CTU

CZECH TECHNICAL UNIVERSITY IN PRAGUE

How Much Should This Symbol Weigh? A GNN-Advised Clause Selection

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

Czech Technical University in Prague, Czech Republic

June 8, 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: Set of first-order logic clauses

Proof search state – two sets of clauses:

- Passive
- Active

Saturation loop:

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

Saturation-based theorem proving

Input: Set of first-order logic clauses

Proof search state – two sets of clauses:

- Passive
- Active

Saturation loop:

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

Saturation-based theorem proving

Input: Set of first-order logic clauses

Proof search state – two sets of clauses:

- Passive
- Active

Saturation loop:

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

Saturation-based theorem proving

Input: Set of first-order logic clauses

Proof search state – two sets of clauses:

- Passive
- Active

Saturation loop:

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

Saturation-based theorem proving

Input: Set of first-order logic clauses

Proof search state – two sets of clauses:

- Passive
- Active

Saturation loop:

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

Saturation-based theorem proving

Input: Set of first-order logic clauses

Proof search state – two sets of clauses:

- Passive
- Active

Saturation loop:

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

Saturation-based theorem proving

Input: Set of first-order logic clauses

Proof search state – two sets of clauses:

- Passive
- Active

Saturation loop:

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

Saturation-based theorem proving

Input: Set of first-order logic clauses

Proof search state – two sets of clauses:

- Passive
- Active

Saturation loop:

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

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
$\vdots$	$\vdots$	$\vdots$

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

We want clause weight function W such that:

$$W(C_+) < W(C_-)$$

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

We want clause weight function W such that:

$$W(C_+) < W(C_-)$$

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	$6w(x_*) + w(E) + w(P) + w(m)$

We want clause weight function W such that:

$$W(C_+) < W(C_-)$$

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	$6w(x_*) + w(E) + w(P) + w(m)$

We want symbol weight $w : \{x_*, E, P, m, i\} \to \mathbb{R}$ such that:

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

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	$6w(x_*) + w(E) + w(P) + w(m)$

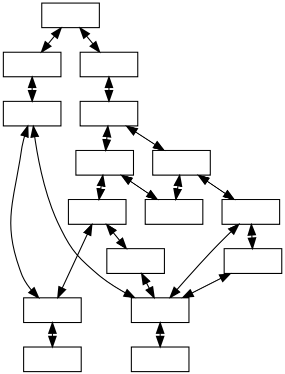
We want symbol weight $w : \{x_*, E, P, m, i\} \to \mathbb{R}$ such that:

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

Example solution w:

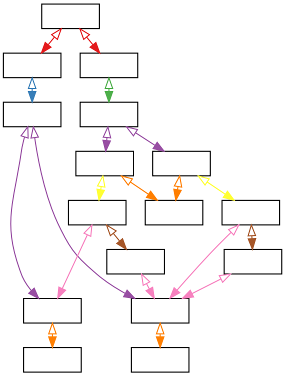
- $w(x_*) = 1$
- $w(P) = 1$
- $w(i) = 6$

Graph convolutional network (GCN)



$$h_d^{(l+1)} = \sigma\left(\sum_{s \in \mathcal{N}_d} \frac{1}{\sqrt{|\mathcal{N}_s|}\sqrt{|\mathcal{N}_d|}}\left(W^{(l)} h_s^{(l)} + b^{(l)}\right)\right)$$

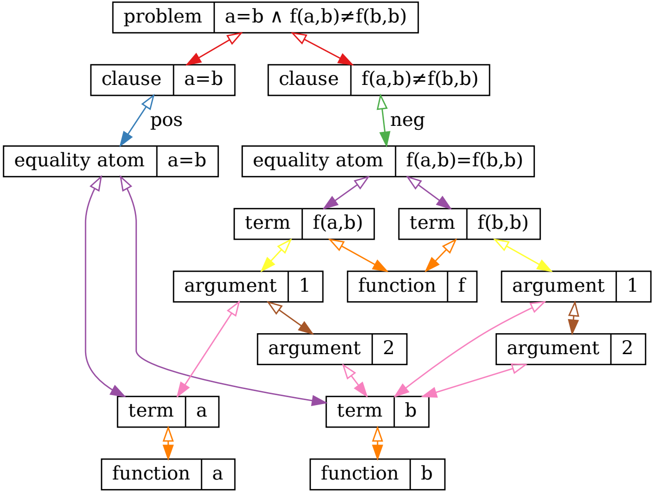
Relational graph convolutional network (R-GCN)



$$h_d^{(l+1)} = \sum_{r \in \mathcal{R}} \sigma\left(\sum_{s \in \mathcal{N}_d^r} \frac{1}{\sqrt{|\mathcal{N}_s^r|}\sqrt{|\mathcal{N}_d^r|}} (W_r^{(l)} h_s^{(l)} + b_r^{(l)})\right)$$

Graph representation of a CNF problem

Input problem: $a = b \wedge f(a, b) \neq f(b, b)$

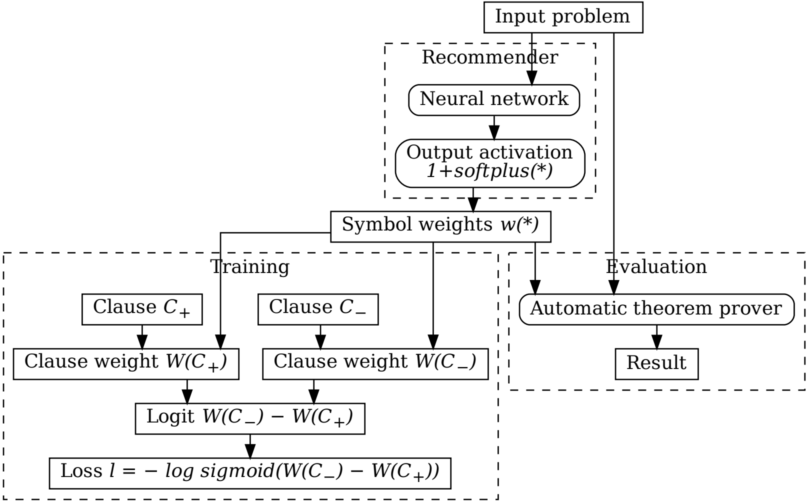


Learning from a successful proof search

Data from a successful proof search:

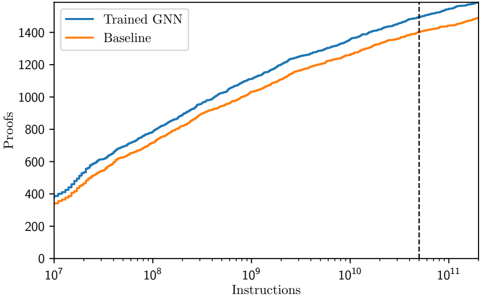
- Proof clauses $\mathcal{C}_+$
  - Ancestors of the empty clause in the inference graph
- Nonproof selected clauses $\mathcal{C}_-$

Symbol weight recommender



Evaluation

Configuration	Proofs found		Compared to B		
	/3149	%	+	−	%
Trained GNN	1494	47.4 %	+141	−49	+6.6 %
Baseline (B)	1402	44.5 %	+0	−0	+0.0 %
B + AVATAR	1485	47.2 %			+5.9 %
B + Goal-directed	1463	46.5 %			+4.4 %



Observations

- Variable weights should be small
- Forcing symbol weights to be positive prevents “vicious circles”

Summary

- Clause selection
  - Prover prioritizes 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)
- Strengths
  - One evaluation of GNN per proof search
  - Negligible computational overhead in proof search
  - Signature-agnostic recommender

Summary

- Clause selection
  - Prover prioritizes 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)
- Strengths
  - One evaluation of GNN per proof search
  - Negligible computational overhead in proof search
  - Signature-agnostic recommender

Thank you for your attention!

Appendix

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: $a(x) = 1 + \mathrm{softplus}(x)$

Graph convolutional network (GCN)

Initial embedding of node d:

$$h_d^{(0)} = (\text{feature vector}) \oplus (\text{trainable vector})$$

Feature vector:

- Clause: role (axiom, assumption, negated conjecture)
- Symbol: introduced in preprocessing, in conjecture

Propagation rule for layer l:

$$h_d^{(l+1)} = \sum_{r \in \mathcal{R}} \sigma \left(\sum_{s \in \mathcal{N}_d^r} \frac{1}{\sqrt{|\mathcal{N}_s^r|}\sqrt{|\mathcal{N}_d^r|}} (W_r^{(l)} h_s^{(l)} + b_r^{(l)}) \right)$$

Output activation function

Output activation function: $1 + \text{softplus}(*)$
Ensures each symbol weight is ≥ 1
Assigning s a negative weight causes an infinite chain:

1. $\neg P(X) \vee P(s(X))$
2. $P(0)$
3. $P(s(0))$
4. $P(s(s(0)))$
5. $P(s(s(s(0))))$

$\vdots$