Learning Reasoning and Understanding in Mathematics

Josef Urban

Czech Technical University in Prague

European Research Council
Established by the European Commission

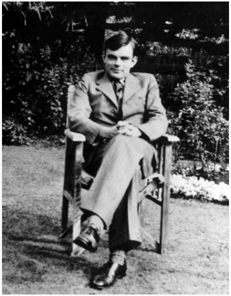
Outline

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:
     - Large-theory Automated Reasoning over Mizar (2003), Isabelle (2005), HOLs (2012,2014), Coq (2016?)
     - AI/ATP/ITP (AITP) systems like MaLARea, Sledgehammer, MizAR, HOL(y)Hammer,
   - 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?
     - What is the disambiguation, conceptualization, conjecturing and knowledge-organization process?
     - “Computing” is just a particular form of “reasoning” (cf. Prolog) - learn programming?

The Plan

1. Make large "formal thought" (Mizar/MML, HOL/Flyspeck ...) accessible to strong reasoning and learning AI tools: DONE (or well under way)
2. Test/Use/Evolve existing AI tools on such large corpora:
   - deductive AI: first-order/higher-order/inductive ATPs, SMTs, decision procs.
   - inductive AI: statistical learning tools (Bayesian, kernels, neural,...),
   - inductive AI: semantic learning tools (ILP - Progol; latent semantics - PCA; probabilistic grammars, ...),
3. Build custom/combined inductive/deductive tools/metasystems:
   - usually combining ATP techniques with ML ideas
   - axiom/clause selection, concept/lemma creation and analogy, strategy generation, etc.
   - high- and low-level feedback loops between reasoning and learning:
   - successful reasoning (a proof) $\rightarrow$ informs learning $\rightarrow$ allows better reasoning $\rightarrow$ and so on ad infinitum ...
4. Continuously test performance, define harder AI tasks as the performance grows

Most of (Math|Mizar) Matches system (MoMM, 2002)

- Load all proof knowledge into an advising system
- some desired properties:
  - when a new conjecture is "efficiently implied" by previous solution, tell us
  - "efficiently implied": generalization with respect to the rich Mizar type system
  - could be extended in various ways towards full theorem proving
- 1M (generalized) proof situations, interreduced in 40 minutes
- hacking of ATP (E prover) indexing datastructures for very large theories
- Most of Mizar Matches!
- More than half proof situations subsumed by a previous one
- Export has to be done correctly, otherwise contradiction subsumes all
- You do not have to do expensive proof search, quantity helps
- Let us do this for all of math!
- Mine Arxiv, Easychair, Planetmath, Wikipedia, textbooks

High-level ATP guidance: Premise Selection

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!)

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!

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).

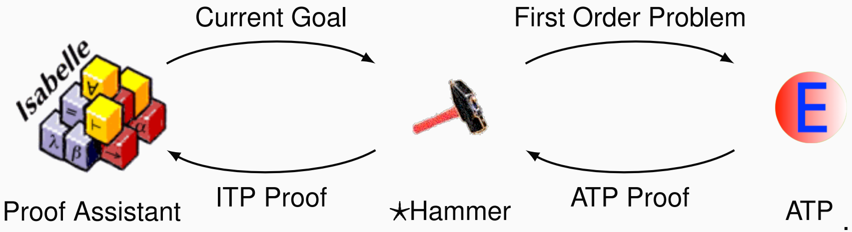
High-level ATP guidance: Premise Selection

- 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)

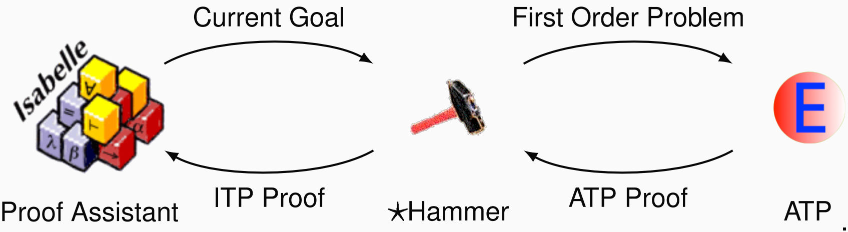
Example system: Mizar Proof Advisor (started 2003)

- train naive-Bayse 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 AI-ATP systems (⋆-Hammers)

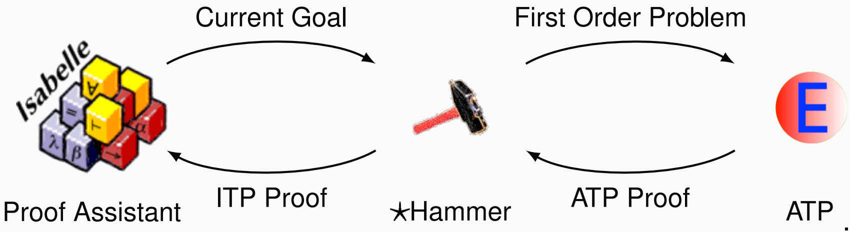


Today’s AI-ATP systems (⋆-Hammers)



How much can it do?

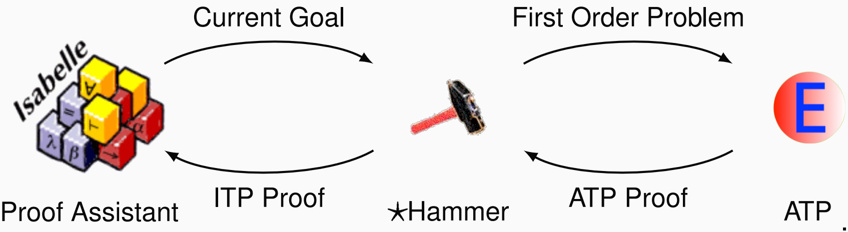
Today's AI-ATP systems (⋆-Hammers)



How much can it do?

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

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



How much can it do?

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

$\approx$ 45% success rate

Low-level Reasoning Guidance

Several systems/methods, I will only mention one

Low-level guidance: 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)
- 2016: Learning guidance now also in Satallax, soon E prover, ITPs, ...

Large-theory Lemmatization and Conjecturing

- Over 1B low-level lemmas in Flyspeck
- 1.5M-7M higher-level lemmas in MML and Flyspeck
- Define fast preprocessing methods to extract the most important ones:
- PageRank, recursive dependency count, recursive use count, etc.
- Use the most important lemmas together with the toplevel theorems - helps by 5-20% (needs more evaluations)
- Conjecturing: guessing the intermediate lemmas in longer proofs (we do not have the methods yet)

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 0.4 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
- MaLARea 0.5 (ordered mode, many changes): solved 77% more problems than the second system

Learning Informal to Formal Translation

- Dense Sphere Packings: A Blueprint for Formal Proofs
  - 400 theorems and 200 concepts mapped [*Hales13*]
  - simple wiki
- Compendium of Continuous Lattices (CCL)
  - 60% formalized in Mizar [*BancerekRudnicki02*]
  - high-level concepts and theorems aligned
- Feit-Thompson theorem by Gonthier [*Gonthier13*]
  - Two graduate books
- 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.

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.

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

```
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

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

```
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

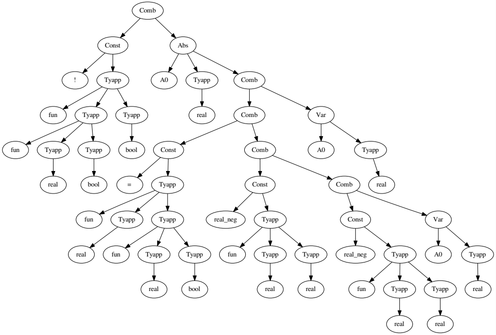
- Experiments with the CYK chart parser linked to semantic methods
- Training and testing examples exported form Flyspeck formulas
  - Along with their informalized versions
- Grammar parse trees
  - Annotate each (nonterminal) symbol with its HOL type
  - Also “semantic (formal)” nonterminals annotate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

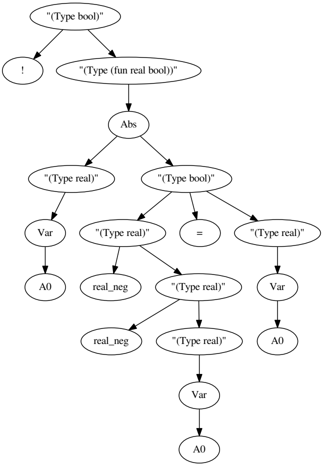
```
(Comb (Const "!" (Tyapp "fun" (Tyapp "fun" (Tyapp "real") (Tyapp "bool"))
(Tyapp "bool"))) (Abs "A0" (Tyapp "real") (Comb (Comb (Const "=" (Tyapp "fun"
(Tyapp "real") (Tyapp "fun" (Tyapp "real") (Tyapp "bool")))) (Comb (Const
"real_neg" (Tyapp "fun" (Tyapp "real") (Tyapp "real"))) (Comb (Const
"real_neg"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

Betting Slide from a talk in Paris in 2014)

- In 20 years, 80% of Flyspeck and MML toplevel theorems will be provable automatically
- (same hardware, same library versions as in2014 - about 40%)
- The same in 30 years - I'll give you 2:1, In 10 years: 60%
- In 25 years, 50% of the toplevel statements in LaTeX-written Msc-level math curriculum textbooks will be parsed automatically and with correct formal semantics
- Hurry up: I will only accept bets up to 10k EUR total (negotiable)