Learning Symbol Weights for Clause Selection

Filip Bártek (filip.bartek@cvut.cz) and Martin Suda

Czech Institute of Informatics, Robotics and Cybernetics

April 19, 2023

This work was supported by the Czech Science Foundation project no. 20-06390Y (JUNIOR grant), the European Regional Development Fund under the Czech project AI&Reasoning no. CZ.02.1.01/0.0/0.0/15_003/00004, the project RICAIP no. 857306 under the EU-H2020 programme, and the Grant Agency of the Czech Technical University in Prague, grant no. SGS20/215/OHK3/3T/37.

Saturation-based theorem proving

Input: Problem in clause normal form (first-order logic clauses)

Proof search state – two sets of clauses:

- Passive
- Active

Saturation loop:

1. Select clause C from Passive.
2. Perform all inferences between C and Active.
   - Add the generated clauses to Passive.
   - If the empty clause is generated, terminate.
3. Move C from Passive to Active.

Saturation-based theorem proving

Input: Problem in clause normal form (first-order logic clauses)

Proof search state – two sets of clauses:

- Passive
- Active

Saturation loop:

1. Select clause C from Passive.
2. Perform all inferences between C and Active.
   - Add the generated clauses to Passive.
   - If the empty clause is generated, terminate.
3. Move C from Passive to Active.

Saturation-based theorem proving

Input: Problem in clause normal form (first-order logic clauses)

Proof search state – two sets of clauses:

- Passive
- Active

Saturation loop:

1. Select clause C from Passive.
2. Perform all inferences between C and Active.
   - Add the generated clauses to Passive.
   - If the empty clause is generated, terminate.
3. Move C from Passive to Active.

Saturation-based theorem proving

Input: Problem in clause normal form (first-order logic clauses)

Proof search state – two sets of clauses:

- Passive
- Active

Saturation loop:

1. Select clause C from Passive.
2. Perform all inferences between C and Active.
   - Add the generated clauses to Passive.
   - If the empty clause is generated, terminate.
3. Move C from Passive to Active.

Saturation-based theorem proving

Input: Problem in clause normal form (first-order logic clauses)

Proof search state – two sets of clauses:

- Passive
- Active

Saturation loop:

1. Select clause C from Passive.
2. Perform all inferences between C and Active.
   - Add the generated clauses to Passive.
   - If the empty clause is generated, terminate.
3. Move C from Passive to Active.

Saturation-based theorem proving

Input: Problem in clause normal form (first-order logic clauses)

Proof search state – two sets of clauses:

- Passive
- Active

Saturation loop:

1. Select clause C from Passive.
2. Perform all inferences between C and Active.
   - Add the generated clauses to Passive.
   - If the empty clause is generated, terminate.
3. Move C from Passive to Active.

Saturation-based theorem proving

Input: Problem in clause normal form (first-order logic clauses)

Proof search state – two sets of clauses:

- Passive
- Active

Saturation loop:

1. Select clause C from Passive.
2. Perform all inferences between C and Active.
   - Add the generated clauses to Passive.
   - If the empty clause is generated, terminate.
3. Move C from Passive to Active.

Saturation-based theorem proving

Input: Problem in clause normal form (first-order logic clauses)

Proof search state – two sets of clauses:

- Passive
- Active

Saturation loop:

1. Select clause C from Passive. – **Which one?**
2. Perform all inferences between C and Active.
   - Add the generated clauses to Passive.
   - If the empty clause is generated, terminate.
3. Move C from Passive to Active.

Clause selection by weight

Clause		Symbol and variable occurrences
C_1	$E(m(i, x_1), x_1)$	5
C_2	$\neg E(m(x_1, x_2), x_3) \vee P(x_1, x_2, x_3)$	9
$\vdots$	$\vdots$	$\vdots$

Machine learning for clause selection

How to train clause selection by machine learning?

Training data from a successful proof search:

- Proof clauses $\mathcal{C}_+$
- Nonproof selected clauses $\mathcal{C}_-$

Machine learning for clause selection

How to train clause selection by machine learning?

Training data from a successful proof search:

- Proof clauses $\mathcal{C}_+$
- Nonproof selected clauses $\mathcal{C}_-$

Generalized clause weight

Clause		Symbol and variable occurrences
C_-	$E(m(i, x_1), x_1)$	5
C_+	$\neg E(m(x_1, x_2), x_3) \vee P(x_1, x_2, x_3)$	9

Generalized clause weight

Clause		Occurrence count				
		x_*	E	P	m	i
C_-	$E(m(i, x_1), x_1)$	2	1	0	1	1
C_+	$\neg E(m(x_1, x_2), x_3) \vee P(x_1, x_2, x_3)$	6	1	1	1	0

Generalized clause weight

Clause	Occurrence count					Clause weight $W(C_*)$
	x_*	E	P	m	i	
C_-	2	1	0	1	1	$2w(x_*) + w(E) + w(m) + w(i)$
C_+	6	1	1	1	0	$6w(x_*) + w(E) + w(P) + w(m)$

Generalized clause weight

Clause	Occurrence count					Clause weight $W(C_*)$
	x_*	E	P	m	i	
C_-	2	1	0	1	1	$2w(x_*) + w(E) + w(m) + w(i)$
C_+	6	1	1	1	0	$6w(x_*) + w(E) + w(P) + w(m)$

$$W(C_+) < W(C_-)$$
$$4w(x_*) + w(P) < w(i)$$

Generalized clause weight

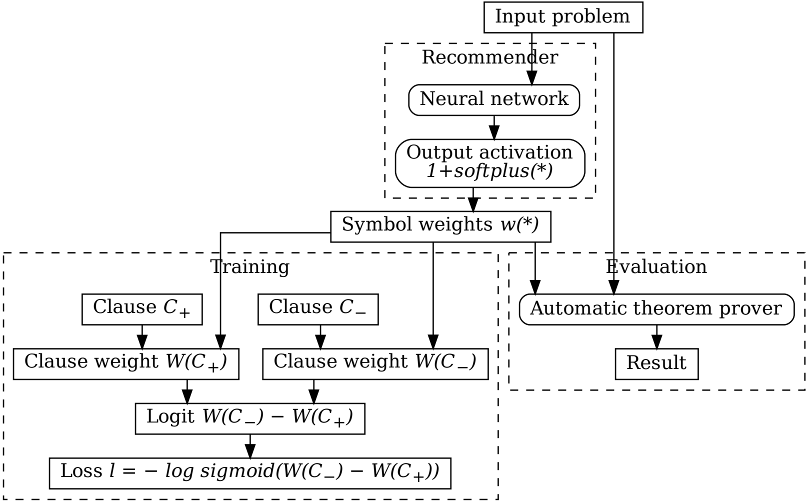
Clause	Occurrence count					Clause weight $W(C_*)$
	x_*	E	P	m	i	
C_-	2	1	0	1	1	$2w(x_*) + w(E) + w(m) + w(i)$
C_+	6	1	1	1	0	$6w(x_*) + w(E) + w(P) + w(m)$

$$W(C_+) < W(C_-)$$
$$4w(x_*) + w(P) < w(i)$$

Example solution:

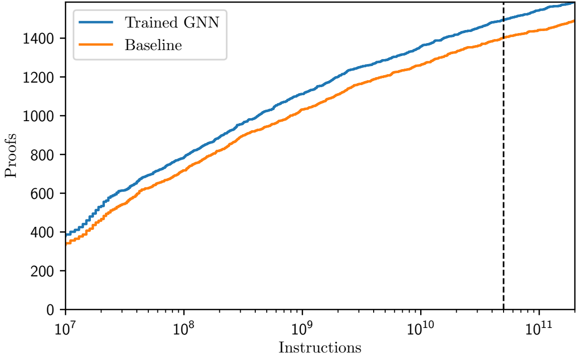
- $w(x_*) = 1$
- $w(P) = 1$
- $w(i) = 6$

Symbol weight recommender



Evaluation

Configuration	Proofs found on 3149 problems	
	Absolute	Relative
Trained GNN	1494	47.4 %
Baseline	1439	44.5 %



Summary

- Clause selection
  - Prover selects clause with the smallest weight
  - Clause weight parameterized by symbol weight
- Trained GNN recommends symbol weights
- Training
  - Training example: clause pair (proof and nonproof) from a successful proof search
  - Proxy task: clause ranking (clause pair classification)

Evaluation

Table: Results of the final empirical evaluation. The reported performance is the number of proofs found on the test set (3149 problems) within 5×10^{10} CPU instructions per proof search.

Configuration	Proofs found	
	Absolute	Relative
Trained graph neural network (GNN)	1494	47.4 %
Baseline	1439	44.5 %

Clause weight

Table: Examples of clauses and their symbol-counting weights

C	$W(C)$
$p(X_1, c, X_2) \vee q(X_1)$	$3w(X) + w(p) + w(q) + w(c)$
$g(X_1, h(X_2)) \approx f(g(X_1, X_2), X_1)$	$5w(X) + w(\approx) + w(f) + 2w(g) + w(h)$
$\neg(h(X_1) \approx h(X_2)) \vee X_1 \approx X_2$	$4w(X) + 2w(\approx) + 2w(h)$

Clause weight

$$W(C) = \sum_{s \in \Sigma \cup \{\approx, X\}} S_C(s) \cdot w(s)$$

Training

- Training example: Pair of clauses C_+ (proof) and C_- (nonproof)
- Proxy task: Clause pair classification
- Example likelihood: $p(C_+, C_-) = \text{sigmoid}(W(C_-) - W(C_+))$
  - p is large when $W(C_-)$ is large and $W(C_+)$ is small
- Loss: negative log-likelihood $\ell = -\log p(C_+, C_-)$

Symbol weight recommender

- Input: Problem
- Output: Variable and symbol weights
  - Output activation function: $a(x) = 1 + \mathrm{softplus}(x)$