Machine Learning for Theorem Proving

Lecture 9 (VU)

Cezary Kaliszyk

Overview

Last Lecture

- deep learning for premise selection
- negative mining, definition embeddings
- state estimation

Today

- deep learning for E-prover
- Enigma

What about learning for ATPs?

Most ATPs work as follows

Proof by contradiction

- Assume that the conjecture does not hold
- Derive that axioms and negated conjecture imply $\perp$

The main process is:

Saturation

- Convert problem to CNF (last lecture)
- Enumerate the consequences of the available clauses
- Goal: get to the empty clause

Additionally to avoid explosion we deal with

Redundancies

Simplify or eliminate some clauses (contract)

Calculus improvements (sources of inefficiency)

- Consider the clause $(a \vee b) \wedge \neg a \wedge \neg b$
- Two ways to derive a contradiction: we first resolve on a then on b or the other way around
- Orders allow specifying that a must come before be (or the other way around)
- This still remains sound and complete.
- Second source of inefficiency comes from the handing of equality

Calculus improvement (formally)

Resolution

$$\frac{C \vee A \quad D \vee \neg B}{(C \vee D)\sigma} \qquad \frac{C \vee A \vee B}{(C \vee A)\sigma}$$

Calculus improvement (formally)

Ordered Resolution

$$\frac{C \vee A \quad D \vee \neg B}{(C \vee D)\sigma} \qquad \frac{C \vee A \vee B}{(C \vee A)\sigma}$$

$A\sigma$ strictly maximal wrt $C\sigma$ and B maximal wrt $D\sigma$.

Calculus improvement (formally)

Ordered Resolution

$$\frac{C \vee A \quad D \vee \neg B}{(C \vee D)\sigma} \qquad \frac{C \vee A \vee B}{(C \vee A)\sigma}$$

$A\sigma$ strictly maximal wrt $C\sigma$ and B maximal wrt $D\sigma$.

Equality axioms?

Paramodulation

$$\frac{C \vee s \neq s'}{C\sigma'} \qquad \frac{C \vee s = t \quad D \vee L[s']}{(C \vee D \vee L[t])\sigma'}$$

Calculus improvement (formally)

Ordered Resolution

$$\frac{C \vee A \quad D \vee \neg B}{(C \vee D)\sigma} \qquad \frac{C \vee A \vee B}{(C \vee A)\sigma}$$

$A\sigma$ strictly maximal wrt $C\sigma$ and B maximal wrt $D\sigma$.

Equality axioms?

Ordered Paramodulation

$$\frac{C \vee s \neq s'}{C\sigma'} \qquad \frac{C \vee s = t \quad D \vee L[s']}{(C \vee D \vee L[t])\sigma'}$$

$(s = t)\sigma$ and $L[s']\sigma'$ maximal in their clauses.

Further calculus improvements

Equalities are no longer instantiated blindly, but we still rewrite in both directions

Given an equality like $x + 0 = x$ we can realize that actually removing all the $... + 0...$ and $...0 + ..$ is safe, and we do not need to add $... + 0$ anywhere.

A procedure that given a set of equalities finds a set of oriented equalities, such that we only need to replace left sides by right sides is called completion. Completion will be tried on some small set of equalities in the practical session, here we only give the rules for it.

Completion

$\mathcal{E}_0$ $\quad\quad$ $>$

Completion Procedure

$$(\mathcal{E}_0, \varnothing) \vdash (\mathcal{E}_1, \mathcal{R}_1) \vdash (\mathcal{E}_2, \mathcal{R}_2) \vdash (\mathcal{E}_3, \mathcal{R}_3) \vdash \cdots$$

deduce $\dfrac{(\mathcal{E}, \mathcal{R})}{(\mathcal{E} \cup \{s \approx t\}, \mathcal{R})}$ if $s \;{}_{\mathcal{R}}\!\leftarrow u \rightarrow_{\mathcal{R}} t$

delete $\dfrac{(\mathcal{E} \cup \{s \approx s\}, R)}{(\mathcal{E}, \mathcal{R})}$

orient $\dfrac{(\mathcal{E} \cup \{s \mathrel{\dot{\approx}} t\}, \mathcal{R})}{(\mathcal{E}, \mathcal{R} \cup \{s \rightarrow t\})}$ if $s > t$

compose $\dfrac{(\mathcal{E}, \mathcal{R} \cup \{s \rightarrow t\})}{(\mathcal{E}, \mathcal{R} \cup \{s \rightarrow u\})}$ if $t \rightarrow_{\mathcal{R}} u$

simplify $\dfrac{(\mathcal{E} \cup \{s \mathrel{\dot{\approx}} t\}, \mathcal{R})}{(\mathcal{E} \cup \{u \mathrel{\dot{\approx}} t\}, \mathcal{R})}$ if $s \rightarrow_{\mathcal{R}} u$

collapse $\dfrac{(\mathcal{E}, \mathcal{R} \cup \{s \rightarrow t\})}{(\mathcal{E} \cup \{u \approx t\}, \mathcal{R})}$ if $s \xrightarrow{\sqsupset}_{\mathcal{R}} u$

Superposition Calculus

Current most efficient theorem provers combine ordered paramodulation with completion. The calculus that combines these is called the superposition calculus and is the basis for (among others):

Current ATP provers

- E, Vampire, Spass, Prover9, Metis

Beyond the Calculus

Given a clause which has a literal and its negation, it does not bring in any information:

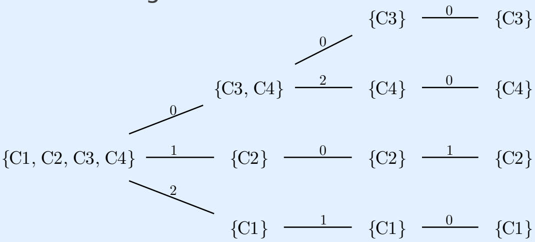
Tautology Detection (and deletion)

$$a \vee b \vee \neg a \vee d$$

Clauses that less general than ones derived in the past

Subsumption (forward and backward)

e.g. E uses Feature Vector Indexing



Still provers often fail deriving billions of clauses and no empty one...

```
fof(6, axiom,![X1]:![X2]:![X4]:gg(X1,sup_sup(X1,X2,X4)),file('i/f/1/goal_138__Q_Restricted_Rewriting.qrstep
fof(32, axiom,![X1]:![X2]:gg(set(product_prod(X1,X1)),transitive_rtrancl(X1,X2)),file('i/f/1/goal_138__Q_Re
fof(55, axiom,![X1]:![X19]:![X20]:(member(product_prod(X1,X1),X19,X20)=>member(product_prod(X1,X1),X19,tran
fof(68, axiom,![X1]:![X5]:![X3]:![X36]:![X20]:![X37]:![X16]:(ord_less_eq(set(product_prod(X1,X3)),X36,X20)=
fof(70, axiom,![X1]:![X20]:transitive_rtrancl(X1,transitive_rtrancl(X1,X20))=transitive_rtrancl(X1,X20),fil
fof(74, axiom,![X1]:![X24]:![X34]:![X33]:((~(member(X1,X24,X34))=>member(X1,X24,X33))=>member(X1,X24,sup_su
fof(78, axiom,![X1]:![X11]:![X13]:transitive_rtrancl(X1,sup_sup(set(product_prod(X1,X1)),transitive_rtrancl
fof(79, axiom,![X1]:![X22]:![X39]:(member(X1,X22,collect(X1,X39))<=>pp(aa(X1,bool,X39,X22))),file('i/f/1/go
fof(85, axiom,![X1]:(semilattice_sup(X1)=>![X23]:![X24]:![X22]:(ord_less_eq(X1,sup_sup(X1,X23,X24),X22)<=>(
fof(86, axiom,![X1]:![X11]:relcomp(X1,X1,X1,transitive_rtrancl(X1,X11),transitive_rtrancl(X1,X11))=transiti
fof(98, axiom,![X1]:![X33]:![X34]:(gg(set(X1),X34)=>(ord_less_eq(set(X1),X33,X34)<=>sup_sup(set(X1),X33,X34
fof(99, axiom,![X1]:![X33]:![X34]:ord_less_eq(set(X1),X33,sup_sup(set(X1),X33,X34)),file('i/f/1/goal_138__Q
fof(100, axiom,![X3]:![X1]:supteq(X1,X3)=sup_sup(set(product_prod(term(X1,X3),term(X1,X3))),supt(X1,X3),id(
fof(102, axiom,![X1]:![X34]:![X33]:ord_less_eq(set(X1),X34,sup_sup(set(X1),X33,X34)),file('i/f/1/goal_138__
fof(103, axiom,![X1]:![X33]:![X18]:![X34]:(ord_less_eq(set(X1),X33,X18)=>(ord_less_eq(set(X1),X34,X18)=>ord
fof(109, axiom,![X1]:![X34]:![X33]:(gg(set(X1),X33)=>(ord_less_eq(set(X1),X34,X33)=>sup_sup(set(X1),X33,X34
fof(114, axiom,![X1]:![X33]:![X18]:![X34]:![X48]:(ord_less_eq(set(X1),X33,X18)=>(ord_less_eq(set(X1),X34,X4
fof(116, axiom,![X1]:![X33]:ord_less_eq(set(X1),X33,X33),file('i/f/1/goal_138__Q_Restricted_Rewriting.qrste
fof(125, axiom,![X1]:![X24]:![X33]:![X34]:(member(X1,X24,X33)=>(~(member(X1,X24,X34))=>member(X1,X24,minus_
fof(127, axiom,![X1]:![X24]:![X33]:![X34]:(member(X1,X24,minus_minus(set(X1),X33,X34))=>~((member(X1,X24,X3
fof(131, axiom,![X1]:![X33]:(gg(set(X1),X33)=>collect(X1,aTP_Lamp_a(set(X1),fun(X1,bool),X33))=X33),file('i
fof(134, axiom,![X1]:(order(X1)=>![X35]:![X49]:((gg(X1,X35)&gg(X1,X49))=>(ord_less_eq(X1,X35,X49)=>(ord_les
fof(136, axiom,![X1]:(preorder(X1)=>![X35]:![X49]:![X50]:(ord_less_eq(X1,X35,X49)=>(ord_less_eq(X1,X49,X50)
fof(143, axiom,![X1]:![X33]:![X34]:(ord_less_eq(set(X1),X33,X34)<=>![X52]:(gg(X1,X52)=>(member(X1,X52,X33)=
fof(160, axiom,![X1]:![X39]:![X35]:![X33]:(pp(aa(X1,bool,X39,X35))=>(member(X1,X35,X33)=>?[X30]:(gg(X1,X30)
fof(171, axiom,![X1]:![X65]:![X66]:(pp(aa(X1,bool,aTP_Lamp_a(set(X1),fun(X1,bool),X65),X66))<=>member(X1,X6
fof(186, axiom,![X67]:semilattice_sup(set(X67)),file('i/f/1/goal_138__Q_Restricted_Rewriting.qrsteps_comp_s
fof(187, axiom,![X67]:preorder(set(X67)),file('i/f/1/goal_138__Q_Restricted_Rewriting.qrsteps_comp_supteq_s
fof(188, axiom,![X67]:order(set(X67)),file('i/f/1/goal_138__Q_Restricted_Rewriting.qrsteps_comp_supteq_subs
fof(207, conjecture,ord_less_eq(set(product_prod(term(a,b),term(a,b))),relcomp(term(a,b),term(a,b),term(a,b
```

Still the search space is huge: Can we use learning?

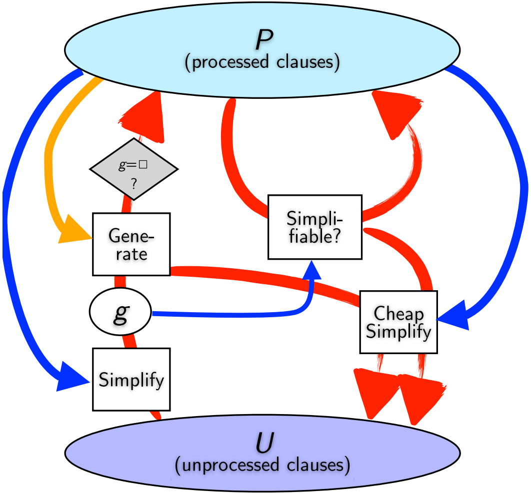
What has been tried

- Strategies: Which strategy to use for which problem (ordering, ...)
- Hints: Which clauses are specially interesting and we should aim for them instead of the conjecture first
- Premise selection can remove some of the axioms before the translation to CNF

What can be chosen in core of the Superposition calculus itself?

- Term ordering
- (Negative) literal selection
- Clause selection

E-Prover given-clause loop



The author of E prover claims that the most important choice: unprocessed clause selection

[*Schulz 2015*]

Learning for E: Data Collection

Dataset is based on Mizar top-level theorems *[Urban 2006]*

- Encoded in FOF

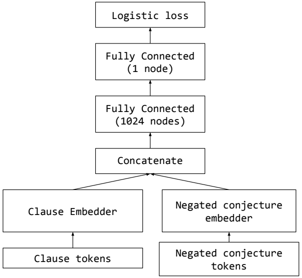
32,521 Mizar theorems with ≥ 1 proof

- training-validation split (90%-10%)
- replay with one strategy

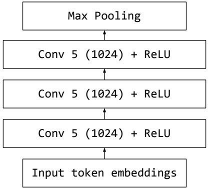
Collect all CNF intermediate steps

- and unprocessed clauses when a proof is found

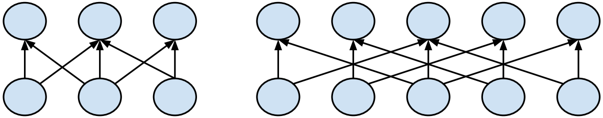
Deep Network Architectures



Overall network



Convolutional Embedding



Non-dilated and dilated convolutions

Recursive Neural Networks

- Curried representation of first-order statements
- Separate nodes for `apply`, `or`, `and`, `not`
- Layer weights learned jointly for the same formula
- Embeddings of symbols learned with rest of network
- Tree-RNN and Tree-LSTM models[1]

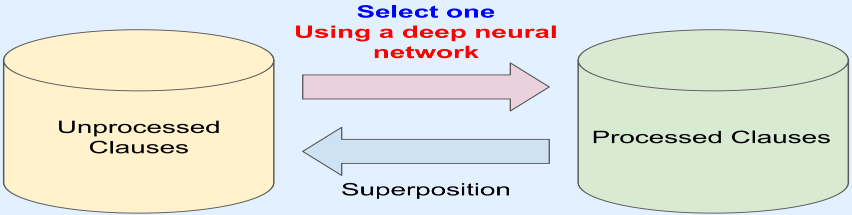
[1] Note that these are related to features originating from term graphs

Model accuracy

Model	Embedding Size	Accuracy: 50-50% split
Tree-RNN-256$\times$2	256	77.5%
Tree-RNN-512$\times$1	256	78.1%
Tree-LSTM-256$\times$2	256	77.0%
Tree-LSTM-256$\times$3	256	77.0%
Tree-LSTM-512$\times$2	256	77.9%
CNN-1024$\times$3	256	80.3%
$\star$CNN-1024$\times$3	256	78.7%
CNN-1024$\times$3	512	79.7%
CNN-1024$\times$3	1024	79.8%
WaveNet-256$\times$3$\times$7	256	79.9%
$\star$WaveNet-256$\times$3$\times$7	256	79.9%
WaveNet-1024$\times$3$\times$7	1024	81.0%
WaveNet-640$\times$3$\times$7(20%)	640	**81.5**%
$\star$WaveNet-640$\times$3$\times$7(20%)	640	79.9%

Improving Proof Search inside E

Overview

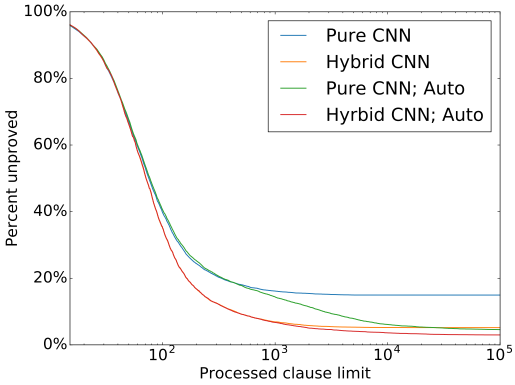


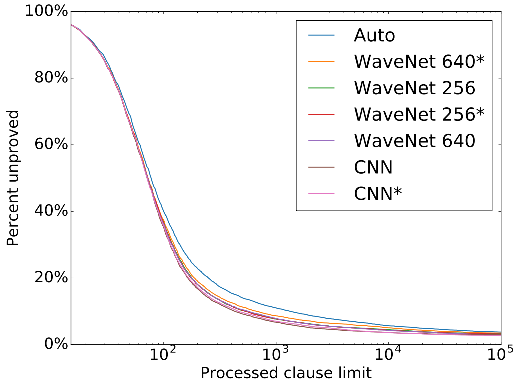
Problem

- Deep neural network evaluation is slow
- Slower than combining selected clause with all processed clauses[2]
- Solution 1: Batching clauses (evaluate as many clauses as possible at a time)
- Solution 2: Combining the neural heuristic with auto

[2] State of 2016

Hybrid heuristic





Overview

- Definitely better than the best E-prover heuristic (auto), especially after 200–1000 steps. But then the difference flattens out. So actually switching to the default heuristic later might make sense.

Harder Mizar top-level statements

DeepMath 1 = neural premise selection
DeepMath 2 = neural clause guidance

Model	**DeepMath 1**	**DeepMath 2**	**Union of 1 and 2**
Auto	578	581	674
⋆WaveNet 640	644	612	767
⋆WaveNet 256	692	712	864
WaveNet 640	629	685	997
⋆CNN	905	812	1,057
CNN	839	935	1,101
Total (unique)	1,451	1,458	1,712

Overall proved 7.4% of the harder statements

Harder Mizar top-level statements

DeepMath 1 = neural premise selection
DeepMath 2 = neural clause guidance

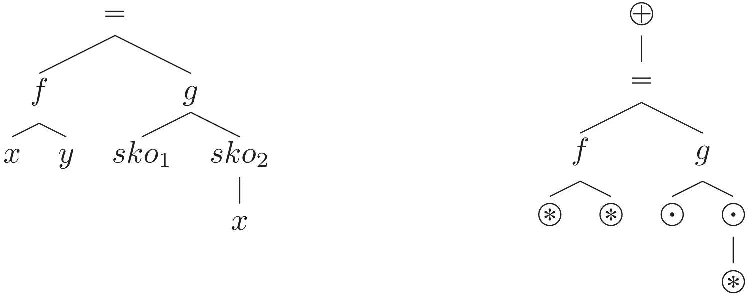
Model	DeepMath 1	DeepMath 2	Union of 1 and 2
Auto	578	581	674
⋆WaveNet 640	644	612	767
⋆WaveNet 256	692	712	864
WaveNet 640	629	685	997
⋆CNN	905	812	1,057
CNN	839	935	1,101
Total (unique)	1,451	1,458	1,712

Overall proved 7.4% of the harder statements

- Somewhat better than the best human defined heuristics, but it is actually the fact that it is complementary that gives new solved problems

ENIGMA

An alternative to deep-network guided E-prover has been developed by Jakubuv and Urban. There, path features and fast random-forest based predictors were shown to significantly improve E's best strategy in reasonable time:



- Evaluation on AIM, 30 seconds
- Single best strategy: 239
- Combination of E's strategies: 261
- Best trained strategy: 318 (includes prediction time)
- Different trained models: 337

Additional Literature (not required)

- Karel Chvalovský, Jan Jakubuv, Martin Suda, and Josef Urban.
  ENIGMA-NG: efficient neural and gradient-boosted inference guidance for E.
  In Pascal Fontaine, editor, *Automated Deduction - CADE 27 - 27th International Conference on Automated Deduction, Natal, Brazil, August 27-30, 2019, Proceedings*, volume 11716 of *Lecture Notes in Computer Science*, pages 197–215. Springer, 2019.
- Sarah M. Loos, Geoffrey Irving, Christian Szegedy, and Cezary Kaliszyk.
  Deep network guided proof search.
  In Thomas Eiter and David Sands, editors, *LPAR-21, 21st International Conference on Logic for Programming, Artificial Intelligence and Reasoning, Maun, Botswana, May 7-12, 2017*, volume 46 of *EPiC Series in Computing*, pages 85–105. EasyChair, 2017.
- Stephan Schulz.
  Simple and efficient clause subsumption with feature vector indexing.
  In Maria Paola Bonacina and Mark E. Stickel, editors, *Automated Reasoning and Mathematics - Essays in Memory of William W. McCune*, volume 7788 of *Lecture Notes in Computer Science*, pages 45–67. Springer, 2013.

Summary

This Lecture

- learning for superposition calculus
- E-prover
- Enigma

Next

- Tableaux and learning for tableaux
- Reinforcement learning in theorem proving
- State evaluation