Recommending Symbol Precedences by Machine Learning

Filip Bártek

Czech Technical University in Prague

March 23, 2022

Reasoning about addition

Axioms of addition of natural numbers:

$$x + s(y) = s(x + y) \qquad (A_s)$$
$$x + 0 = x \qquad (A_0)$$

Conjecture:

$$0 + s(s(0)) = s(s(0)) \qquad (C)$$

How can we prove that C follows from A_s and A_0?

Reasoning about addition

Negate the conjecture:

$$0 + s(s(0)) \neq s(s(0)) \qquad (NC)$$
$$x + s(y) = s(x + y) \qquad (A_s)$$
$$x + 0 = x \qquad (A_0)$$

Attempt to infer a contradiction.

Reasoning about addition

Transform the axioms into rewrite rules:

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

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

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

Reasoning about addition

Transform the axioms into rewrite rules:

$$0 + s(s(0)) \neq s(s(0)) \qquad (NC)$$
$$x + s(y) \to s(x + y) \qquad (R_{+s})$$
$$x + 0 \to x \qquad (R_0)$$

Rewrite NC using R_{+s} and R_0:

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

Reasoning about addition

What if we oriented A_s from right to left?

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

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

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

Reasoning about addition

What if we oriented A_s from right to left?

$$0 + s(s(0)) \neq s(s(0)) \qquad (NC)$$
$$x + s(y) \leftarrow s(x + y) \qquad (R_{s+})$$
$$x + 0 \rightarrow x \qquad (R_0)$$

Completion procedure yields an infinite sequence of new rules:

$$x + s(0) \rightarrow s(x) \qquad (R_1)$$
$$x + s(s(0)) \rightarrow s(s(x)) \qquad (R_2)$$
$$\vdots$$
$$x + s^n(0) \rightarrow s^n(x) \qquad (R_n)$$
$$\vdots$$

Symbol precedence matters

- R_{+s} solves the problem deterministically.
- R_{s+} may lead to an infinite sequence of inferences.

Symbol precedence matters

- R_{+s} solves the problem deterministically.
- R_{s+} may lead to an infinite sequence of inferences.

In an automated theorem prover (ATP), the orientation of equations depends on the *simplification term ordering* such as the Knuth-Bendix ordering (KBO), which is defined by a *symbol precedence*.

- If $+ > s$, then $x + s(y) >_{kbo} s(x + y)$ (R_{+s}).
- If $s > +$, then $s(x + y) >_{kbo} x + s(y)$ (R_{s+}).

First-order logic (FOL) with equality

Examples of clauses:

- $x + 0 = x$
- $x + s(y) = s(x + y)$
- $0 + s(s(0)) \neq s(s(0))$
- $\neg Nat(x) \vee x + s(y) = s(x + y)$
- $\neg R(x, y) \vee \neg R(y, z) \vee R(x, z)$
- $\neg R(x, y) \vee \neg R(y, x) \vee x = y$
- $\neg R(x, y) \vee x = y \vee S(x, y)$
- $\neg P \vee \neg Q \vee R \vee S$

Precedence recommendation

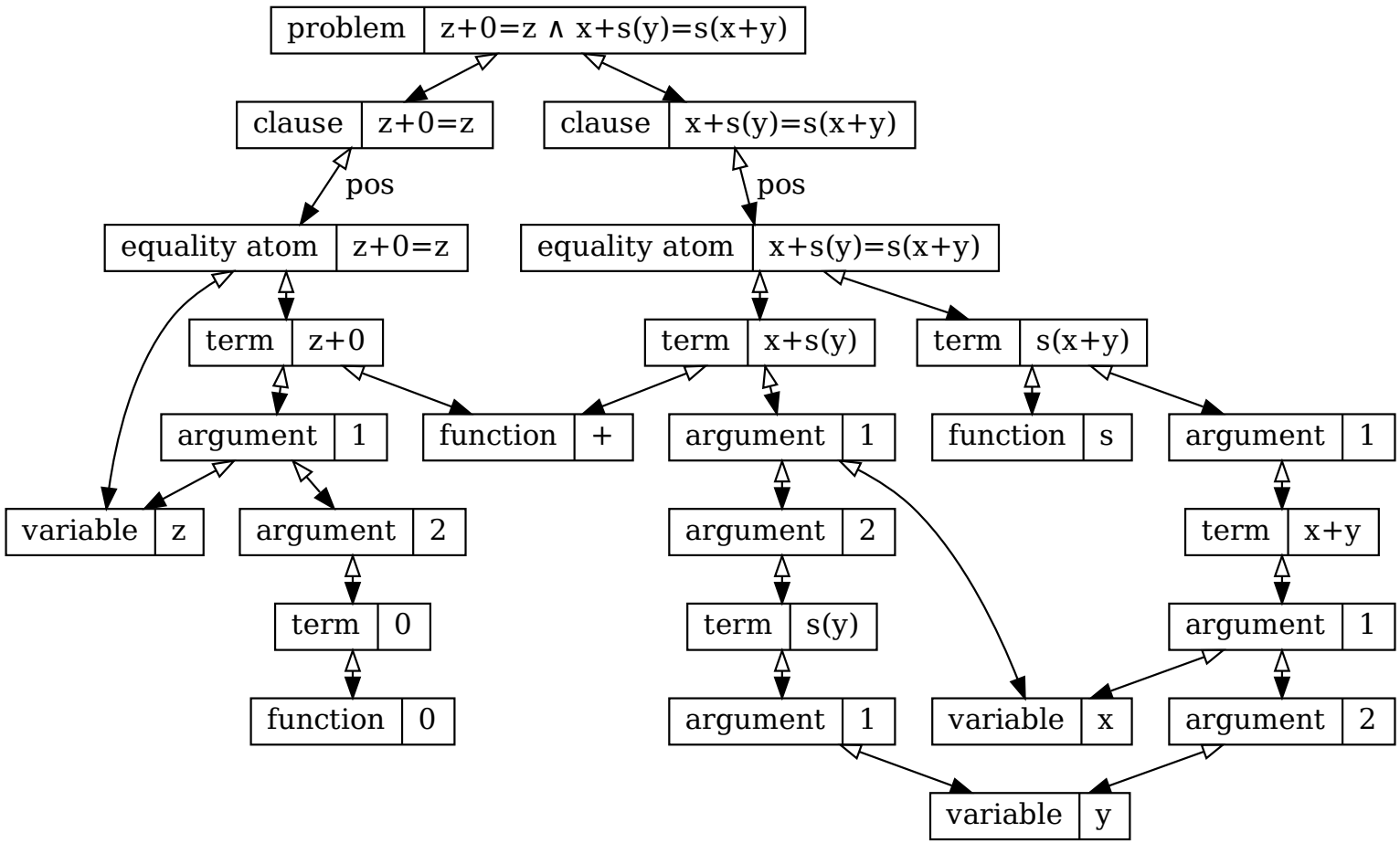
- Input: a FOL problem in clause normal form (CNF)
- Output: a symbol precedence
  - Try to minimize the runtime.
- Target algorithm: an automated theorem prover (ATP)
  - A fixed configuration
  - Ordered resolution
  - Superposition calculus
- Challenge: unaligned signatures across problems
- State of the art: simple heuristics (for example `invfreq`)

Our precedence recommender

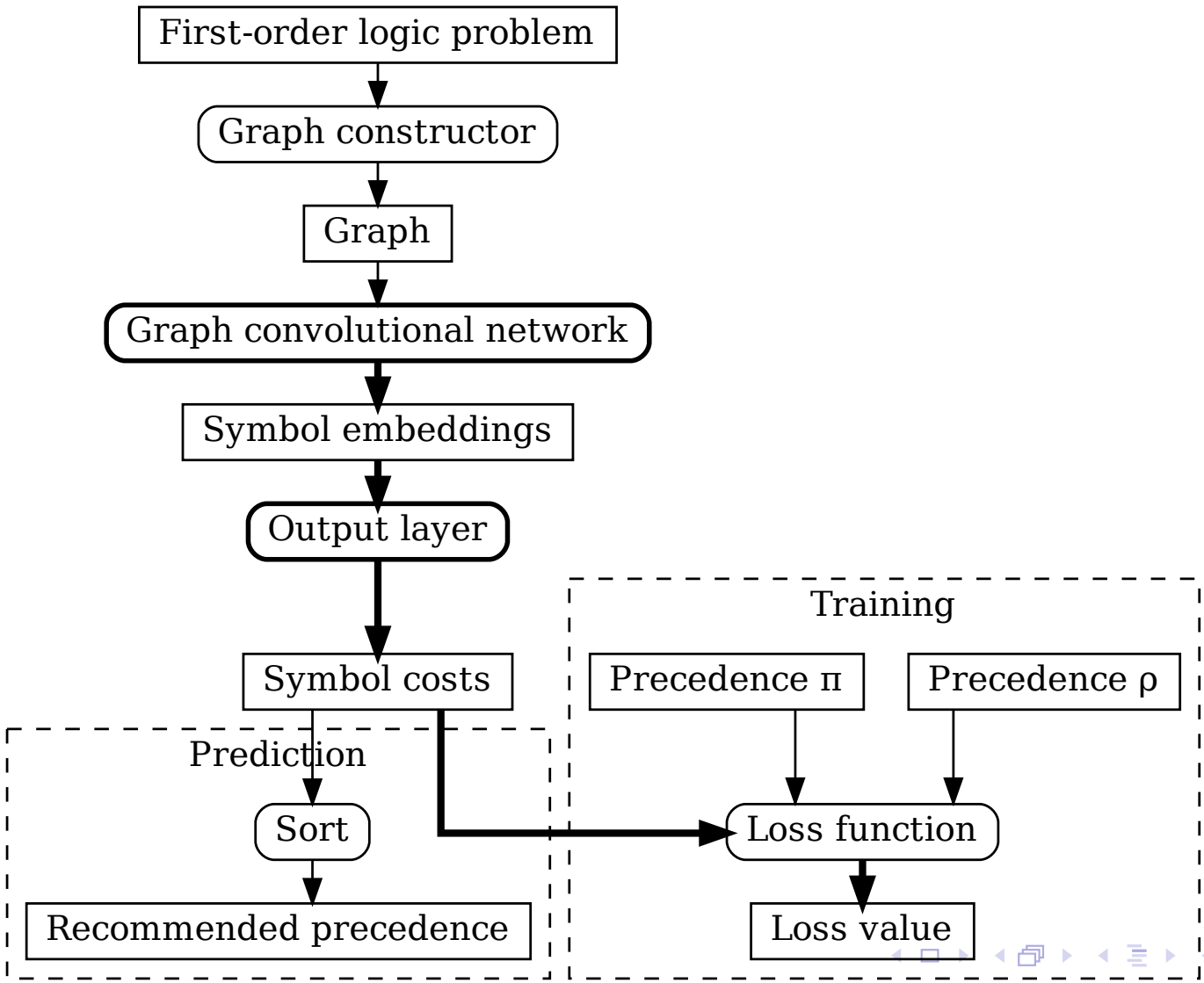
- Machine learning from ATP runs with random precedences
- Generalizes across problems
  - Symbol semantics
  - Signature length
- Approach:
  - ML core: Graph convolutional network (GCN)
  - Nested proxy tasks:
    - Precedence pair classification
    - Symbol cost regression

Graph representation of a CNF problem

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



Architecture



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
Simple (predicate only)	3827.2	1.94	+57.6	1.015
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 corresponds to a graph
- Graph convolutional network (GCN) predicts symbol costs
- Sorting symbols by costs yields a precedence
- Training data: precedence pairs "$\pi \prec_P \rho$"
  ("Precedence π is better than precedence ρ in problem P.")
- Final recommender outperforms the "frequency" heuristic by 4.8 % on TPTP

Thank you for your attention!

Filip Bártek
filip.bartek@cvut.cz

Question 1

Question

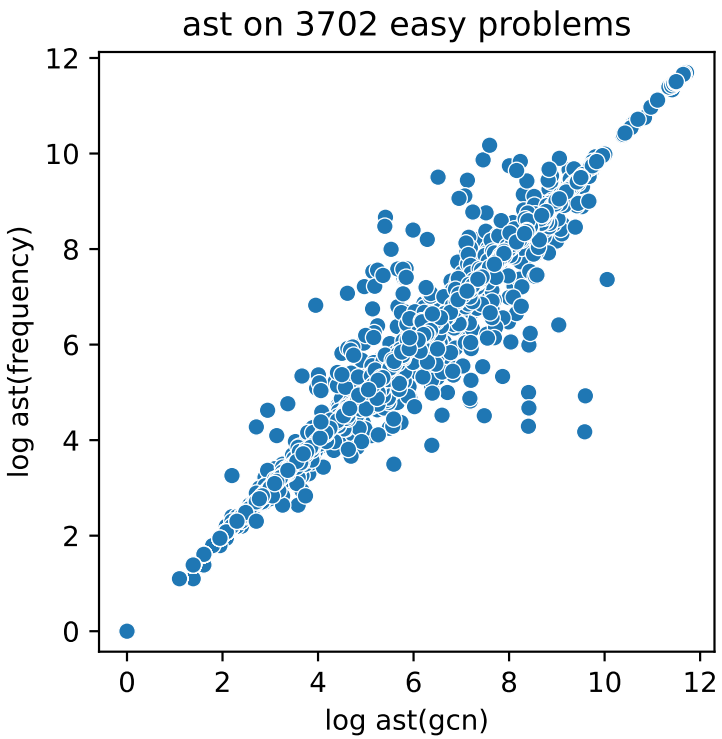
What is the relation between the results presented in the Table 1 of the paper on the Neural Precedence Recommender and the size of the set of created search space nodes during the created proofs?

Symbol cost model	Successes on $\mathcal{P}_{\text{val}}$		Improvement over baseline	
	Mean	Std	Absolute	Relative
GCN (predicate and function)	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
Simple (predicate only)	3827.2	1.94	+57.6	1.015
Frequency (baseline)	3769.6	3.07	0.0	1.000

Question 1

Question

What is the relation between the results presented in the Table 1 of the paper on the Neural Precedence Recommender and the size of the set of created search space nodes during the created proofs?



- Total: 3702
- $\mathrm{ast}(\mathrm{gcn}) = \mathrm{ast}(\mathrm{freq})$: 1899
- $\mathrm{ast}(\mathrm{gcn}) < \mathrm{ast}(\mathrm{freq})$: 981
- $\mathrm{ast}(\mathrm{gcn}) > \mathrm{ast}(\mathrm{freq})$: 822

Question 2

Question

What methods are considered to be applied for integration of the proving strategy and schedule optimization?

Answer

Combine Sequential Model-Based Optimization (SMBO) with fast schedule optimization. Iteratively construct a portfolio:

1. Train a probabilistic empirical performance model (EPM).
2. Estimate value of new strategies using the EPM, empirical runtimes of the portfolio, and fast schedule optimization.
3. Empirically evaluate the best strategy.
4. Expand the portfolio if the new strategy improves it.

Question 0

Question

Will these datasets be available for re-use by public?

Comment

The source code has been published.
I hope to publish the datasets when I consolidate the results for my dissertation thesis.

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

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:

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

Precedence cost

Reminder: c_i is the 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:

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

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

$$C\left(\text{  }\right) < C\left(\text{  }\right)$$

Proof:

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

Loss function

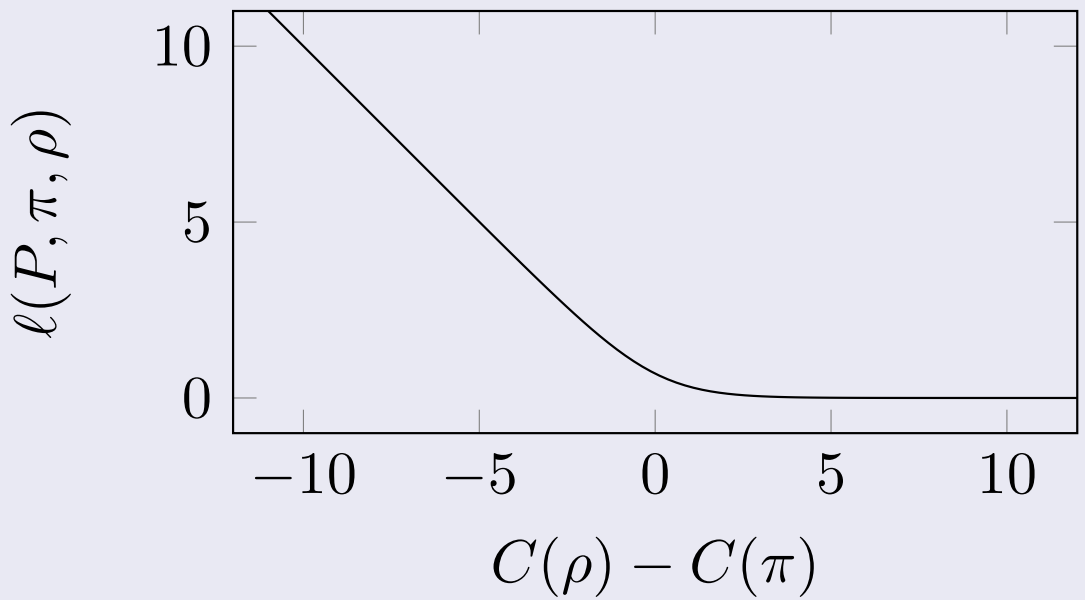
Training example $\pi \prec_P \rho$

Precedence π is better than precedence ρ for problem P.

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

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

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



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

$$
\begin{aligned}
x + 0 &\to x \\
x + s(y) &\approx s(x + y) && (\to_{+s},\ \leftarrow_{s+}) \\
x * 0 &\to 0 \\
x * s(y) &\to x + (x * y)
\end{aligned}
$$

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

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

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
- $\sigma(r) \not\geq \sigma(l)$
- 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:

- $\sigma = mgu(s, s')$
- $\sigma(t) \not\geq \sigma(s)$
- $\sigma(v) \not\geq \sigma(u)$
- $\sigma(s = t)$ 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
- $\sigma(s = t) \not\geq \sigma(u = v)$

Negative superposition

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

where:

- $\sigma = mgu(s, s')$
- $\sigma(t) \not\geq \sigma(s)$
- $\sigma(v) \not\geq \sigma(u)$
- $\sigma(s = t)$ 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