Learning Precedences from Simple Symbol Features[1]

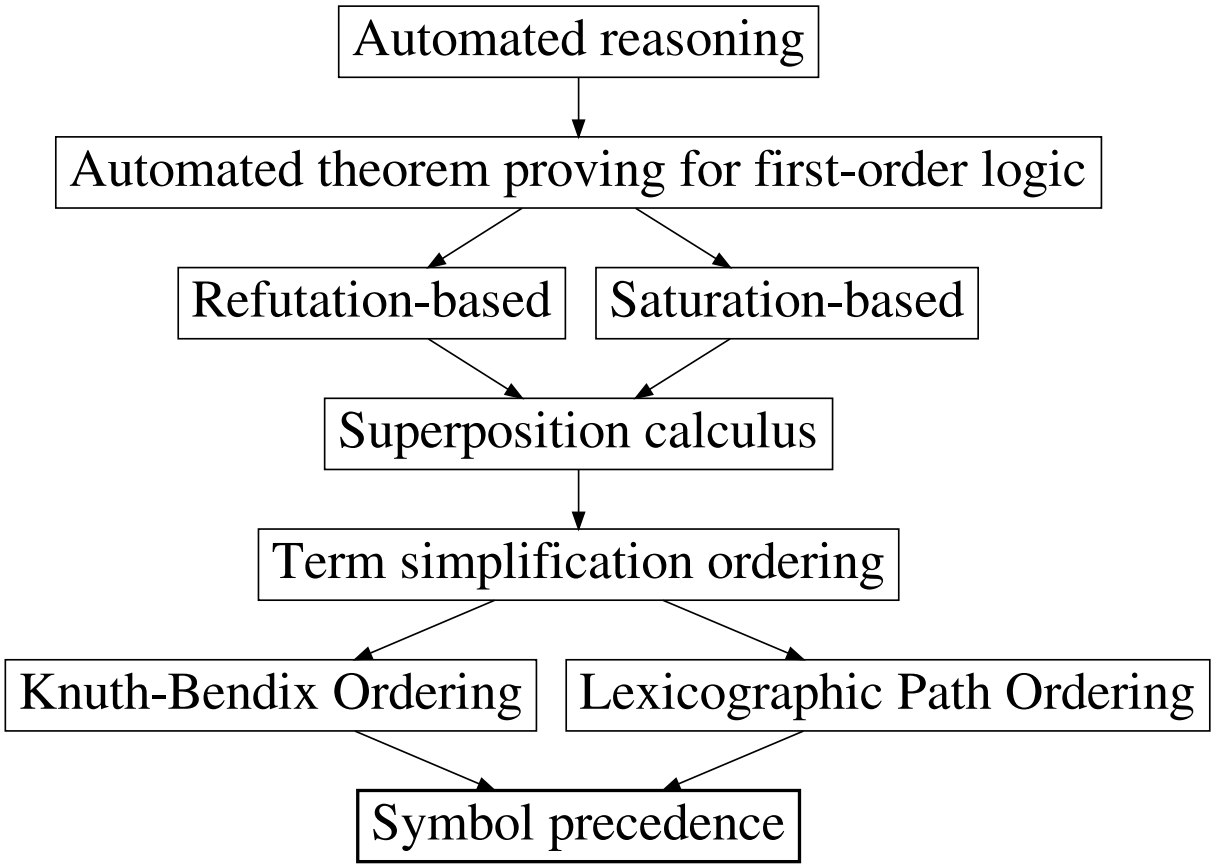
Filip Bártek Martin Suda

Czech Technical University in Prague, Czech Republic

November 4, 2020

[1]Supported by the ERC Consolidator grant AI4REASON no. 649043 under the EU-H2020 programme and the Czech Science Foundataion project 20-06390Y.

Context


Automated reasoning
Automated theorem proving for first-order logic
Refutation-based
Saturation-based
Superposition calculus
Term simplification ordering
Knuth-Bendix Ordering
Lexicographic Path Ordering
Symbol precedence

Basic concepts

Concept	Notation	Example
First-order problem	$P = (\Sigma, Cl)$	
Symbols	$\Sigma = (s_1, s_2, \ldots, s_n)$	
Symbol precedence	$\pi_P \sim Perm(\Sigma)$	$(0, 3, 1, 2)$
Order matrix	$O(\pi_P)$	$\begin{bmatrix} 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 \end{bmatrix}$
Flattened matrix	$\overrightarrow{O(\pi_P)}$	0111001000000110
Abstract solving time	$ast(P, \pi_P)$	

Main research question

Given a problem for a prover to solve,
how can we propose a good symbol precedence
using simple symbol features?

Architecture overview

Training:

1. For each training problem P in isolation:
   1. Try to solve P with 100 random precedences using Vampire
   2. Assign a cost to each of the precedences
   3. Assign a cost to each symbol pair by
      1. Fitting a linear regressor of precedence cost
      2. Extracting feature coefficients (feature $\sim$ symbol pair)
2. Train a symbol pair cost regressor
   - Represent each symbol by simple syntactic features

Proposing a precedence on a new problem:

1. Predict the cost of every symbol pair
2. Construct a precedence that minimizes the cumulative symbol pair cost

Precedence cost

For each problem $P \in \mathcal{P}_{train}$:

1. For 100 uniformly random precedences π_P:
   Run Vampire and measure $ast(P, \pi_P)$.
2. Assign a penalty to precedences that fail to solve the problem (namely by timing out).
   - Penalty: Maximum $ast(P, \pi_P)$ over successful runs on P
3. Normalize the costs:
   1. Log-scale
   2. Standardize (ensure mean 0 and standard deviation 1)

Let $cost_{std}(\pi_P)$ be the normalized cost of π_P.

Symbol pair cost

For each problem $P \in \mathcal{P}_{train}$, train a linear model:

- Input: $O(\pi_P)$ – order matrix of a precedence

  Example: $\pi_P = (0, 3, 1, 2), O(\pi_P) = \begin{bmatrix} 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 \end{bmatrix}$
- Target: $cost_{std}(\pi_P)$ – normalized cost of π_P
- Regression method: Lasso (linear regression with L_1 penalty)
- Trained model coefficients: W_P – preference matrix
  $W_{P\,i,j} \sim$ cost attributed to $[\![s_i$ is ordered before $s_j]\!]$
- Prediction: $cost_{proxy}(\pi_P, W_P) = \overrightarrow{O(\pi_P)} \cdot \overrightarrow{W_P}$

Symbol features

6 simple features:

- Arity
- Frequency (number of occurrences)
- Number of clauses that contain the symbol
- Is it in conjecture?
- Is it in a unit clause?
- Is it introduced (Tseitin or Skolem)?

Feature vector of symbol s_i: $fv(s_i)$

Symbol pair cost model M

Generalizing across problems

- Input: feature vector $[fv(s_i), fv(s_j)]$
  of a symbol pair s_i, s_j from a problem $P \in \mathcal{P}_{train}$
- Target: $W_{P\,i,j}$ (cost attributed to $[\![s_i$ is ordered before $s_j]\!]$)
- Training sample distribution:
  1. Sample problems from $\mathcal{P}_{train}$ uniformly.
  2. Given a problem, weight symbol pairs by absolute target value.
- Learning method:
  - Elastic-Net (linear regression with L_1 and L_2 penalties)
  - Gradient Boosting Machine
- Prediction: $M([fv(s_i), fv(s_j)])$

Proposing a good precedence

Given a previously unseen problem P:

1. Estimate a preference matrix $\widehat{W_P}$.
   $\widehat{W_P}_{i,j} = M([fv(s_i), fv(s_j)])$.
2. Construct a precedence $\widehat{\pi_P}$ that approximately minimizes
   $cost_{proxy}(\widehat{\pi_P}, \widehat{W_P}) = \overrightarrow{O(\widehat{\pi_P})} \cdot \overrightarrow{\widehat{W_P}}$.
   - NP-hard in general
   - 2-approximation greedy algorithm by Cohen et al. [1]:
     Repeatedly append symbol with the smallest potential:
     $c(s_i) = \sum_{s_j \in \Sigma_{avail}} \widehat{W_P}_{i,j} - \sum_{s_j \in \Sigma_{avail}} \widehat{W_P}_{j,i}$

Experimental evaluation

- Learning only predicate precedences
- 5 evaluation iterations, each using:
  - 1000 training problems
  - 1000 test problems

Case	Successes out of 1000	
	Mean	Std
Best of 10 random	501.8	13.85
`invfreq`	475.8	13.83
Elastic-Net	471.6	10.21
Gradient Boosting	464.8	13.78
Elastic-Net without weighting	461.2	8.11
Random	450.4	10.25

Elastic-Net feature coefficients

of symbol pairs

Split	Left symbol			Right symbol		
	Arity	Freq.	Unit freq.	Arity	Freq.	Unit freq.
0		−0.01		−0.98	0.01	
1		−0.48			0.08	0.44
2			−0.64		0.36	
3	0.88	−0.03	0.01		−0.03	0.05
4		−0.62			0.30	0.07
$\mathcal{P}_{train}$			−0.57		0.43	

The other 3 features have all zero coefficients.

- Training sets 1, 2, 4, $\mathcal{P}_{train}$:
  Similar to `invfreq` heuristic (low frequency $\sim$ early inference)
- Training sets 0, 3:
  Similar to arity heuristic (high arity $\sim$ early inference)

Summary

We designed a system that:

- Trains on random precedences using preference matrices
- Proposes a good precedence using a greedy algorithm

We evaluated the system on TPTP for predicate precedences.

Future:

- Stabilize training
- Evaluate also for function symbol precedences
- Richer symbol representation using graph neural networks

Section 5

Bonus slides

References

[1] William W. Cohen, Robert E. Schapire, and Yoram Singer. Learning to order things. *CoRR*, abs/1105.5464, 2011. URL `http://arxiv.org/abs/1105.5464`.

[2] G. Sutcliffe. The TPTP Problem Library and Associated Infrastructure. From CNF to TH0, TPTP v6.4.0. *Journal of Automated Reasoning*, 59 (4):483–502, 2017.

Experimental setup

- Only predicate precedences are learned.
  Function symbols are ordered by `invfreq`.
- Problems from TPTP [2] – CNF and FOF (clausified with Vampire)
  - $\mathcal{P}_{train}$ (8217 problems): at most 200 predicate symbols, at least 1 out of 24 random predicate precedences yield success
  - $\mathcal{P}_{test}$ (15751 problems): at most 1024 predicate symbols
- 5 evaluation iterations (splits): 1000 training problems and 1000 test problems
- 100 precedences per training problem
- Vampire configuration: time limit: 10 seconds, memory limit: 8192 MB, literal comparison mode: predicate, function symbol precedence: `invfreq`, saturation algorithm: discount, age-weight ratio: 1:10, AVATAR: disabled
- 10^6 symbol pair samples to train M

Elastic-Net feature coefficients

of individual symbols

Training set	Arity	Frequency	Unit frequency
0	−.98	.01	−.01
1		.56	.44
2		.36	.64
3	−.88		.04
4		.93	.07
$\mathcal{P}_{train}$		.43	.57

Symbol order: descending by predicted value

- Sets 1, 2, 4, $\mathcal{P}_{train}$:
  - Descending by frequency: low frequency $\sim$ early inference
  - Similar to `invfreq` and `vampire --sp frequency`
- Sets 0, 3:
  - Ascending by arity: high arity $\sim$ early inference
  - Similar to `vampire --sp arity`