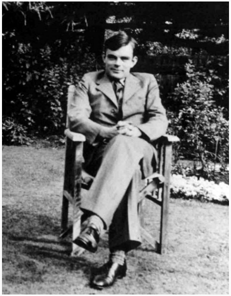
Outline

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!

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 ...

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?

- 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

Irrationality of 2 (informal text)

tiny proof from Hardy & Wright:

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

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

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

Irrationality of 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 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]);;
```

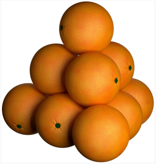
Irrationality of 2 in Isabelle/HOL

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

Big Example: The Flyspeck project

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

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



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

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, ...
- 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 I try to do ...
- ... typically by learning in various ways from the knowledge bases ...

Mizar demo

`http://grid01.ciirc.cvut.cz/~mptp/out4.ogv`

High-level ATP guidance: Premise Selection

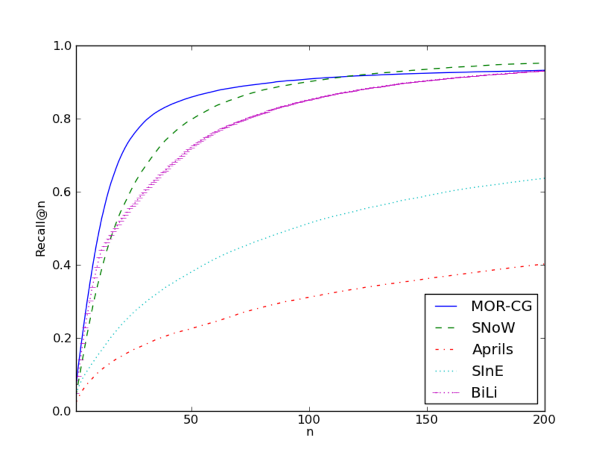
- Early 2003: Can existing ATPs be used over the freshly translated Mizar library?
- About 80000 nontrivial math facts at that time – impossible to use them all
- Is good premise selection for proving a new conjecture possible at all?
- Or is it a mysterious power of mathematicians? (Penrose)
- Today: 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)

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
- 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!)

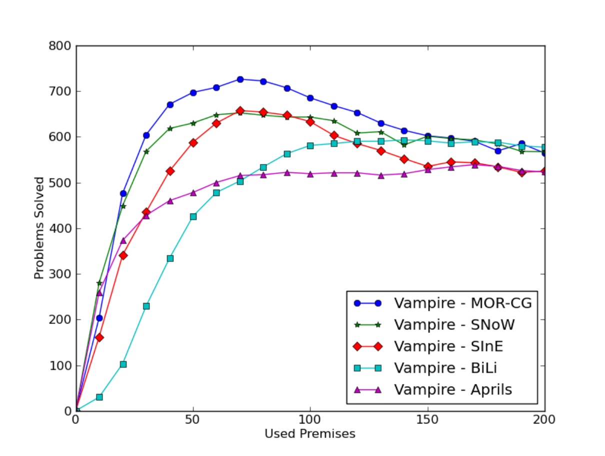
ML Evaluation of methods on MPTP2078 – recall

- 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

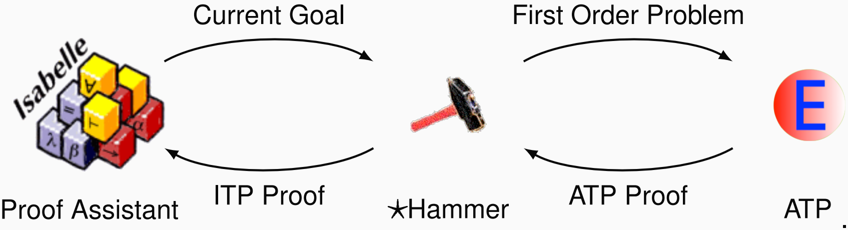


ATP Evaluation of methods on MPTP2078

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



Today's AI-ATP systems (⋆-Hammers)



How much can it do?

- Isabelle (Auth, Jinja) – Sledgehammer
- Flyspeck (+ HOL Light and Multivariate), HOL4 – HOL(y)Hammer
- Mizar / MML – MizAR

$\approx$ 45% success rate

Recent Improvements

- Semantic features encoding term matching
- Distance-weighted k-nearest neighbor, TF-IDF, LSI, better ensembles (MePo)
- Matching and transfering concepts and theorems between libraries (Gauthier & Kaliszyk)
- Lemmatization – extracting and considering millions of low-level lemmas
- Neural sequence models, definitional embeddings (Google Research)

FACE_OF_POLYHEDRON_POLYHEDRON

```
let FACE_OF_POLYHEDRON_POLYHEDRON = prove
 (`!s:real^N->bool c. polyhedron s /\ c face_of s ==> polyhedron c`,
  REPEAT STRIP_TAC THEN FIRST_ASSUM
   (MP_TAC o GEN_REWRITE_RULE I [POLYHEDRON_INTER_AFFINE_MINIMAL]) THEN
  REWRITE_TAC[RIGHT_IMP_EXISTS_THM; SKOLEM_THM] THEN
  SIMP_TAC[LEFT_IMP_EXISTS_THM; RIGHT_AND_EXISTS_THM; LEFT_AND_EXISTS_THM] THEN
  MAP_EVERY X_GEN_TAC
   [`f:(real^N->bool)->bool`; `a:(real^N->bool)->real^N`;
    `b:(real^N->bool)->real`] THEN
  STRIP_TAC THEN
  MP_TAC(ISPECL [`s:real^N->bool`; `f:(real^N->bool)->bool`;
                 `a:(real^N->bool)->real^N`; `b:(real^N->bool)->real`]
         FACE_OF_POLYHEDRON_EXPLICIT) THEN
  ANTS_TAC THENL [ASM_REWRITE_TAC[] THEN ASM_MESON_TAC[]; ALL_TAC] THEN
  DISCH_THEN(MP_TAC o SPEC `c:real^N->bool`) THEN ASM_REWRITE_TAC[] THEN
  ASM_CASES_TAC `c:real^N->bool = {}` THEN
  ASM_REWRITE_TAC[POLYHEDRON_EMPTY] THEN
  ASM_CASES_TAC `c:real^N->bool = s` THEN ASM_REWRITE_TAC[] THEN
  DISCH_THEN SUBST1_TAC THEN MATCH_MP_TAC POLYHEDRON_INTERS THEN
  REWRITE_TAC[FORALL_IN_GSPEC] THEN
  ONCE_REWRITE_TAC[SIMPLE_IMAGE_GEN] THEN
  ASM_SIMP_TAC[FINITE_IMAGE; FINITE_RESTRICT] THEN
  REPEAT STRIP_TAC THEN REWRITE_TAC[IMAGE_ID] THEN
  MATCH_MP_TAC POLYHEDRON_INTER THEN
  ASM_REWRITE_TAC[POLYHEDRON_HYPERPLANE]);;
```

FACE_OF_POLYHEDRON_POLYHEDRON

```
polyhedron s /\ c face_of s ==> polyhedron c
```

HOL Light proof: could not be re-played by ATPs.

Alternative proof found by a hammer based on `FACE_OF_STILLCONVEX`:
Face *t* of a convex set *s* is equal to the intersection of *s* with the affine hull of *t*.

```
FACE_OF_STILLCONVEX:
 !s t:real^N->bool. convex s ==>
 (t face_of s <=>
  t SUBSET s /\ convex(s DIFF t) /\ t = (affine hull t) INTER s)
POLYHEDRON_IMP_CONVEX:
 !s:real^N->bool. polyhedron s ==> convex s
POLYHEDRON_INTER:
 !s t:real^N->bool. polyhedron s /\ polyhedron t
   ==> polyhedron (s INTER t)
POLYHEDRON_AFFINE_HULL:
 !s. polyhedron(affine hull s)
```

Low-level guidance for tableau: Machine Learning Connection Prover (MaLeCoP)

- MaLeCoP: put the AI methods inside a tableau ATP
- the learning/deduction feedback loop runs across problems and inside problems
- The more problems/branches you solve/close, the more solutions you can learn from
- The more solutions you can learn from, the more problems you solve
- first prototype (2011): very slow learning-based advice (1000 times slower than inference steps)
- already about 20-time proof search shortening on MPTP Challenge compared to leanCoP
- second version (2015): Fairly Efficient MaLeCoP (= FEMaLeCoP)
- about 15% improvement over untrained leanCoP on the MPTP problems

Low-level guidance for superposition: ENIGMA

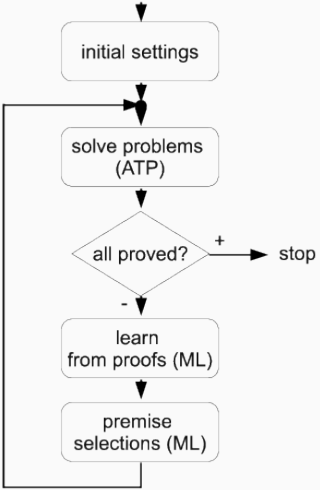
- Train a fast classifier distinguishing good and bad generated clauses
- Plug it into a superposition prover (E prover) as a clause evaluation heuristic
- Combine it with various ways with more standard (common-sense) guiding methods
- Very recent work, 86% improvement of the best tactic

Examples of self-evolving metasystems

- Various positive feedback loops
- Machine Learner for Automated Reasoning (MaLARea)
- Blind Strategymaker (BliStr)

Machine Learner for Automated Reasoning

Feedback loop interleaving ATP with learning premise selection



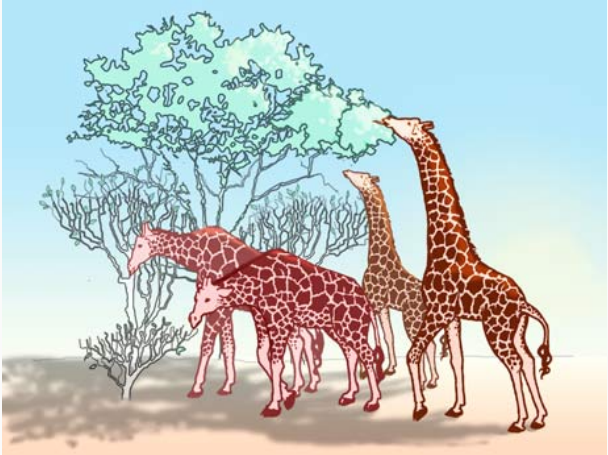
MaLARea

- MaLARea 0.4 (CASC@Turing) - unordered mode, explore & exploit, etc.
- The more problems you solve (and fail to solve), the more solutions (and failures) you can learn from
- The more you can learn from, the more you solve
- In some sense also conjecturing (omiting definitions)
- The CASC@Turing performance curve flat for quite a while:
- `http://www.cs.miami.edu/~tptp/CASC/J6/TuringWWWFiles/ResultsPlots.html#MRTProblems`
- CASC 2013, MaLARea 0.5 (ordered mode, many changes): solved 77% more problems than the second system
- `http://www.cs.miami.edu/~tptp/CASC/24/WWWFiles/DivisionSummary1.html`

BliStr: Blind Strategymaker

- Problem: how do we put all the sophisticated ATP techniques together?
- E.g., Is conjecture-based guidance better than proof-trace guidance?
- Grow a population of diverse strategies by iterative local search and evolution!
- Dawkins: The Blind Watchmaker

BliStr: Blind Strategymaker



- The 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

BliStr: Blind Strategymaker

- Use clusters of similar solvable problems to train for unsolved problems
- Interleave low-time training with high-time evaluation
- Thus co-evolve the strategies and their training problems
- 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"*

Learning Informal to Formal Translation

- Dense Sphere Packings: A Blueprint for Formal Proofs
  - 400 theorems and 200 concepts mapped [*Hales13*]
  - simple wiki
- Feit-Thompson theorem by Gonthier [*Gonthier13*]
  - Two graduate books
- Compendium of Continuous Lattices (CCL)
  - 60% formalized in Mizar [*BancerekRudnicki02*]
  - high-level concepts and theorems aligned
- ProofWiki with detailed proofs and symbol linking
  - General topology corresponence with Mizar
  - Similar projects (PlanetMath, ...)

Aligned Formal and Informal Math - Flyspeck *[CICM13, ITP'13]*

Informal Formal

Definition of [fan, blade] DSKAGVP (fan) [fan $\leftrightarrow$ FAN]

Let (V, E) be a pair consisting of a set $V \subset \mathbb{R}^3$ and a set E of unordered pairs of distinct elements of V. The pair is said to be a *fan* if the following properties hold.

1. (CARDINALITY) V is finite and nonempty. [cardinality $\leftrightarrow$ fan1]
2. (ORIGIN) $\mathbf{0} \notin V$. [origin $\leftrightarrow$ fan2]
3. (NONPARALLEL) If $\{\mathbf{v}, \mathbf{w}\} \in E$, then $\mathbf{v}$ and $\mathbf{w}$ are not parallel. [nonparallel $\leftrightarrow$ fan6]
4. (INTERSECTION) For all $\varepsilon, \varepsilon' \in E \cup \{\{\mathbf{v}\} \ : \ \mathbf{v} \in V\}$, [intersection $\leftrightarrow$ fan7]

$$C(\varepsilon) \cap C(\varepsilon') = C(\varepsilon \cap \varepsilon').$$

When $\varepsilon \in E$, call $C^0(\varepsilon)$ or $C(\varepsilon)$ a *blade* of the fan.

basic properties

The rest of the chapter develops the properties of fans. We begin with a completely trivial consequence of the definition.

Informal Formal

Lemma [] CTVTAQA (subset-fan)

If (V, E) is a fan, then for every $E' \subset E$, (V, E') is also a fan.

Proof

This proof is elementary.

Informal Formal

Lemma [fan cyclic] XOHLED

[$E(v) \leftrightarrow$ set_of_edge] Let (V, E) be a fan. For each $\mathbf{v} \in V$, the set

$$E(\mathbf{v}) = \{\mathbf{w} \in V \ : \ \{\mathbf{v}, \mathbf{w}\} \in E\}$$

is cyclic with respect to $(\mathbf{0}, \mathbf{v})$.

Proof

If $\mathbf{w} \in E(\mathbf{v})$, then $\mathbf{v}$ and $\mathbf{w}$ are not parallel. Also, if $\mathbf{w} \neq \mathbf{w}' \in E(\mathbf{v})$, then

Informal Formal

```
#DSKAGVP?

let FAN=new_definition`FAN(x,V,E) <=> ((UNIONS E) SUBSET V) /\ graph(E) /\ fan1(x,V,E) /\ fan2(x,V,
fan6(x,V,E)/\ fan7(x,V,E)`;;
```

basic properties

The rest of the chapter develops the properties of fans. We begin with a completely trivial consequence of the definition.

Informal Formal

```
let CTVTAQA=prove(`!(x:real^3) (V:real^3->bool) (E:(real^3->bool)->bool) (E1:(real^3->bool)->bool)
FAN(x,V,E) /\ E1 SUBSET E
==>
FAN(x,V,E1)`,

REPEAT GEN_TAC
THEN REWRITE_TAC[FAN;fan1;fan2;fan6;fan7;graph]
THEN ASM_SET_TAC[]);;
```

Informal Formal

```
let XOHLED=prove(`!(x:real^3) (V:real^3->bool) (E:(real^3->bool)->bool) (v:real^3).
FAN(x,V,E) /\ v IN V
==> cyclic_set (set_of_edge v V E) x v`,

MESON_TAC[CYCLIC_SET_EDGE_FAN]);;
```

Statistical Parsing of Informalized HOL

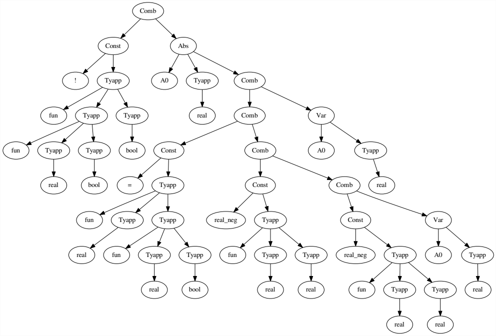
- Experiments with the CYK chart parser linked to semantic methods
- Training and testing examples exported form Flyspeck formulas
  - Along with their informalized versions
- Grammar parse trees
  - Annotate each (nonterminal) symbol with its HOL type
  - Also “semantic (formal)” nonterminals annotate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

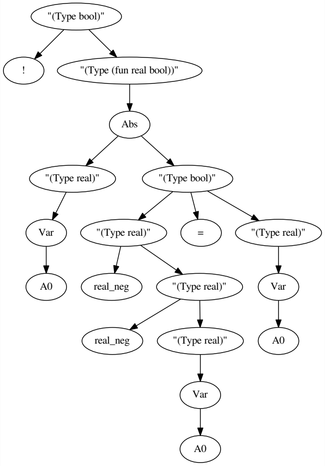
```
(Comb (Const "!" (Tyapp "fun" (Tyapp "fun" (Tyapp "real") (Tyapp "bool"))
(Tyapp "bool"))) (Abs "A0" (Tyapp "real") (Comb (Comb (Const "=" (Tyapp "fun"
(Tyapp "real") (Tyapp "fun" (Tyapp "real") (Tyapp "bool")))) (Comb (Const
"real_neg" (Tyapp "fun" (Tyapp "real") (Tyapp "real"))) (Comb (Const
"real_neg"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

- 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)

Experiments with Informalized Flyspeck

- 22000 Flyspeck theorem statements informalized
  - 72 overloaded instances like "+" for `vector_add`
  - 108 infix operators
  - forget all "prefixes"
    - `real_, int_, vector_, nadd_, hreal_, matrix_, complex_`
    - `ccos, cexp, clog, csin, ...`
    - `vsum, rpow, nsum, list_sum, ...`
  - Deleting all brackets, type annotations, and casting functors
    - `Cx` and `real_of_num` (which alone is used 17152 times).
- online parsing/proving demo system
- 100-fold cross-validation

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
```

Results over Flyspeck

- First version (2015): In 39.4% of the 22,000 Flyspeck sentences the correct (training) HOL parse tree is among the best 20 parses
- its average rank: 9.34
- Second version (2016): 67.7% success in top 20 and average rank 3.35
- 24% of them AITP provable

Pointers to Formal Parsing

- Demo of the probabilistic/semantic parser trained on informal/formal Flyspeck pairs:
- `http://colo12-c703.uibk.ac.at/hh/parse.html`
- The linguistic/semantic methods explained in `http://dx.doi.org/10.1007/978-3-319-22102-1_15`
- Compare with Wolfram Alpha:
- `https://www.wolframalpha.com/input/?i=sin+0+*+x+%3D+cos+pi+%2F+2`

Further Challenges in AI over Large Formal KBs

- Refactoring of long ATP proofs for human consumption – 70k-long proof by Bob Veroff & Prover9, 20k by David Stanovsky & Waldmeister, etc.
- Using strong AI/ATP to help automated disambiguation/understanding of arXiv, Stacks, everything?
- Emulating the layer on which mathematicians think – learning from natural language proofs and theories, concept and theory invention
- Conjecturing in large theories – several methods possible (recently tried concept/theory matching)
- What will it take to prove Brouwer or Jordan fully automatically?

Acknowledgments

- Prague Automated Reasoning Group `http://arg.ciirc.cvut.cz/`:
  - Petr Stepanek, Jiri Vyskocil, Jan Jakubuv, Chad Brown, Martin Suda, Ondrej Kuncar, David Stanovsky, Krystof Hoder, Petr Pudlak, ...
- HOL(y)Hammer group in Innsbruck:
  - Cezary Kaliszyk, Thibault Gauthier, Michael Faerber
- ATP and ITP people:
  - Stephan Schulz, Geoff Sutcliffe, Andrej Voronkov, Jens Otten, Larry Paulson, Jasmin Blanchette, John Harrison, Tom Hales, Tobias Nipkow, Mark Adams, Ramana Kumar, Andrzej Trybulec, Piotr Rudnicki, Adam Pease, ...
- Learning2Reason people at Radboud University Nijmegen:
  - Tom Heskes, Daniel Kuehlwein, Evgeni Tsivtsivadze, Herman Geuvers ....
- Google Research: Christian Szegedy, Geoffrey Irving, Alex Alemi, Francois Chollet
- ... and many more ...
- Funding: Marie-Curie, NWO, ERC

Thanks and Advertisement

- Thanks for your attention!
- AITP: `http://aitp-conference.org`
- ATP/ITP/Math vs AI/Machine-Learning people, Computational linguists
- Two EU-funded PhD positions on the AI4REASON project
- `http://ai4reason.org/ai4reasonphd.txt`
- Good background in logic and programming
- Interest in AI, Automated/Formal Reasoning, Machine Learning or Computational Linguistics
- Email to `Josef.Urban@gmail.com`