Recommending Symbol Precedences by Machine Learning

Filip Bártek

Czech Technical University in Prague

November 23, 2022

Example

$$0 + s(s(0)) \neq s(s(0)) \qquad (NC)$$

$$x + 0 = x \qquad (A_0)$$

$$x + s(y) = s(x + y) \qquad (A_s)$$

$$x + s(y) \rightarrow s(x + y) \qquad (R_{+s})$$

$$x + s(y) \leftarrow s(x + y) \qquad (R_{s+})$$

$$\underline{0 + s(s(0))} \neq s(s(0)) \xrightarrow{R_{+s}} s(\underline{0 + s(0)}) \neq s(s(0)) \xrightarrow{R_{+s}} s(s(\underline{0 + 0})) \neq s(s(0)) \xrightarrow{R_0} s(s(0)) \neq s(s(0))$$

$$x + s(0) \rightarrow s(x) \qquad (R_1)$$

$$x + s(s(0)) \rightarrow s(s(x)) \qquad (R_2)$$

$$\vdots$$

$$x + s^n(0) \rightarrow s^n(x) \qquad (R_n)$$

$$\vdots$$

Example

$$0 + s(s(0)) \neq s(s(0)) \qquad (NC)$$

$$x + 0 \to x \qquad (R_0)$$

$$x + s(y) = s(x + y) \qquad (A_s)$$

$$x + s(y) \to s(x + y) \qquad (R_{+s})$$

$$x + s(y) \leftarrow s(x + y) \qquad (R_{s+})$$

$$\underline{0 + s(s(0))} \neq s(s(0)) \xrightarrow{R_{+s}}$$

$$s(\underline{0 + s(0)}) \neq s(s(0)) \xrightarrow{R_{+s}}$$

$$s(s(\underline{0 + 0})) \neq s(s(0)) \xrightarrow{R_0}$$

$$s(s(0)) \neq s(s(0))$$

$$x + s(0) \to s(x) \qquad (R_1)$$

$$x + s(s(0)) \to s(s(x)) \qquad (R_2)$$

$$\vdots$$

$$x + s^n(0) \to s^n(x) \qquad (R_n)$$

$$\vdots$$

Example

$$0 + s(s(0)) \neq s(s(0)) \qquad (NC)$$

$$x + 0 \to x \qquad (R_0)$$

$$x + s(y) = s(x + y) \qquad (A_s)$$

$$x + s(y) \to s(x + y) \qquad (R_{+s})$$

$$x + s(y) \leftarrow s(x + y) \qquad (R_{s+})$$

$$\underline{0 + s(s(0))} \neq s(s(0)) \xrightarrow{R_{+s}}$$

$$s(\underline{0 + s(0)}) \neq s(s(0)) \xrightarrow{R_{+s}}$$

$$s(s(\underline{0 + 0})) \neq s(s(0)) \xrightarrow{R_0}$$

$$s(s(0)) \neq s(s(0))$$

$$x + s(0) \to s(x) \qquad (R_1)$$

$$x + s(s(0)) \to s(s(x)) \qquad (R_2)$$

$$\vdots$$

$$x + s^n(0) \to s^n(x) \qquad (R_n)$$

$$\vdots$$

Context

Saturation-based automated theorem proving for first-order logic

Context

Saturation-based automated theorem proving for first-order logic

Context

Saturation-based automated theorem proving for first-order logic

Saturation-based theorem proving

Two sets of clauses:

- Passive
- Active

Saturation loop:

1. Select clause C from Passive
2. Perform inferences between C and all clauses in Active
   - Restricted by *simplification term ordering*
   - Add the newly inferred clauses to Passive
3. Move C from Passive to Active

Saturation-based theorem proving

Two sets of clauses:

- Passive
- Active

Saturation loop:

1. Select clause C from Passive
2. Perform inferences between C and all clauses in Active
   - Restricted by *simplification term ordering*
   - Add the newly inferred clauses to Passive
3. Move C from Passive to Active

Saturation-based theorem proving

Two sets of clauses:

- Passive
- Active

Saturation loop:

1. Select clause C from Passive
2. Perform inferences between C and all clauses in Active
   - Restricted by *simplification term ordering*
   - Add the newly inferred clauses to Passive
3. Move C from Passive to Active

Saturation-based theorem proving

Two sets of clauses:

- Passive
- Active

Saturation loop:

1. Select clause C from Passive
2. Perform inferences between C and all clauses in Active
   - Restricted by *simplification term ordering*
   - Add the newly inferred clauses to Passive
3. Move C from Passive to Active

Saturation-based theorem proving

Two sets of clauses:

- Passive
- Active

Saturation loop:

1. Select clause C from Passive
2. Perform inferences between C and all clauses in Active
   - Restricted by *simplification term ordering*
   - Add the newly inferred clauses to Passive
3. Move C from Passive to Active

Saturation-based theorem proving

Two sets of clauses:

- Passive
- Active

Saturation loop:

1. Select clause C from Passive
2. Perform inferences between C and all clauses in Active
   - Restricted by *simplification term ordering*
   - Add the newly inferred clauses to Passive
3. Move C from Passive to Active

Simplification term ordering

Affects:

- Ordered resolution
- Superposition inferences

Popular ordering schemes:

- Knuth-Bendix ordering
- Lexicographic path ordering

Specified by *symbol precedence*

Precedence recommender

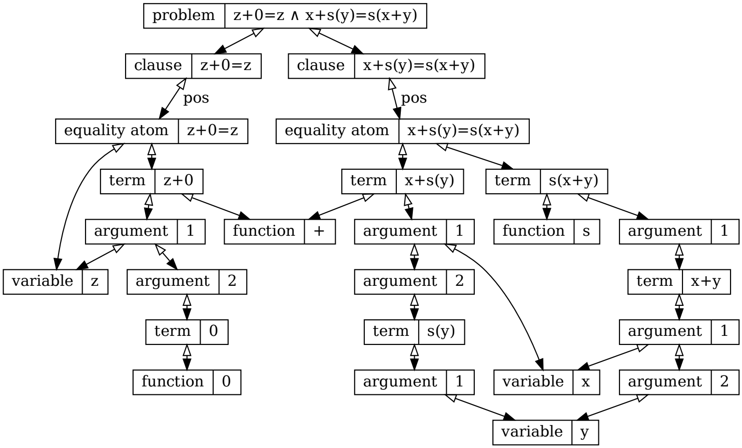
- Input: A first-order logic (FOL) problem in clause normal form (CNF)
- Output: A symbol precedence
    - Try to minimize the runtime.
- Target algorithm: An automated theorem prover (ATP)
    - A fixed configuration
    - Superposition calculus with ordered resolution
- Challenge: Unaligned signatures across problems
- State of the art: Simple heuristics (for example `invfreq`)

Neural precedence recommender

- Machine learning from ATP runs with random precedences
- Generalizes across problems
  - Symbol semantics
  - Signature length
- Approach:
  - ML core: Graph convolutional network (GCN)
  - Proxy task: Precedence pair classification

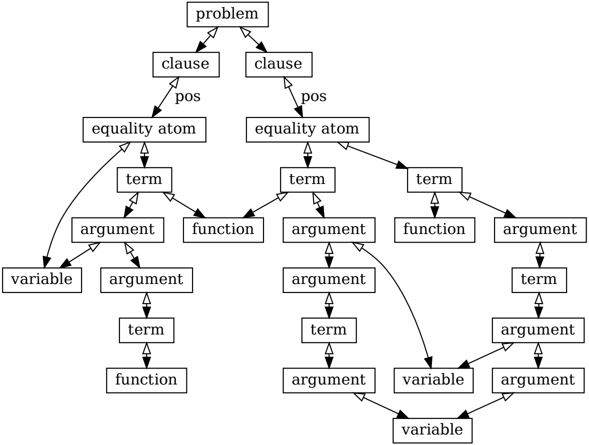
Graph representation of a CNF problem

Input problem: $z + 0 = z \wedge x + s(y) = s(x + y)$



Graph representation of a CNF problem

Input problem: $z + 0 = z \wedge x + s(y) = s(x + y)$



Graph convolutional network (GCN)

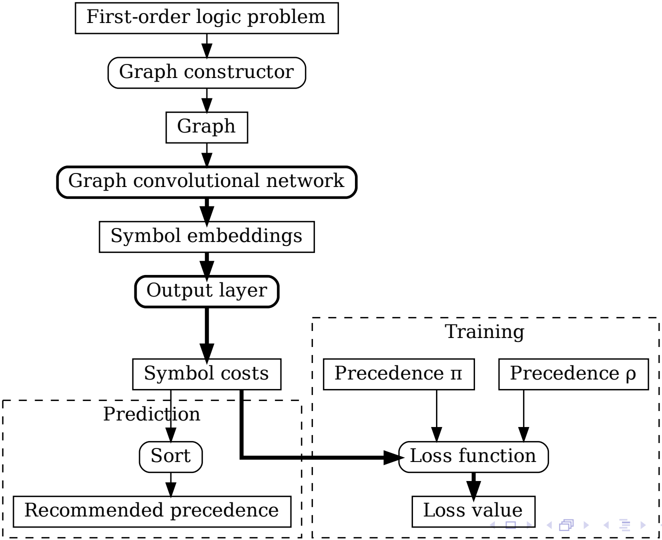
Initial embedding of node d:

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

Propagation rule for layer l:

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

Recommender architecture



Training data

Training example (P, π, ρ):

- Problem P
  - Sampled from the target distribution (for example TPTP FOL)
- Precedences π and ρ such that $\pi \prec_P \rho$
  - π solves P faster than ρ.
  - Given P, precedences are sampled uniformly.
  - Uncomparable precedence pairs are discarded.

Training data

Training example (P, π, ρ):

- Problem P
  - Sampled from the target distribution (for example TPTP FOL)
- Precedences π and ρ such that $\pi \prec_P \rho$
  - π solves P faster than ρ.
  - Given P, precedences are sampled uniformly.
  - Uncomparable precedence pairs are discarded.

Training data

Training example (P, π, ρ):

- Problem P
  - Sampled from the target distribution (for example TPTP FOL)
- Precedences π and ρ such that $\pi \prec_P \rho$
  - π solves P faster than ρ.
  - Given P, precedences are sampled uniformly.
  - Uncomparable precedence pairs are discarded.

Precedence cost

c_i is the predicted cost of the i-th symbol.

Cost of symbol precedence π over signature of length n

$$C(\pi) = \frac{2}{n(n+1)} \sum_{i=1}^{n} i \cdot c_{\pi(i)}$$

Lemma (Precedence cost minimization)

The precedence cost C is minimized by any precedence that sorts the symbols by their costs in non-increasing order:

$$\operatorname*{argmin}_{\rho \in \mathrm{Perm}(n)} C(\rho) = \mathrm{argsort}^{-}(c_1, \ldots, c_n)$$

Loss function

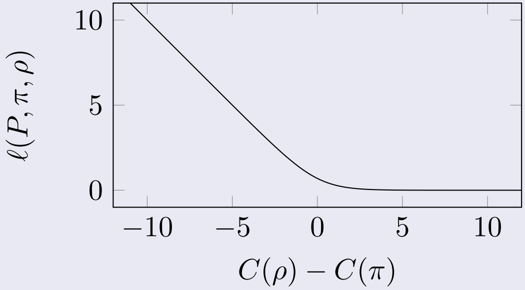
Training example $\pi \prec_P \rho$

Precedence π is better than precedence ρ for problem P.

Reminder: $C(\pi)$ is the predicted cost of precedence π.

Loss on training example $\pi \prec_P \rho$

$$\ell(P, \pi, \rho) = -\log \operatorname{sigmoid}(C(\rho) - C(\pi))$$



Loss example

- $\pi = [1, 2, 3]$: $C(\pi) = \frac{1}{6}(c_1 + 2c_2 + 3c_3)$
- $\rho = [3, 2, 1]$: $C(\rho) = \frac{1}{6}(c_3 + 2c_2 + 3c_1)$

$$
\begin{aligned}
\ell(P, \pi, \rho) &= -\log \operatorname{sigmoid}(C(\rho) - C(\pi)) \\
&= -\log \operatorname{sigmoid} \frac{1}{6}(2c_1 - 2c_3) \\
&= -\log \operatorname{sigmoid} \frac{1}{3}(c_1 - c_3)
\end{aligned}
$$

Evaluation

Symbol cost model	Success count[1]		Improvement	
	Mean	Std	Absolute	Relative
GCN (pred. and func.)	3951.6	1.62	+182.0	1.048
GCN (predicate only)	3923.6	2.24	+154.0	1.041
GCN (function only)	3874.2	1.83	+104.6	1.028
Frequency (baseline)	3769.6	3.07	0.0	1.000

[1] Total number of validation problems: 7648. Number of repetitions: 5.

Summary

- First-order logic (FOL) problem $\rightarrow$ clause normal form (CNF) $\rightarrow$ directed heterogeneous graph
- Graph convolutional network (GCN) predicts symbol costs
- Precedence recommendation: Sort symbols by costs
- Training:
  - Training example: (P, π, ρ) such that $\pi \prec_P \rho$
  - Proxy task: Precedence pair classification

Thank you for your attention!

filip.bartek@cvut.cz

Summary

- First-order logic (FOL) problem $\rightarrow$ clause normal form (CNF) $\rightarrow$ directed heterogeneous graph
- Graph convolutional network (GCN) predicts symbol costs
- Precedence recommendation: Sort symbols by costs
- Training:
  - Training example: (P, π, ρ) such that $\pi \prec_P \rho$
  - Proxy task: Precedence pair classification

Thank you for your attention!

`filip.bartek@cvut.cz`

Notation overview

$\pi(i)$	index of the i-th symbol in precedence π
c	vector of symbol costs (output of the GCN)
c_i	cost of the i-th symbol
$C(\pi)$	cost of symbol precedence π
$\ell(P, \pi, \rho)$	loss for training example $\pi \prec_P \rho$

Prediction of symbol precedence π

Require: Problem P

Ensure: Symbol precedence π

$c \leftarrow \mathrm{GCN}(P)$ {Forward pass through the GCN to obtain vector of symbol costs c}

$\pi \leftarrow \mathrm{argsort}^{-}(c_1, \ldots, c_n)$ {Sort symbols by c in non-increasing order to obtain precedence π}

return π

Precedence cost normalization

Assumption: Uniform distribution on precedences π

$$\mathbb{E}_\pi[C(\pi)] = \mathbb{E}_\pi\left[Z_n \sum_{i=1}^{n} i \cdot c(\pi(i))\right] = Z_n \sum_{i=1}^{n} i \cdot \mathbb{E}_\pi[c(\pi(i))]$$

$$= Z_n \left(\sum_{i=1}^{n} i\right) \mathbb{E}_i[c(i)] = \frac{2}{n(n+1)} \frac{n(n+1)}{2} \mathbb{E}_i[c(i)] = \mathbb{E}_i[c(i)]$$

Proof sketch: Precedence cost minimization

$$C(\pi) = \frac{2}{n(n+1)} \sum_{i=1}^{n} i \cdot c_{\pi(i)}$$

Lemma (Precedence cost minimization)

$$\operatorname*{argmin}_{\rho \in \mathrm{Perm}(n)} C(\rho) = \mathrm{argsort}^{-}(c_1, \ldots, c_n)$$

Example

$$C\left(\quad\right) < C\left(\quad\right)$$

Proof:

$$C\left(\quad\right) - C\left(\quad\right) \propto ((2 \cdot 3 + 3 \cdot 4) - (2 \cdot 4 + 3 \cdot 3))$$
$$= 18 - 17 = 1 > 0$$

Concise representation of training examples

Cost of symbol precedence π over signature of length n

$$C(\pi) = \frac{2}{n(n+1)} \sum_{i=1}^{n} i \cdot c_{\pi(i)} = \frac{2}{n(n+1)} \sum_{i=1}^{n} c_i \cdot \pi^{-1}(i)$$

Loss of training example $\pi \prec_P \rho$

$$\begin{aligned}\ell(P, \pi, \rho) &= -\log \operatorname{sigmoid}(C(\rho) - C(\pi)) \\ &= -\log \operatorname{sigmoid} \frac{2}{n(n+1)} \sum_{i=1}^{n} i(c_{\rho(i)} - c_{\pi(i)}) \\ &= -\log \operatorname{sigmoid} \frac{2}{n(n+1)} \sum_{i=1}^{n} c_i(\rho^{-1}(i) - \pi^{-1}(i))\end{aligned}$$

Concise representation of $\pi \prec_P \rho$: $\rho^{-1} - \pi^{-1}$

Precedence determines orientability of identities

$$f(x, a, y) \approx f(y, b, x)$$

- If $a > f$ and $a > b$, then $f(x, a, y) >_{lpo} f(y, b, x)$.
- If $f > a$ and $f > b$, then $f(x, a, y)$ and $f(y, b, x)$ are $>_{lpo}$-incomparable.

Precedence determines termination of completion

Set of identities:

$$x + 0 \approx x$$
$$x + s(y) \approx s(x + y)$$

- LPO($+ > s$): Completion orients from left to right and terminates.
- LPO($s > +$): Completion diverges:

$$x + 0 \to x$$
$$x + s(0) \to s(x)$$
$$x + s(s(0)) \to s(s(x))$$
$$\vdots$$
$$x + s^n(0) \to s^n(x)$$
$$\vdots$$

Precedence may inflate the search space

1. $>_{+s} = \text{LPO}(* > + > s)$
2. $>_{s+} = \text{LPO}(s > + > *)$

$$x + 0 \to x$$
$$x + s(y) \approx s(x + y) \qquad (\to_{+s}, \leftarrow_{s+})$$
$$x * 0 \to 0$$
$$x * s(y) \to x + (x * y)$$

1. With $>_{+s}$, the initial set is complete.
2. With $>_{s+}$, completion yields a large number of new rules:

$$x + s^n(0) \to s^n(x) \qquad (\forall n \in \mathbb{N})$$
$$x + (x * (x' + y')) \approx x * (x' + s(y')) \qquad \text{(unorientable)}$$

Superposition in Vampire

Selection selects at least one literal in each non-empty clause.
For example, it may select all maximal literals.

$$\frac{\underline{l = r} \vee C \quad \underline{L[s]} \vee D}{\sigma(L[r] \vee C \vee D)}$$

where:

- $\sigma = mgu(l, s)$
- s is not a variable
- $\sigma(r) \not\succeq \sigma(l)$
- L is not an equality literal

Positive superposition

$$\frac{C \vee s = t \quad D \vee u[s'] = v}{\sigma(C \vee D \vee u[t] = v)}$$

where:

- $\sigma = mgu(s, s')$
- $\sigma(t) \not\succeq \sigma(s)$
- $\sigma(v) \not\succeq \sigma(u)$
- $\sigma(s = t)$ is strictly maximal with respect to $\sigma(C)$, and C contains no selected literal
- $\sigma(u = v)$ is strictly maximal with respect to $\sigma(D)$, and D contains no selected literal
- s' is not a variable
- $\sigma(s = t) \not\succeq \sigma(u = v)$

Negative superposition

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

where:

- $\sigma = mgu(s, s')$
- $\sigma(t) \not\succeq \sigma(s)$
- $\sigma(v) \not\succeq \sigma(u)$
- $\sigma(s = t)$ is strictly maximal with respect to $\sigma(C)$, and C contains no selected literal
- $u \neq v$ is selected, or nothing is selected in $D \vee u \neq v$ and $\sigma(u \neq v)$ is maximal with respect to $\sigma(D)$
- s' is not a variable