

BEYOND DEDUCTION

Josef Urban

Czech Technical University in Prague



Stephan at AITP'16

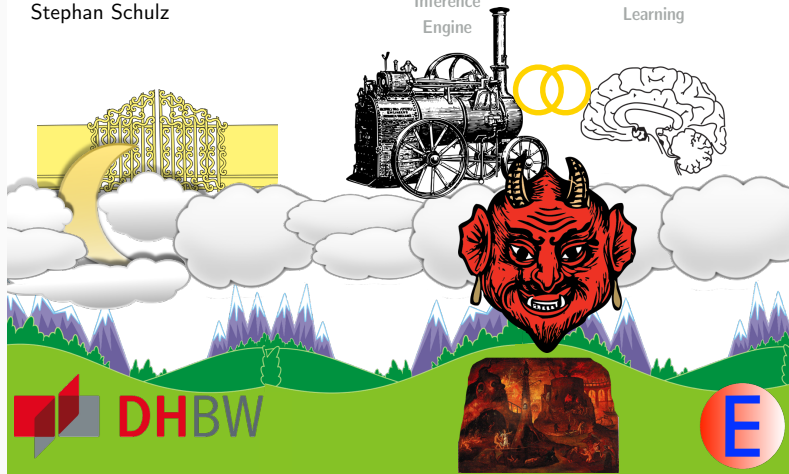
Deduction and Induction

A Match Made in Heaven or a Deal with the Devil?

Stephan Schulz

The
Inference
Engine

Machine
Learning



Questions

- Where/how is machine learning useful in reasoning?
- How do we make it more useful and expand to more tasks?

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.)

Alan Turing 1950

“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.”

Computing machinery and intelligence

AI vs Early Logic-based ATP (Stephanie Dick)

- AI practitioners did not recognize Resolution as a step towards their goals. Reflecting on the history of their field, Argonne practitioners wrote:
- *“When [our colleague] submitted a lovely paper on qualified hyperresolution to one of the main AI journals, a senior editor did not even send it out for refereeing; he just returned a short note stating, “the JACM [Journal of the Association of Computing Machinery] is still publishing such papers, although I don’t know why.” This last phrase symbolized the broader AI community in our eyes. Like perceptrons, formal logic had [. . .] been evaluated and found lacking. Artificial Intelligence sought to make computers like people. Automated Reasoning sought to find problem-solving paths that would be inaccessible to humans, to open up reasoning and logic themselves, to untether them from the laws of human thought.”*
- Part of the criticism from AI, and others, concerned the fact that Resolution and tools like it, took mathematical proof further and further from human view, displacing the indubitable clarity and understanding that earlier logical systems had tried to provide.

Applied Mathematics Division



“Proving theorems in mathematics and logic is too complex a task for total automation, for it requires insight, deep thought, and much knowledge and experience.”



Larry Wos






Press **Esc** to exit full screen

Weighting Mechanism

Ross Overbeek, 1971

Weighting is the process that assigns measures of complexity to clauses in the clause space. The definition(s) of complexity can be chosen by the user to reflect some predisposition that may be, for example, based on an intuitive notion of how to direct the proof search.

- Aura Reference Manual



Low-level ATP guidance: Prover9 hints

- The Prover9 community (ADAM workshop): non-associative algebra, 20-50k long proofs by Prover9 and Waldmeister
- Prover9 hints strategy (Bob Veroff): extract hints from easier proofs to guide more difficult proofs
- To get good hints Bob wants as little conjecture-based inferences as possible:
- Get an “essentially forward proof” by various Prover9 setting
- Exploration to get good hints (not really automated yet)

P9 Example

```
list(given_selection).  
  
% high  
  
part(Hha, high, hint_age, hint & weight < 500 & hint_age < 200000)  
    = 500.  
  
part(Hw, high, weight, hint & weight < 500) = 25.  
part(Ha, high, age, hint & weight < 500) = 5.  
part(Hr, high, random, hint & weight < 500) = 5.  
  
% -false instead of true in case no truth value  
part(Wf, low, weight, false) = 1.  
part(Wnf, low, weight, -false) = 100.  
  
% just in case something isn't covered  
part(TheRest, low, weight, all) = 1.  
  
end_of_list.
```

High-level ATP guidance: Premise Selection

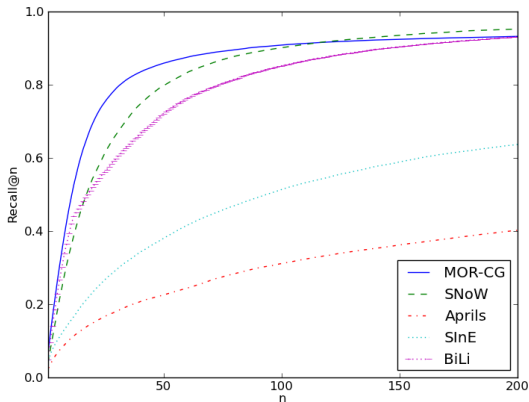
- Can existing ATPs be used over large math libraries?
- Is good premise selection for proving a new conjecture possible at all?
- Or is it a **mysterious power of mathematicians**? (Penrose, intuition?)
- Or should we use some complete exhaustive human-designed algorithms?
- Today: Premise selection is **not a mysterious property of mathematicians!**
- Complete human-engineering is inferior to learning from a large corpus of proofs

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%
- My bet: at least 80% in 20 years
- <http://ai4reason.org/aichallenges.html>

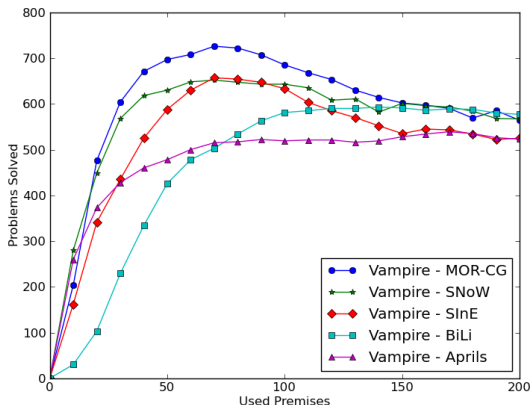
ML Evaluation of methods on MPTP2078 – recall

- 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!



Recent Improvements and Additions

- Semantic features encoding term matching/unification [IJCAI'15]
- Distance-weighted k-nearest neighbor, TF-IDF, LSI, better ensembles (MePo)
- 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)

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

Feature Statistics

- MPTP2078 and MML1147 – 4.5k and 150k formulas

Method	Speed (sec)		Number of features		Learning and prediction (sec)	
	MPTP2078	MML1147	total	unique	knn	naive Bayes
SYM	0.25	10.52	30996	2603	0.96	11.80
TRM _α	0.11	12.04	42685	10633	0.96	24.55
TRM ₀	0.13	13.31	35446	6621	1.01	16.70
MAT _∅	0.71	38.45	57565	7334	1.49	24.06
MAT _r	1.09	71.21	78594	20455	1.51	39.01
MAT _l	1.22	113.19	75868	17592	1.50	37.47
MAT ₁	1.16	98.32	82052	23635	1.55	41.13
MAT ₂	5.32	4035.34	158936	80053	1.65	96.41
MAT _U	6.31	4062.83	180825	95178	1.71	112.66
PAT	0.34	64.65	118838	16226	2.19	52.56
ABS	11	10800	56691	6360	1.67	23.40
UNI	25	N/A	1543161	6462	21.33	516.24

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

- MaLeCoP: put the AI methods inside a tableau ATP (J. Otten - leanCoP)
- 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
- Recently Monte Carlo search (M. Faerber: MonteCop)
- Reinforcement learning (in progress)

Low-level guidance for superposition: ENIGMA

- Train a fast classifier (LIBLINEAR) distinguishing good and bad generated clauses
- Plug it into a superposition prover (E prover) as a clause evaluation heuristic
- ENIGMA: Efficient learNing-based Inference Guiding MACHine
- input: positive and negative examples (good/bad clauses as feature vectors)
- output: model (a vector of feature weights)
- evaluation of a clause feature vector: dot product with the model
- Combine it with various ways with more standard (common-sense) guiding methods
- Very recent work, 86% improvement of the best E tactic on the AIM 2016 CASC benchmark
- About 90% precision in predicting good/bad clauses
- Similar work using (much slower) neural guidance by Google (70-80% precision)

Other guidance for ATPs

- Knowledge base of abstracted lemmas from previous proofs in E (drawing analogies between different theories)
- nearest-neighbor guidance: ConjectureRelativeSymbolWeight in E
- further symbol weighting based on axiom relevance in E
- semantic (model-based) guidance: Prover9
- Waldmeister: theory recognition, optimization of term orderings, etc.
- Our recent work: search for good term orderings in Vampire
- Ongoing work for iProver, SMTs: do not enumerate instances but try the most probable ones

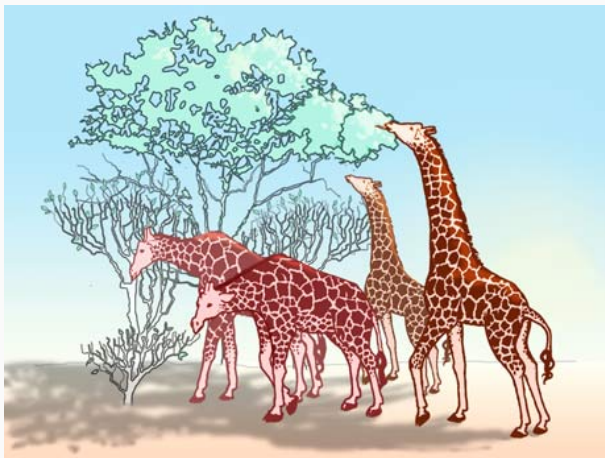
Large-theory Lemmatization and Conjecturing

- 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
- Currently by learning statistical theory analogies and using probabilistic grammars

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
- Recent improvements: BliStrTune – hierarchical approach
- Combine search for low-level and high-level parameters in a loop
- Include multiple ENIGMA models

The E strategy with longest specification in Jan 2012

```
G-E--_029_K18_F1_PI_AE_SU_R4_CS_SP_S0Y:
```

```
--definitional-cnf=24 --simplify-with-unprocessed-units --tstp-in
--split-aggressive --split-clauses=4 --split-reuse-defs
--simul-paramod --forward-context-sr --destructive-er-aggressive
--destructive-er --prefer-initial-clauses -winvfrequank -c1 -Ginvfreq
-F1 --delete-bad-limit=150000000 -WSelectMaxLComplexAvoidPosPred
-H' (
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) )'
-s --print-statistics --print-pid --resources-info --memory-limit=192
```

Its clause evaluation heuristic

G-E--_029_K18_F1_PI_AE_SU_R4_CS_SP_S0Y:

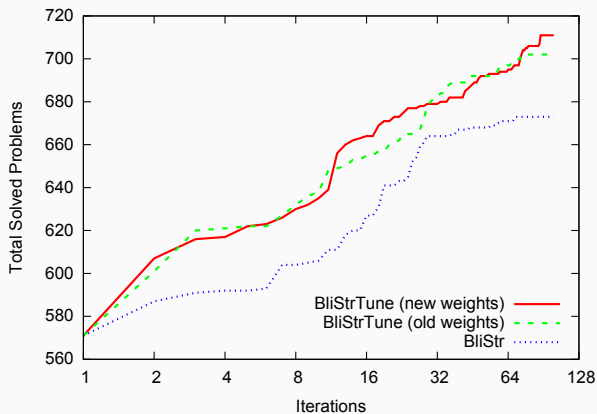
```
4 * ConjectureGeneralSymbolWeight (
    SimulateSOS,100,100,100,50,50,10,50,1.5,1.5,1),
3 * ConjectureGeneralSymbolWeight (
    PreferNonGoals,200,100,200,50,50,1,100,1.5,1.5,1),
1 * Clauseweight (PreferProcessed,1,1,1),
1 * FIFOWeight (PreferProcessed)
```

The E strategy with longest specification in May 2014

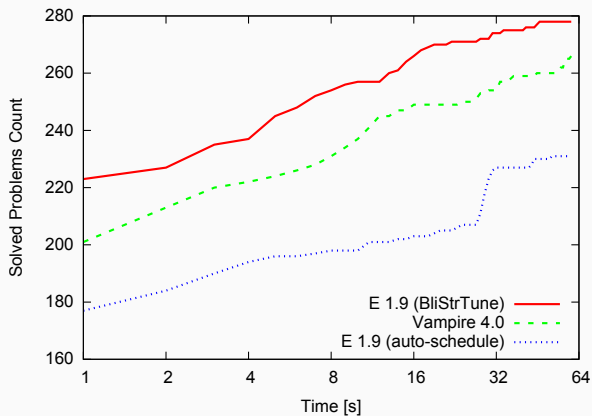
atpstr_my_c7bb78cc4c665670e6b866a847165cb4bf997f8a:

```
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)
```

BliStr on 1000 Mizar@Turing training problems



BliStr on 400 Mizar@Turing testing problems



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 from 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:

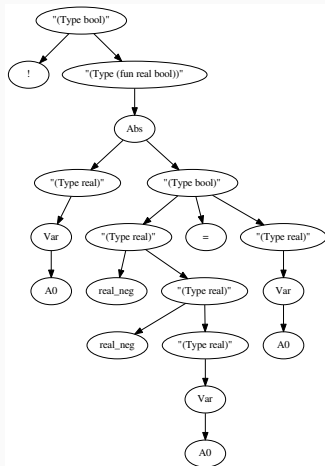
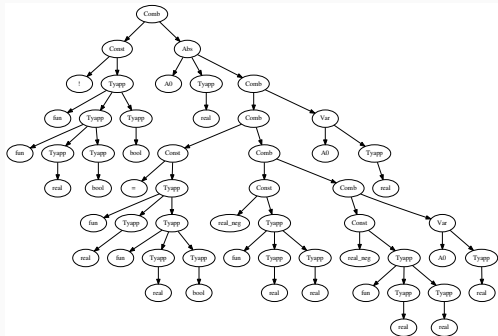
• REAL_NEGNEG: $\forall x. - -x = x$

```
(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)

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
- Probabilistic conjecturing as a by-product

Thanks

- Thanks for your attention!
- If interested, come to AITP: <http://aitp-conference.org>
- ATP/ITP/Math vs AI/Machine-Learning people, Computational linguists