Automatic Theorem Proving with Vampire II

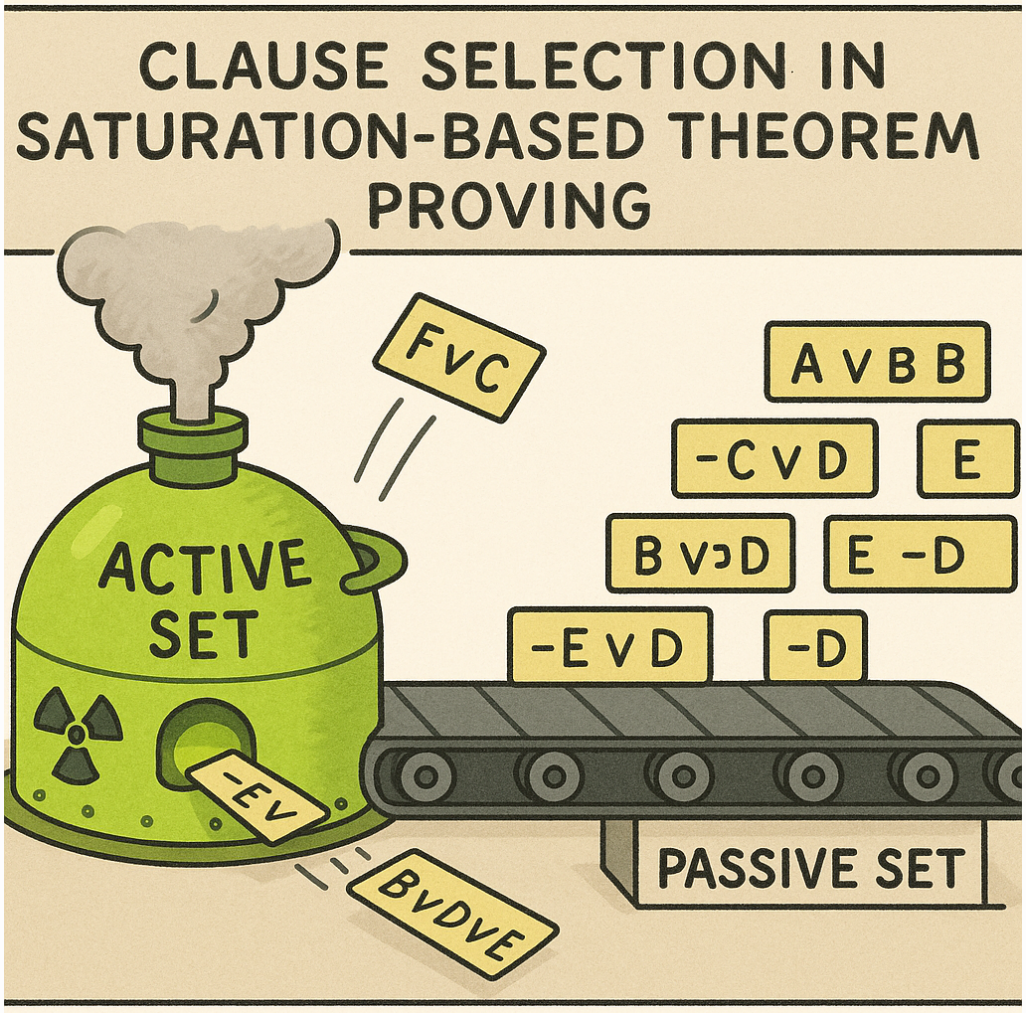
Martin Suda

Czech Technical University in Prague, Czech Republic

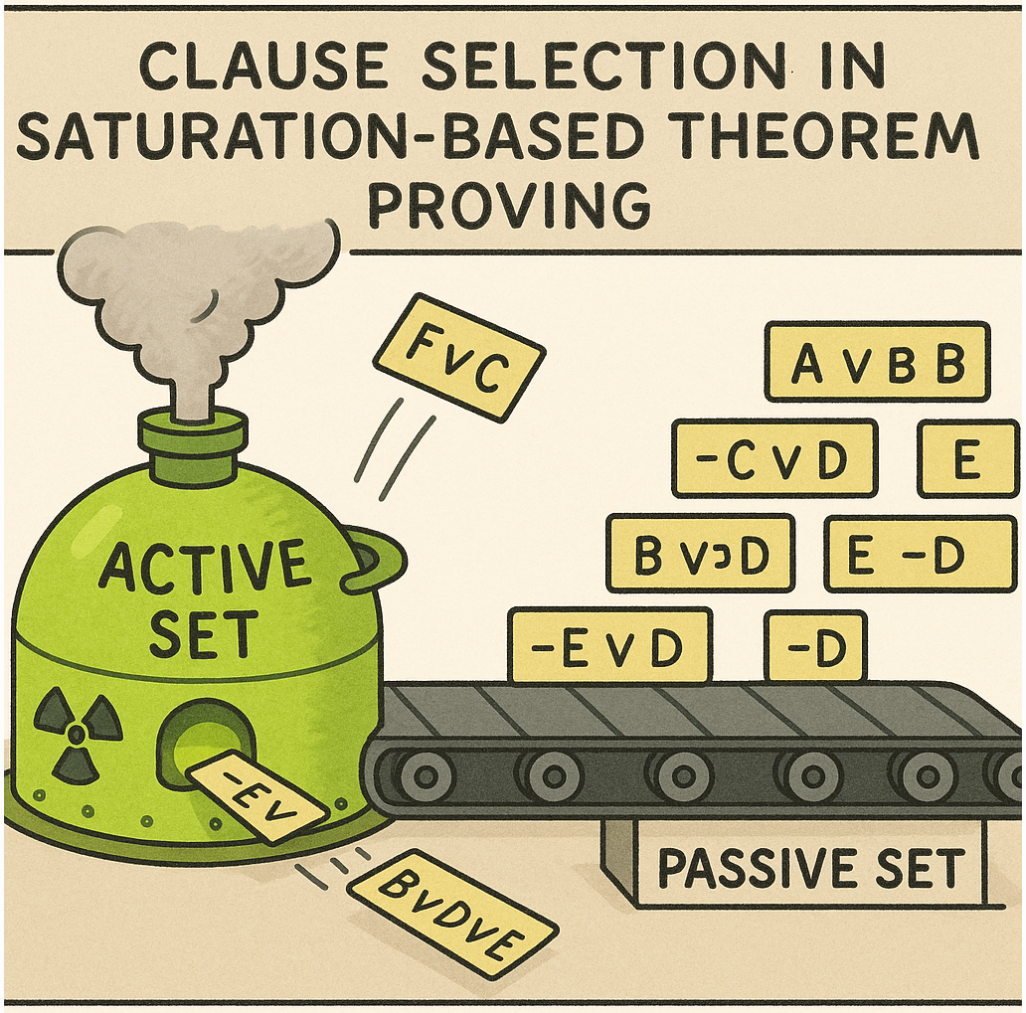
VTSA 2025, Liege, Belgium

Outline

Saturation-based Theorem Proving

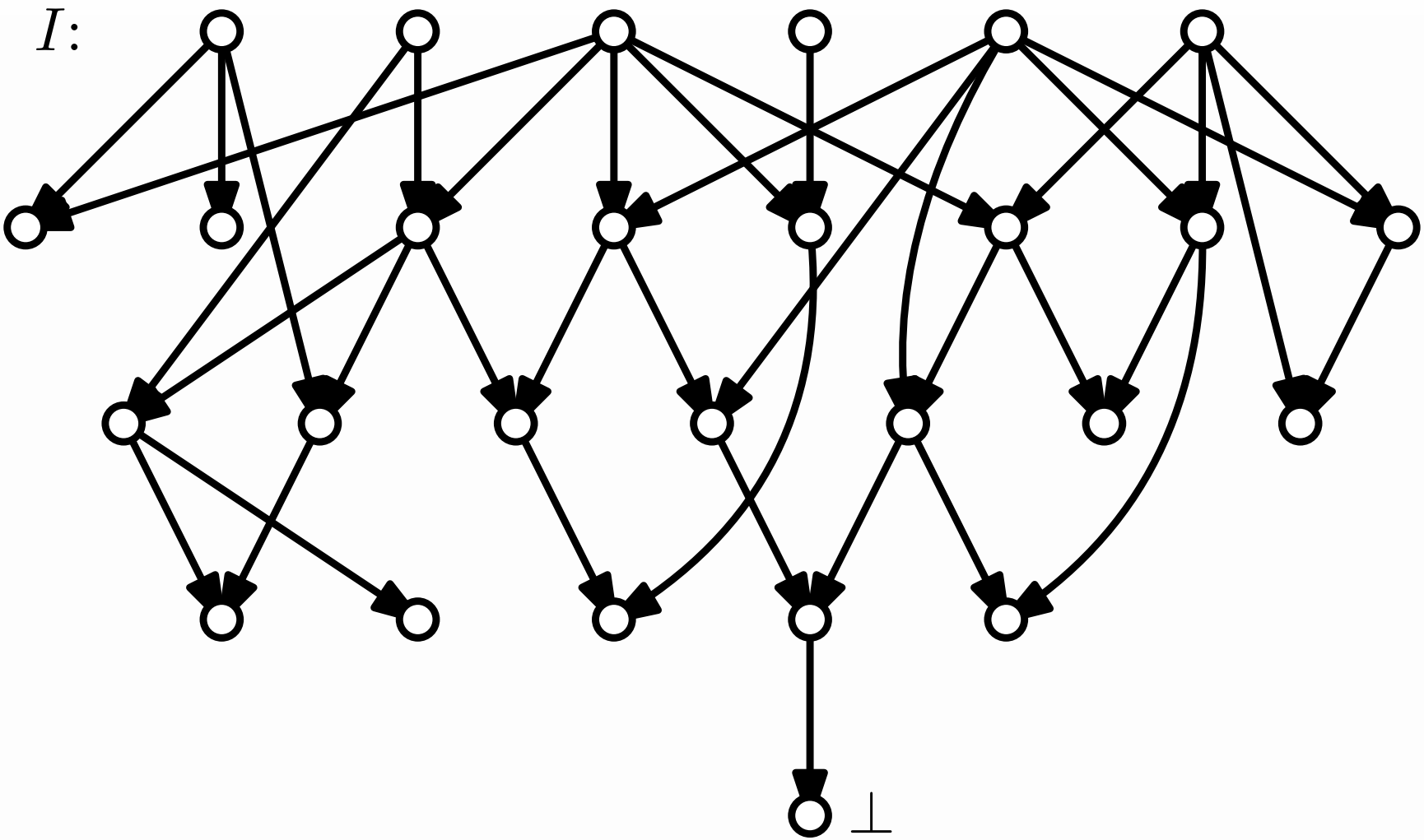


Saturation-based Theorem Proving

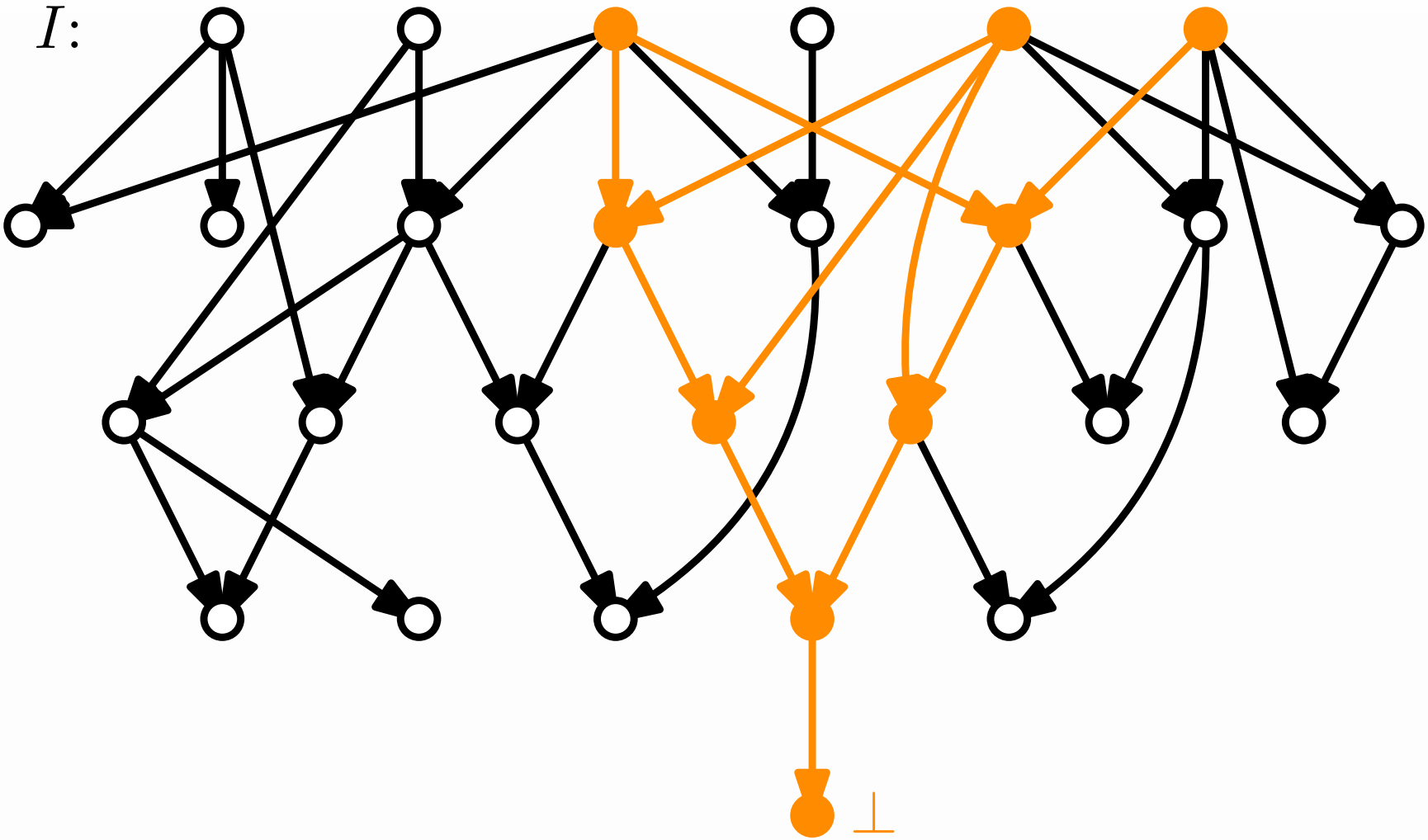


At a typical successful end: $|Passive| \gg |Active| \gg |Proof|$

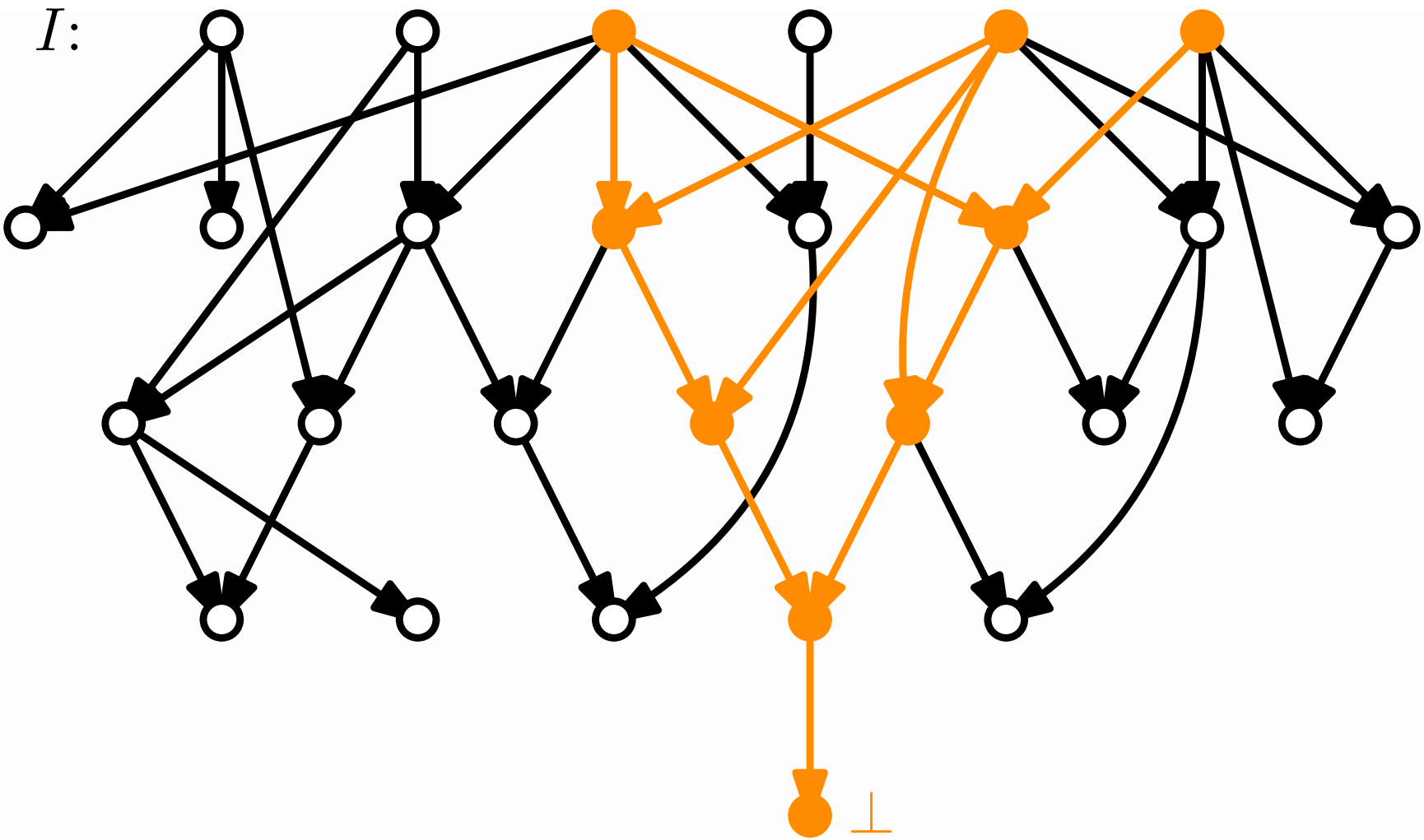
The Proof Is Often Just A Tiny Part



The Proof Is Often Just A Tiny Part



The Proof Is Often Just A Tiny Part



How close can we actually hope get to the perfect clause selection?

How is Clause Selection Traditionally Done?

Take simple clause evaluation criteria:

- *age*: prefer clauses that were generated long time ago
- *weight*: prefer clauses with fewer symbols

How is Clause Selection Traditionally Done?

Take simple clause evaluation criteria:

- *age*: prefer clauses that were generated long time ago
- *weight*: prefer clauses with fewer symbols

Combine them into a single scheme:

- have a *priority queue* ordering $Passive$ for each criterion
- *alternate* between selecting from the queues using a fixed *ratio*

How is Clause Selection Traditionally Done?

Take simple clause evaluation criteria:

- *age*: prefer clauses that were generated long time ago
- *weight*: prefer clauses with fewer symbols

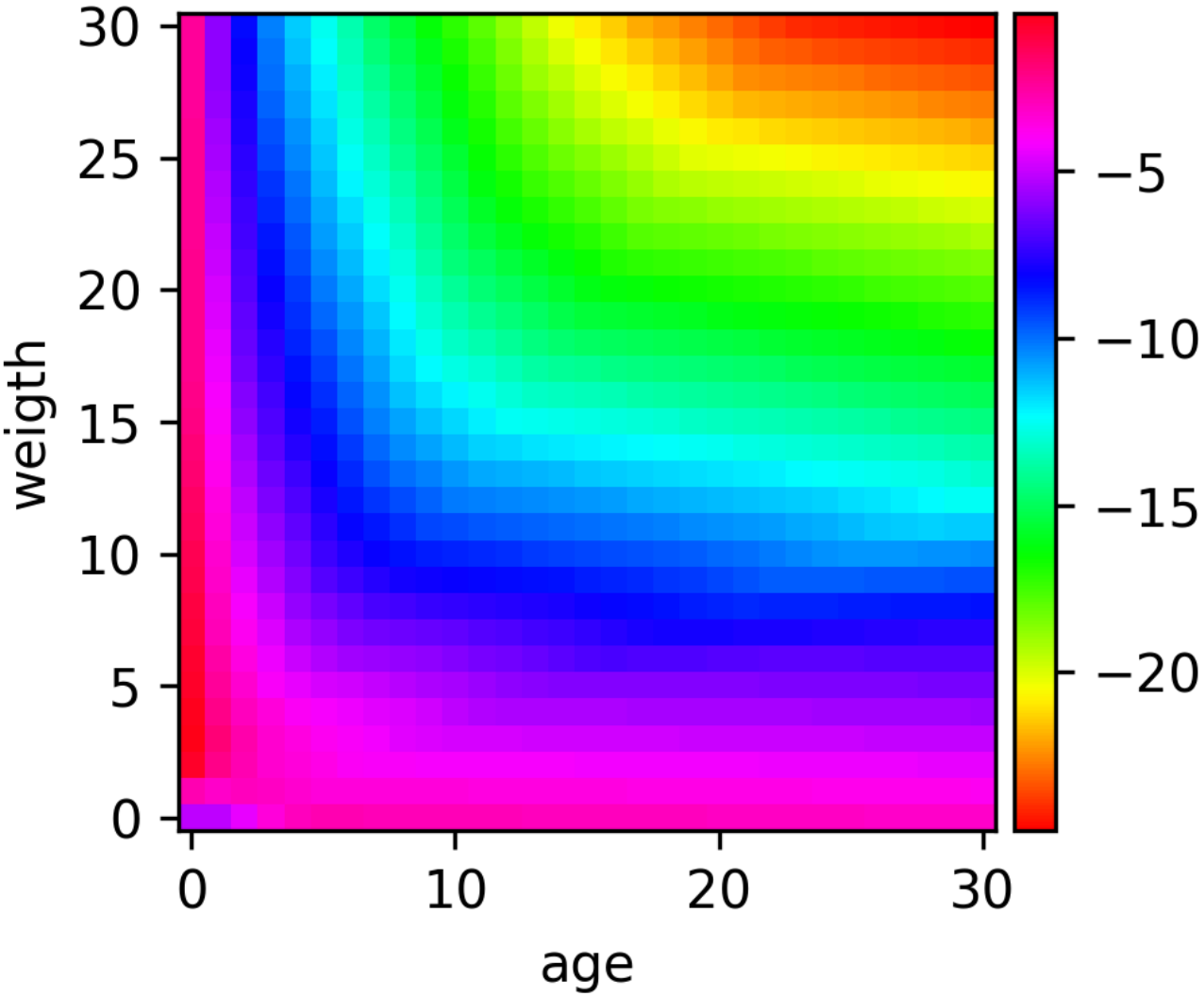
Combine them into a single scheme:

- have a *priority queue* ordering $Passive$ for each criterion
- *alternate* between selecting from the queues using a fixed *ratio*

Example (Organizing $Passive$ via two priority queues)

$\mathcal{S}$:

Sneak Peek: What Do NNs Think of Age and Weigth?



Key Reinforcement Learning Concepts

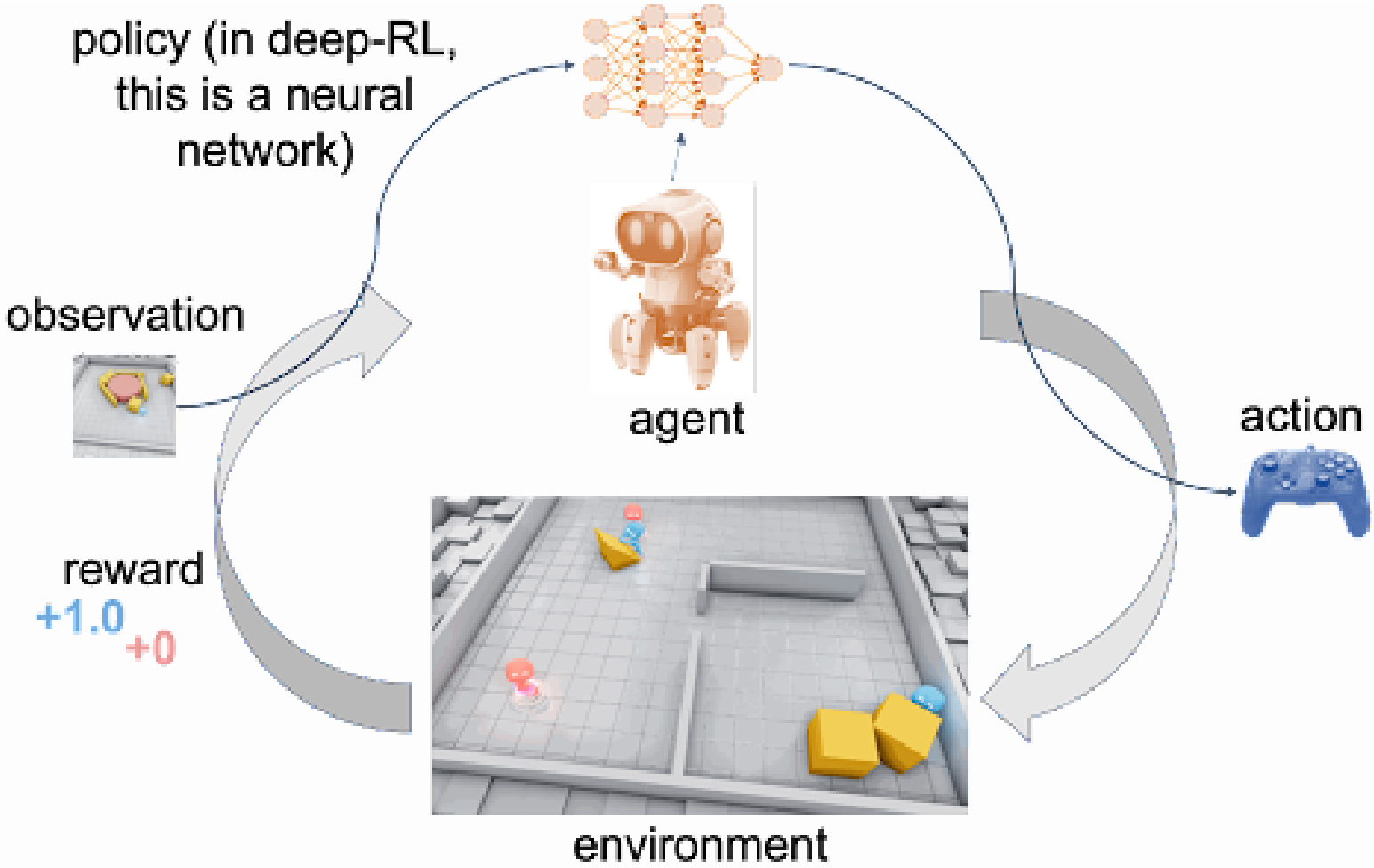


*Illustration from anyscale.com.

Why Reinforcement Learning?

Why Reinforcement Learning?

Inspired by the great successes:

- ATARI games (DQN)
  V. Mnih et al. Playing ATARI with deep reinforcement learning. CoRR, 2013.
- Board games (AlphaZero)
  D. Silver et al. A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play. Science, 2018.
- ...
- "I wan't to try it on my pet problem too!"

Why Reinforcement Learning?

Inspired by the great successes:

- ATARI games (DQN)
  V. Mnih et al. Playing ATARI with deep reinforcement learning. CoRR, 2013.
- Board games (AlphaZero)
  D. Silver et al. A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play. Science, 2018.
- ...
- "I wan't to try it on my pet problem too!"

What's really unique about RL?

- It programs itself (sometimes even optimally, in the limit)
- It could discover fundamentally novel tricks and hacks!

Saturation as an Reinforcement-Learning Environment

Agent

- the clause selection heuristic

Saturation as an Reinforcement-Learning Environment

Agent

- the clause selection heuristic

Action

- the next clause to select *from the current passive set*

Saturation as an Reinforcement-Learning Environment

Agent

- the clause selection heuristic

Action

- the next clause to select *from the current passive set*

State / Observation

- *static* - the conjecture we are trying to prove
- *evolving* - the internal state of the prover at particular moment

Saturation as an Reinforcement-Learning Environment

Agent

- the clause selection heuristic

Action

- the next clause to select *from the current passive set*

State / Observation

- *static* - the conjecture we are trying to prove
- *evolving* - the internal state of the prover at particular moment

Reward

- Score 1 point for solving a problem (within the time limit) ???

Saturation as an Reinforcement-Learning Environment

Agent

- the clause selection heuristic

Action

- the next clause to select *from the current passive set*

State / Observation

- *static* - the conjecture we are trying to prove
- *evolving* - the internal state of the prover at particular moment

Reward

- Score 1 point for solving a problem (within the time limit) ???

➥ TRAIL [Crouse et al.'21], [McKeown'23], [Shminke'23], ...

Design Decisions

Guiding Principle

The new design accommodates the old heuristic as an attainable point in the space of possible solutions.

Design Decisions

Guiding Principle

The new design accommodates the old heuristic as an attainable point in the space of possible solutions.

State / Observation

- the evolving state of an ATP is a large amorphous blob
- there is no state in the SoTA clause-selection heuristics
- let's discard state too $\Rightarrow$ assumption of *state-less* environment

Design Decisions

Guiding Principle

The new design accommodates the old heuristic as an attainable point in the space of possible solutions.

State / Observation

- the evolving state of an ATP is a large amorphous blob
- there is no state in the SoTA clause-selection heuristics
- let's discard state too $\Rightarrow$ assumption of *state-less* environment

Reward

- refusing the play the honest, super-sparse reward game

Design Decisions

Guiding Principle

The new design accommodates the old heuristic as an attainable point in the space of possible solutions.

State / Observation

- the evolving state of an ATP is a large amorphous blob
- there is no state in the SoTA clause-selection heuristics
- let's discard state too $\Rightarrow$ assumption of *state-less* environment

Reward

- refusing the play the honest, super-sparse reward game
- like in ENIGMA [Jakubův&Urban17]:
  Learn to recognize and prefer for selection clauses that look like those that contributed to a proof in past successful runs.

Towards the RL-Inspired Learning Operator

A *trace* of a successful proof attempt on problem P is a tuple

$$T = (P, \mathcal{C}, \mathcal{C}^+, \{\mathcal{P}_i\}_{i \in I_T}).$$

Towards the RL-Inspired Learning Operator

A *trace* of a successful proof attempt on problem P is a tuple

$$T = (P, \mathcal{C}, \mathcal{C}^+, \{\mathcal{P}_i\}_{i \in I_T}).$$

Learning operator (for clause selection)

- input: neural network $N_{\boldsymbol{\theta}}$ (learnable params $\boldsymbol{\theta}$), set of traces $\mathcal{T}$
- output: updated parameters $\boldsymbol{\theta}'$,
  such that $N_{\boldsymbol{\theta}'}$ is better at solving problems like those from $\mathcal{T}$

Towards the RL-Inspired Learning Operator

A *trace* of a successful proof attempt on problem P is a tuple

$$T = (P, \mathcal{C}, \mathcal{C}^+, \{\mathcal{P}_i\}_{i \in I_T}).$$

Learning operator (for clause selection)

- input: neural network $N_{\boldsymbol{\theta}}$ (learnable params $\boldsymbol{\theta}$), set of traces $\mathcal{T}$
- output: updated parameters $\boldsymbol{\theta}'$,
  such that $N_{\boldsymbol{\theta}'}$ is better at solving problems like those from $\mathcal{T}$

Logits and Policy

Assuming $N_{\boldsymbol{\theta}}$ produces a score $N_{\boldsymbol{\theta}}(C) = l_C$ for each clause C, then

Towards the RL-Inspired Learning Operator

A *trace* of a successful proof attempt on problem P is a tuple

$$T = (P, \mathcal{C}, \mathcal{C}^+, \{\mathcal{P}_i\}_{i \in I_T}).$$

Learning operator (for clause selection)

- input: neural network $N_{\boldsymbol{\theta}}$ (learnable params $\boldsymbol{\theta}$), set of traces $\mathcal{T}$
- output: updated parameters $\boldsymbol{\theta}'$,
  such that $N_{\boldsymbol{\theta}'}$ is better at solving problems like those from $\mathcal{T}$

Logits and Policy

Assuming $N_{\boldsymbol{\theta}}$ produces a score $N_{\boldsymbol{\theta}}(C) = l_C$ for each clause C, then

$$\pi_{C,\boldsymbol{\theta}} = \mathsf{softmax}_C\big(\{l_D\}_{D \in \mathcal{P}}\big) = \frac{e^{l_C}}{\sum_{D \in \mathcal{P}} e^{l_D}}$$

is the (stochastic) clause selection policy defined by $N_{\boldsymbol{\theta}}$

The RL-Inspired Operator

Policy Gradient Theorem [Williams'92]

To improve a policy in terms of the expected return we update

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha r_C \nabla_{\boldsymbol{\theta}} \log \pi_{C,\boldsymbol{\theta}},$$

where r_C is the return / reward at the corresponding step.

The RL-Inspired Operator

Policy Gradient Theorem [Williams'92]

To improve a policy in terms of the expected return we update

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha r_C \nabla_{\boldsymbol{\theta}} \log \pi_{C,\boldsymbol{\theta}},$$

where r_C is the return / reward at the corresponding step.

Our Operator:

Each moment in time i is an independent opportunity to improve, with

$$\delta_i^T = \text{mean}_{C \in \mathcal{P}_i^+} \nabla_{\boldsymbol{\theta}} \log \pi_{C,\boldsymbol{\theta}},$$

for a trace $T = (P, \mathcal{C}, \mathcal{C}^+, \{\mathcal{P}_i\}_{i \in I_T})$ and $\mathcal{P}_i^+ = \mathcal{P}_i \cap \mathcal{C}^+$. Then

$$\delta^T = \text{mean}_{i \in I_T} \delta_i^T \text{ and } \delta = \text{mean}_{T \in \mathcal{T}} \delta^T.$$

Neural Clause Evaluation

Aim for a balance between expressivity and speed of inference!

Neural Clause Evaluation

Aim for a balance between expressivity and speed of inference!

One-off GNN Invocation:

- Graph Neural Networks
- name-invariant formula representations
- relatively expensive; the more context the better
- here: only apply to the input CNF (i.e., only one GNN call)

Neural Clause Evaluation

Aim for a balance between expressivity and speed of inference!

One-off GNN Invocation:

- Graph Neural Networks
- name-invariant formula representations
- relatively expensive; the more context the better
- here: only apply to the input CNF (i.e., only one GNN call)

Generalizing Age and Weight with RvNNs:

- Recursive Neural Networks
- g-age: grow along the clause derivation tree
- g-weight: grow along the clause syntax tree
- share substructures (dag) and cache results

Neural Clause Evaluation

Aim for a balance between expressivity and speed of inference!

One-off GNN Invocation:

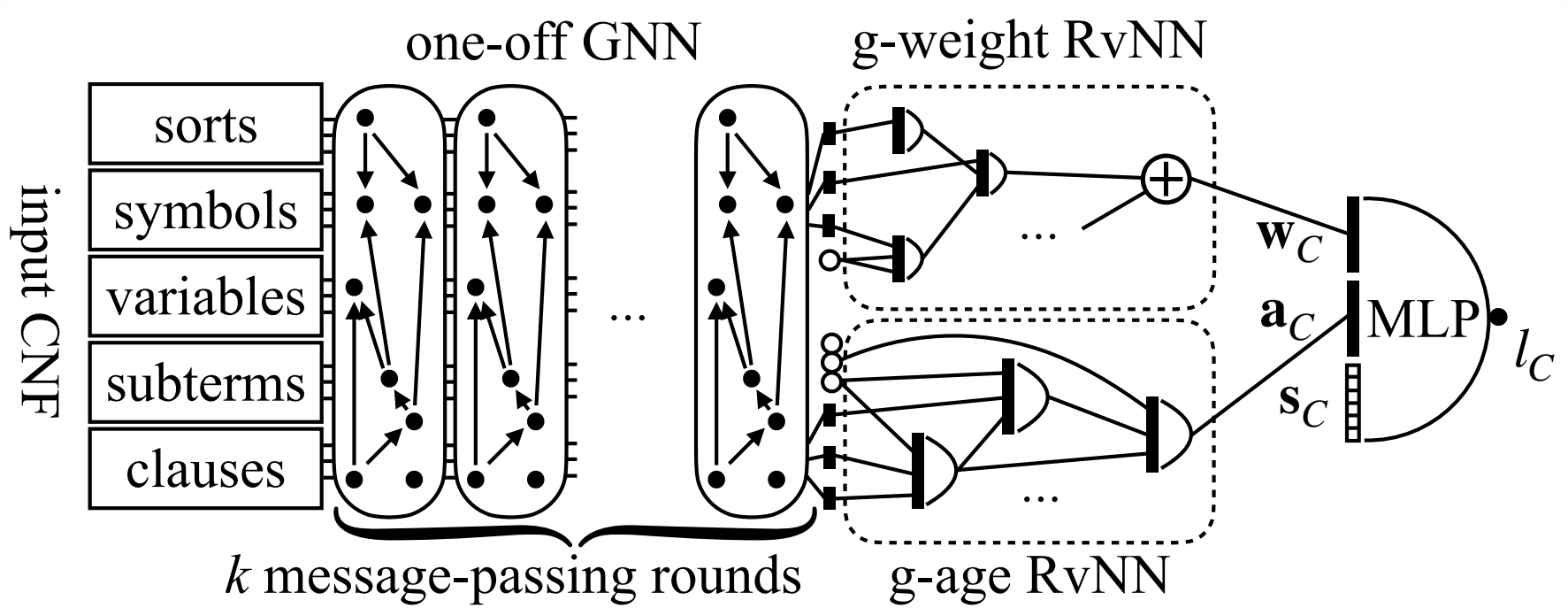
- Graph Neural Networks
- name-invariant formula representations
- relatively expensive; the more context the better
- here: only apply to the input CNF (i.e., only one GNN call)

Generalizing Age and Weight with RvNNs:

- Recursive Neural Networks
- g-age: grow along the clause derivation tree
- g-weight: grow along the clause syntax tree
- share substructures (dag) and cache results

Simple Hand-Crafted Features on Top!

Architecture Diagram



Implementation

Single Clause Queue:

- ordered by the computed logits $N_{\boldsymbol{\theta}}(C) = l_C$
- Could we also sample?

Implementation

Single Clause Queue:

- ordered by the computed logits $N_{\boldsymbol{\theta}}(C) = l_C$
- Could we also sample?

Delayed Insertion Buffer:

- insertions into passive are lazy
- only evaluate things in buffer when selection is called

Implementation

Single Clause Queue:

- ordered by the computed logits $N_{\boldsymbol{\theta}}(C) = l_C$
- Could we also sample?

Delayed Insertion Buffer:

- insertions into passive are lazy
- only evaluate things in buffer when selection is called

Iterative Improvement Loop:

- run (guided/plain) prover, collect traces, train from traces
- repeat

Experiments

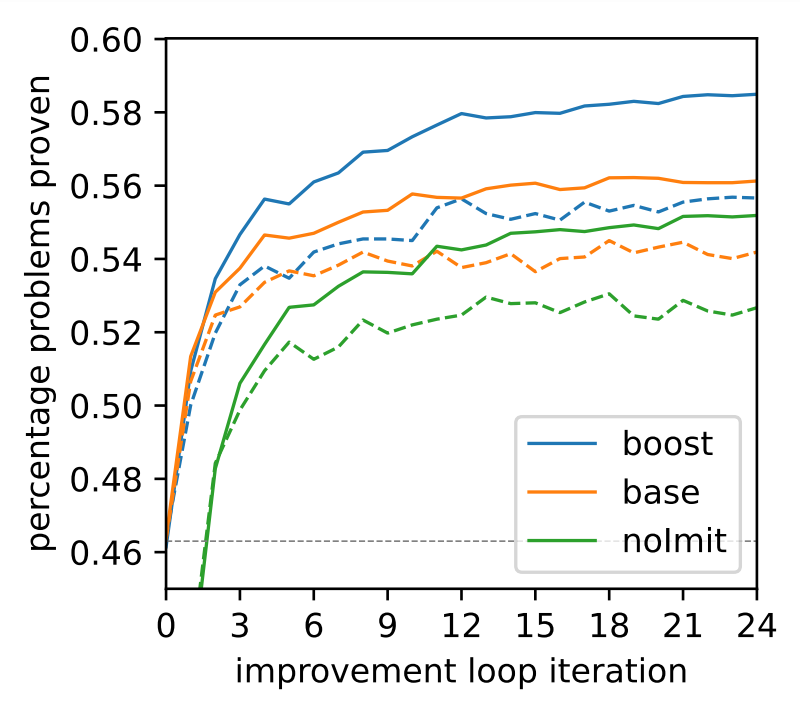
Setup:

- TPTP v9 CNF+FOF, $19\,477$ problems (train/test split)
- Vampire's default strategy (1:1 age-weight alternation)
- limit of $30\,000$ Mi ($\sim$10 s) per proof attempt

Experiments

Setup:

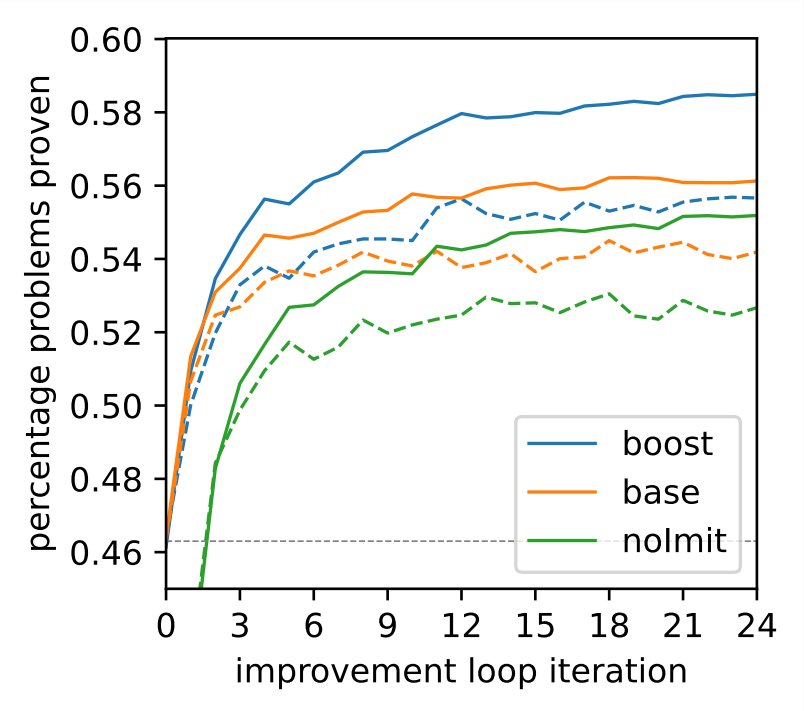
- TPTP v9 CNF+FOF, $19\,477$ problems (train/test split)
- Vampire's default strategy (1:1 age-weight alternation)
- limit of $30\,000$ Mi ($\sim$10 s) per proof attempt

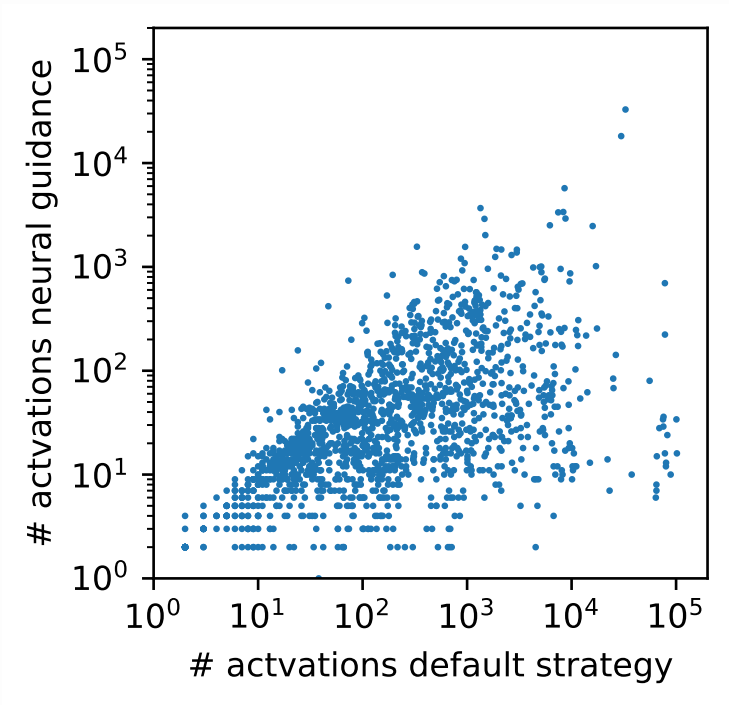


Experiments

Setup:

- TPTP v9 CNF+FOF, $19\,477$ problems (train/test split)
- Vampire's default strategy (1:1 age-weight alternation)
- limit of $30\,000$ Mi ($\sim$10 s) per proof attempt





Experiments II

Solving Hard Problems:

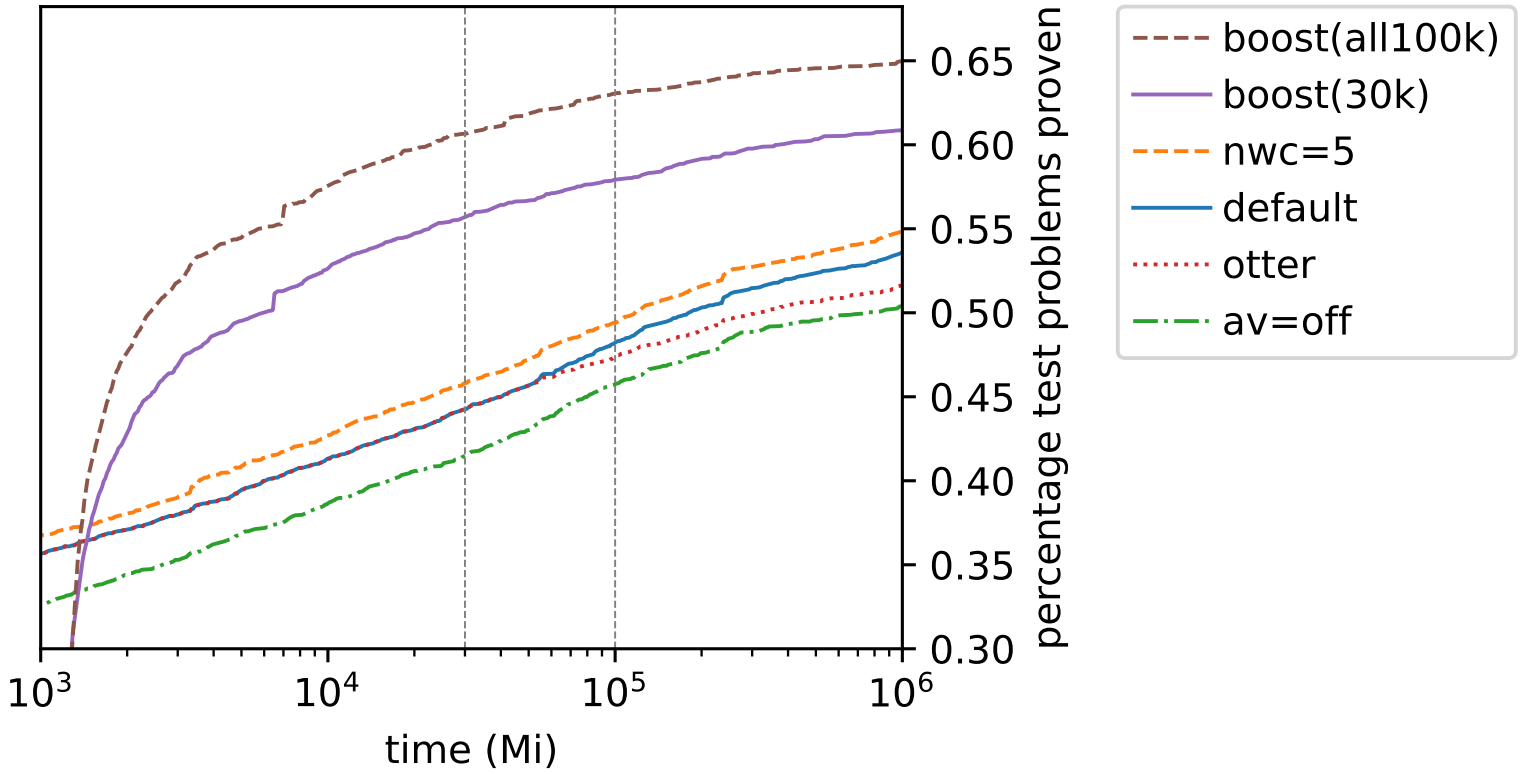
- overfit to TPTP with $100\,000$ Mi-limited runs
- ran for 12.4 days
- solved 130 rating 1.0 (49 never solved, 8 status UNK)

Experiments II

Solving Hard Problems:

- overfit to TPTP with $100\,000$ Mi-limited runs
- ran for 12.4 days
- solved 130 rating 1.0 (49 never solved, 8 status UNK)

Put Into Perspective:



Mini-Conclusion: Machine Learning for Solvers

Pick a Heuristic

- many don't-care non-deterministic choice points
- normally governed by hand-crafted heuristics

Mini-Conclusion: Machine Learning for Solvers

Pick a Heuristic

- many don't-care non-deterministic choice points
- normally governed by hand-crafted heuristics

Replace it with a neural network

- give it access to the relevant context for making decisions
- feature engineering / auto-learning of task-relevant node embeddings (GNNs, RvNNs,...)
- faithful representations / easy-to-compute abstractions

Mini-Conclusion: Machine Learning for Solvers

Pick a Heuristic

- many don't-care non-deterministic choice points
- normally governed by hand-crafted heuristics

Replace it with a neural network

- give it access to the relevant context for making decisions
- feature engineering / auto-learning of task-relevant node embeddings (GNNs, RvNNs,...)
- faithful representations / easy-to-compute abstractions

Train the network from previous runs

- What lead to a success? / What was a reason for failure?
- The proxy (ML) task vs the true target

Outline

A First-order Resolution Calculus

Example

$$\frac{\neg \mathrm{man}(X) \vee \mathrm{mortal}(X) \quad \mathrm{man}(\mathrm{socrates})}{}$$

A First-order Resolution Calculus

Example

$$\frac{\neg \mathrm{man}(X) \vee \mathrm{mortal}(X) \quad \mathrm{man}(\mathrm{socrates})}{\mathrm{mortal}(\mathrm{socrates})}$$

Intuition:

A First-order Resolution Calculus

Example

$$\frac{\neg \mathrm{man}(X) \vee \mathrm{mortal}(X) \quad \mathrm{man}(\mathrm{socrates})}{\mathrm{mortal}(\mathrm{socrates})}$$

Intuition: A first order variable (such as X) stands for all possible objects of the domain (recall the implicit $\forall$ in front of each clause).

A First-order Resolution Calculus

Example

$$\frac{\neg \mathrm{man}(X) \vee \mathrm{mortal}(X) \quad \mathrm{man}(\mathrm{socrates})}{\mathrm{mortal}(\mathrm{socrates})}$$

Intuition: A first order variable (such as X) stands for all possible objects of the domain (recall the implicit $\forall$ in front of each clause). First-order resolution inference first instantiates variables of the premise clauses to make the pivots complementary and then resolves as usual.

A First-order Resolution Calculus

Example

$$\frac{\neg \mathrm{man}(X) \vee \mathrm{mortal}(X) \quad \mathrm{man}(\mathrm{socrates})}{\mathrm{mortal}(\mathrm{socrates})}$$

Intuition: A first order variable (such as X) stands for all possible objects of the domain (recall the implicit $\forall$ in front of each clause). First-order resolution inference first instantiates variables of the premise clauses to make the pivots complementary and then resolves as usual.

In full generality, we have

$$\frac{L_1 \vee C_1 \quad \neg L_2 \vee C_2}{(C_1 \vee C_2)\sigma} \text{ (BR).}$$

where σ is the most general unifier of L_1 and L_2. (Fact analogously).

A First-order Resolution Calculus

Example

$$\frac{\neg \mathrm{man}(X) \vee \mathrm{mortal}(X) \quad \mathrm{man}(\mathrm{socrates})}{\mathrm{mortal}(\mathrm{socrates})}$$

Intuition: A first order variable (such as X) stands for all possible objects of the domain (recall the implicit $\forall$ in front of each clause). First-order resolution inference first instantiates variables of the premise clauses to make the pivots complementary and then resolves as usual.

In full generality, we have

$$\frac{L_1 \vee C_1 \quad \neg L_2 \vee C_2}{(C_1 \vee C_2)\sigma} \text{ (BR).}$$

where σ is the most general unifier of L_1 and L_2. (Fact analogously).

Here: $\sigma = \{X/\mathrm{socrates}\}$

A Bit Less Trivial Example(s)

Are the following two atoms unifiable?

$$p(Z, f(X, b, Z)) \qquad p(h(X), f(g(a), Y, Z))$$

A Bit Less Trivial Example(s)

Are the following two atoms unifiable?

$$p(Z, f(X, b, Z)) \qquad p(h(X), f(g(a), Y, Z))$$

And what about these?

$$p(g(X, a, Z), Z) \qquad p(g(k(Z), Y, Z), h(X))$$

Some Terminology First

Recall: Terms

- every variable $x \in \mathcal{V}$ is a term
- if $f/k \in \Sigma_F$ and $t_1, \ldots, t_k$ are terms, then $f(t_1, \ldots, t_k)$ is a term

Some Terminology First

Recall: Terms

- every variable $x \in \mathcal{V}$ is a term
- if $f/k \in \Sigma_F$ and $t_1, \ldots, t_k$ are terms, then $f(t_1, \ldots, t_k)$ is a term

Convention: Variables: $x, y, z, \ldots$, constants: $c, d, \ldots$, terms: $s, t, \ldots$

Some Terminology First

Recall: Terms

- every variable $x \in \mathcal{V}$ is a term
- if $f/k \in \Sigma_F$ and $t_1, \ldots, t_k$ are terms, then $f(t_1, \ldots, t_k)$ is a term

Convention: Variables: $x, y, z, \ldots$, constants: $c, d, \ldots$, terms: $s, t, \ldots$

Subterms and co.

- A subterm of t is a substring of t which is also a term.

Some Terminology First

Recall: Terms

- every variable $x \in \mathcal{V}$ is a term
- if $f/k \in \Sigma_F$ and $t_1, \ldots, t_k$ are terms, then $f(t_1, \ldots, t_k)$ is a term

Convention: Variables: $x, y, z, \ldots$, constants: $c, d, \ldots$, terms: $s, t, \ldots$

Subterms and co.

- A subterm of t is a substring of t which is also a term.
- If s is a subterm of t, we say that s occurs in t.

Some Terminology First

Recall: Terms

- every variable $x \in \mathcal{V}$ is a term
- if $f/k \in \Sigma_F$ and $t_1, \ldots, t_k$ are terms, then $f(t_1, \ldots, t_k)$ is a term

Convention: Variables: $x, y, z, \ldots$, constants: $c, d, \ldots$, terms: $s, t, \ldots$

Subterms and co.

- A subterm of t is a substring of t which is also a term.
- If s is a subterm of t, we say that s occurs in t.
- $Var(t)$ denotes the set of variables which occur in t.

Some Terminology First

Recall: Terms

- every variable $x \in \mathcal{V}$ is a term
- if $f/k \in \Sigma_F$ and $t_1, \ldots, t_k$ are terms, then $f(t_1, \ldots, t_k)$ is a term

Convention: Variables: $x, y, z, \ldots$, constants: $c, d, \ldots$, terms: $s, t, \ldots$

Subterms and co.

- A subterm of t is a substring of t which is also a term.
- If s is a subterm of t, we say that s occurs in t.
- $Var(t)$ denotes the set of variables which occur in t.
- A term t is called ground if $Var(t) = \emptyset$.

Substitutions

A substitution is a finite mapping from variables to terms.

Substitutions

A substitution is a finite mapping from variables to terms.

Notation and conventions

We write $\theta = \{x_1/t_1, \ldots, x_n/t_n\}$ where

- $x_1, \ldots, x_n$ are distinct variables,

Substitutions

A substitution is a finite mapping from variables to terms.

Notation and conventions

We write $\theta = \{x_1/t_1, \ldots, x_n/t_n\}$ where

- $x_1, \ldots, x_n$ are distinct variables,
- $t_1, \ldots, t_n$ are terms,

Substitutions

A substitution is a finite mapping from variables to terms.

Notation and conventions

We write $\theta = \{x_1/t_1, \ldots, x_n/t_n\}$ where

- $x_1, \ldots, x_n$ are distinct variables,
- $t_1, \ldots, t_n$ are terms,
- $x_i \neq t_i$ for every $i = 1, \ldots, n$.

Substitutions

A substitution is a finite mapping from variables to terms.

Notation and conventions

We write $\theta = \{x_1/t_1, \ldots, x_n/t_n\}$ where

- $x_1, \ldots, x_n$ are distinct variables,
- $t_1, \ldots, t_n$ are terms,
- $x_i \neq t_i$ for every $i = 1, \ldots, n$.

Substitutions

A substitution is a finite mapping from variables to terms.

Notation and conventions

We write $\theta = \{x_1/t_1, \ldots, x_n/t_n\}$ where

- $x_1, \ldots, x_n$ are distinct variables,
- $t_1, \ldots, t_n$ are terms,
- $x_i \neq t_i$ for every $i = 1, \ldots, n$.

Variables $x_1, \ldots, x_n$ are said to be bound to the terms $t_1, \ldots, t_n$, resp.
A pair x_i/t_i is called a binding.

Substitutions II

Consider a substitution $\theta = \{x_1/t_1, \ldots, x_n/t_n\}$

Substitutions II

Consider a substitution $\theta = \{x_1/t_1, \ldots, x_n/t_n\}$

- if $n = 0$: empty substitution; denoted ϵ

Substitutions II

Consider a substitution $\theta = \{x_1/t_1, \ldots, x_n/t_n\}$

- if $n = 0$: empty substitution; denoted ϵ
- θ is called ground if all t_i are ground terms

Substitutions II

Consider a substitution $\theta = \{x_1/t_1, \ldots, x_n/t_n\}$

- if $n = 0$: empty substitution; denoted ϵ
- θ is called ground if all t_i are ground terms
- θ is called a pure variable substitution, if all t_i are variables

Substitutions II

Consider a substitution $\theta = \{x_1/t_1, \ldots, x_n/t_n\}$

- if $n = 0$: empty substitution; denoted ϵ
- θ is called ground if all t_i are ground terms
- θ is called a pure variable substitution, if all t_i are variables
- θ is called a renaming, if $\{x_1, \ldots, x_n\} = \{t_1, \ldots, t_n\}$

Substitutions II

Consider a substitution $\theta = \{x_1/t_1, \ldots, x_n/t_n\}$

- if $n = 0$: empty substitution; denoted ϵ
- θ is called ground if all t_i are ground terms
- θ is called a pure variable substitution, if all t_i are variables
- θ is called a renaming, if $\{x_1, \ldots, x_n\} = \{t_1, \ldots, t_n\}$
- $Dom(\theta) = \{x_1, \ldots, x_n\}$, $Rng(\theta) = \{t_1, \ldots, t_n\}$

Substitutions II

Consider a substitution $\theta = \{x_1/t_1, \ldots, x_n/t_n\}$

- if $n = 0$: empty substitution; denoted ϵ
- θ is called ground if all t_i are ground terms
- θ is called a pure variable substitution, if all t_i are variables
- θ is called a renaming, if $\{x_1, \ldots, x_n\} = \{t_1, \ldots, t_n\}$
- $Dom(\theta) = \{x_1, \ldots, x_n\}$, $Rng(\theta) = \{t_1, \ldots, t_n\}$

Substitutions II

Consider a substitution $\theta = \{x_1/t_1, \ldots, x_n/t_n\}$

- if $n = 0$: empty substitution; denoted ϵ
- θ is called ground if all t_i are ground terms
- θ is called a pure variable substitution, if all t_i are variables
- θ is called a renaming, if $\{x_1, \ldots, x_n\} = \{t_1, \ldots, t_n\}$
- $Dom(\theta) = \{x_1, \ldots, x_n\}$, $Rng(\theta) = \{t_1, \ldots, t_n\}$

Example

For example $\{x/y, y/z, z/x\}$ is a renaming.

Substitutions II

Consider a substitution $\theta = \{x_1/t_1, \ldots, x_n/t_n\}$

- if $n = 0$: empty substitution; denoted ϵ
- θ is called ground if all t_i are ground terms
- θ is called a pure variable substitution, if all t_i are variables
- θ is called a renaming, if $\{x_1, \ldots, x_n\} = \{t_1, \ldots, t_n\}$
- $Dom(\theta) = \{x_1, \ldots, x_n\}$, $Rng(\theta) = \{t_1, \ldots, t_n\}$

Example

For example $\{x/y, y/z, z/x\}$ is a renaming. So is ϵ.

Applying a Substitution

The result of applying a substitution θ to a term s is a term $s\theta$ obtained by the simultaneous replacement of each occurrence of a variable from $Dom(\theta)$ in s by the corresponding term in $Rng(\theta)$.

Applying a Substitution

The result of applying a substitution θ to a term s is a term $s\theta$ obtained by the simultaneous replacement of each occurrence of a variable from $Dom(\theta)$ in s by the corresponding term in $Rng(\theta)$.

Formally

- For a variable $x \in Dom(\theta)$ we set $x\theta = \theta(x)$.

Applying a Substitution

The result of applying a substitution θ to a term s is a term $s\theta$ obtained by the simultaneous replacement of each occurrence of a variable from $Dom(\theta)$ in s by the corresponding term in $Rng(\theta)$.

Formally

- For a variable $x \in Dom(\theta)$ we set $x\theta = \theta(x)$.
- For a variable $x \notin Dom(\theta)$ we set $x\theta = x$.

Applying a Substitution

The result of applying a substitution θ to a term s is a term $s\theta$ obtained by the simultaneous replacement of each occurrence of a variable from $Dom(\theta)$ in s by the corresponding term in $Rng(\theta)$.

Formally

- For a variable $x \in Dom(\theta)$ we set $x\theta = \theta(x)$.
- For a variable $x \notin Dom(\theta)$ we set $x\theta = x$.
- Extend "homomorphically": $f(t_1, \ldots, t_n)\theta = f(t_1\theta, \ldots, t_n\theta)$.

Applying a Substitution

The result of applying a substitution θ to a term s is a term $s\theta$ obtained by the simultaneous replacement of each occurrence of a variable from $Dom(\theta)$ in s by the corresponding term in $Rng(\theta)$.

Formally

- For a variable $x \in Dom(\theta)$ we set $x\theta = \theta(x)$.
- For a variable $x \notin Dom(\theta)$ we set $x\theta = x$.
- Extend "homomorphically": $f(t_1, \ldots, t_n)\theta = f(t_1\theta, \ldots, t_n\theta)$.

Applying a Substitution

The result of applying a substitution θ to a term s is a term $s\theta$ obtained by the simultaneous replacement of each occurrence of a variable from $Dom(\theta)$ in s by the corresponding term in $Rng(\theta)$.

Formally

- For a variable $x \in Dom(\theta)$ we set $x\theta = \theta(x)$.
- For a variable $x \notin Dom(\theta)$ we set $x\theta = x$.
- Extend "homomorphically": $f(t_1, \ldots, t_n)\theta = f(t_1\theta, \ldots, t_n\theta)$.

Example (Why "simultaneous"?)

With $\theta = \{x/f(y), y/c\}$ we want $g(x, y)\theta$ to be

Applying a Substitution

The result of applying a substitution θ to a term s is a term $s\theta$ obtained by the simultaneous replacement of each occurrence of a variable from $Dom(\theta)$ in s by the corresponding term in $Rng(\theta)$.

Formally

- For a variable $x \in Dom(\theta)$ we set $x\theta = \theta(x)$.
- For a variable $x \notin Dom(\theta)$ we set $x\theta = x$.
- Extend "homomorphically": $f(t_1, \ldots, t_n)\theta = f(t_1\theta, \ldots, t_n\theta)$.

Example (Why "simultaneous"?)

With $\theta = \{x/f(y), y/c\}$ we want $g(x, y)\theta$ to be

$$g(x, y)\theta = g(f(y), c).$$

Generality Order On Terms

Definition

- $s\theta$ is called an instance of s

Generality Order On Terms

Definition

- $s\theta$ is called an instance of s
- if θ is a renaming, then $s\theta$ is called a variant of s

Generality Order On Terms

Definition

- $s\theta$ is called an instance of s
- if θ is a renaming, then $s\theta$ is called a variant of s
- s is more general than t if t is an instance of s

Generality Order On Terms

Definition

- $s\theta$ is called an instance of s
- if θ is a renaming, then $s\theta$ is called a variant of s
- s is more general than t if t is an instance of s

Generality Order On Terms

Definition

- $s\theta$ is called an instance of s
- if θ is a renaming, then $s\theta$ is called a variant of s
- s is more general than t if t is an instance of s

Example

- $f(y, x)$ is a variant of $f(x, y)$,

Generality Order On Terms

Definition

- $s\theta$ is called an instance of s
- if θ is a renaming, then $s\theta$ is called a variant of s
- s is more general than t if t is an instance of s

Example

- $f(y, x)$ is a variant of $f(x, y)$, since $f(y, x) = f(x, y)\{x/y, y/x\}$

Generality Order On Terms

Definition

- $s\theta$ is called an instance of s
- if θ is a renaming, then $s\theta$ is called a variant of s
- s is more general than t if t is an instance of s

Example

- $f(y, x)$ is a variant of $f(x, y)$, since $f(y, x) = f(x, y)\{x/y, y/x\}$
- $f(x, y')$ is a variant of $f(x, y)$,

Generality Order On Terms

Definition

- $s\theta$ is called an instance of s
- if θ is a renaming, then $s\theta$ is called a variant of s
- s is more general than t if t is an instance of s

Example

- $f(y, x)$ is a variant of $f(x, y)$, since $f(y, x) = f(x, y)\{x/y, y/x\}$
- $f(x, y')$ is a variant of $f(x, y)$, since $f(x, y') = f(x, y)\{y/y', y'/y\}$

Generality Order On Terms

Definition

- $s\theta$ is called an instance of s
- if θ is a renaming, then $s\theta$ is called a variant of s
- s is more general than t if t is an instance of s

Example

- $f(y, x)$ is a variant of $f(x, y)$, since $f(y, x) = f(x, y)\{x/y, y/x\}$
- $f(x, y')$ is a variant of $f(x, y)$, since $f(x, y') = f(x, y)\{y/y', y'/y\}$
  (Note: we had to add a binding y'/y to get a renaming.)

Generality Order On Terms

Definition

- $s\theta$ is called an instance of s
- if θ is a renaming, then $s\theta$ is called a variant of s
- s is more general than t if t is an instance of s

Example

- $f(y, x)$ is a variant of $f(x, y)$, since $f(y, x) = f(x, y)\{x/y, y/x\}$
- $f(x, y')$ is a variant of $f(x, y)$, since $f(x, y') = f(x, y)\{y/y', y'/y\}$
  (Note: we had to add a binding y'/y to get a renaming.)
- $f(x, x)$ is *not a variant* of $f(x, y)$.

Generality Order On Terms

Definition

- $s\theta$ is called an instance of s
- if θ is a renaming, then $s\theta$ is called a variant of s
- s is more general than t if t is an instance of s

Example

- $f(y, x)$ is a variant of $f(x, y)$, since $f(y, x) = f(x, y)\{x/y, y/x\}$
- $f(x, y')$ is a variant of $f(x, y)$, since $f(x, y') = f(x, y)\{y/y', y'/y\}$
  (Note: we had to add a binding y'/y to get a renaming.)
- $f(x, x)$ is *not a variant* of $f(x, y)$. Why?

Generality Order On Terms

Definition

- $s\theta$ is called an instance of s
- if θ is a renaming, then $s\theta$ is called a variant of s
- s is more general than t if t is an instance of s

Example

- $f(y, x)$ is a variant of $f(x, y)$, since $f(y, x) = f(x, y)\{x/y, y/x\}$
- $f(x, y')$ is a variant of $f(x, y)$, since $f(x, y') = f(x, y)\{y/y', y'/y\}$
  (Note: we had to add a binding y'/y to get a renaming.)
- $f(x, x)$ is *not a variant* of $f(x, y)$. Why?

Lemma (Variant)

A term t is a variant of s iff t is an instance of s and s an instance of t.

Composition

The composition of substitutions θ and η, written $\theta\eta$, is a substitution satisfying $(\theta\eta)(x) = (x\theta)\eta$ for every variable x.

Composition

The composition of substitutions θ and η, written $\theta\eta$, is a substitution satisfying $(\theta\eta)(x) = (x\theta)\eta$ for every variable x. (Why is it finite?)

Composition

The composition of substitutions θ and η, written $\theta\eta$, is a substitution satisfying $(\theta\eta)(x) = (x\theta)\eta$ for every variable x. (Why is it finite?)

Lemma (Composition)

Let $\theta = \{x_1/t_1, \ldots, x_n/t_n\}$ and $\eta = \{y_1/s_1, \ldots, y_m/s_m\}$. Then $\theta\eta$ can be constructed from the sequence

$$x_1/t_1\eta, \ldots, x_n/t_n\eta, y_1/s_1, \ldots, y_m/s_m$$

1. by removing the bindings $x_i/t_i\eta$ for which $x_i = t_i\eta$ and
2. those bindings y_j/s_j for which $y_j \in \{x_1, \ldots, x_n\}$ and
3. forming a substitution from the resulting sequence.

Composition

The composition of substitutions θ and η, written $\theta\eta$, is a substitution satisfying $(\theta\eta)(x) = (x\theta)\eta$ for every variable x. (Why is it finite?)

Lemma (Composition)

Let $\theta = \{x_1/t_1, \ldots, x_n/t_n\}$ and $\eta = \{y_1/s_1, \ldots, y_m/s_m\}$. Then $\theta\eta$ can be constructed from the sequence

$$x_1/t_1\eta, \ldots, x_n/t_n\eta, y_1/s_1, \ldots, y_m/s_m$$

1. by removing the bindings $x_i/t_i\eta$ for which $x_i = t_i\eta$ and
2. those bindings y_j/s_j for which $y_j \in \{x_1, \ldots, x_n\}$ and
3. forming a substitution from the resulting sequence.

Example

For $\theta = \{u/z, x/3, y/f(x, 1)\}$ and $\eta = \{x/4, z/u\}$ we get

Composition

The composition of substitutions θ and η, written $\theta\eta$, is a substitution satisfying $(\theta\eta)(x) = (x\theta)\eta$ for every variable x. (Why is it finite?)

Lemma (Composition)

Let $\theta = \{x_1/t_1, \ldots, x_n/t_n\}$ and $\eta = \{y_1/s_1, \ldots, y_m/s_m\}$. Then $\theta\eta$ can be constructed from the sequence

$$x_1/t_1\eta, \ldots, x_n/t_n\eta, y_1/s_1, \ldots, y_m/s_m$$

1. by removing the bindings $x_i/t_i\eta$ for which $x_i = t_i\eta$ and
2. those bindings y_j/s_j for which $y_j \in \{x_1, \ldots, x_n\}$ and
3. forming a substitution from the resulting sequence.

Example

For $\theta = \{u/z, x/3, y/f(x, 1)\}$ and $\eta = \{x/4, z/u\}$ we get

$$\theta\eta = \{x/3, y = f(4, 1), z/u\}.$$

A Substitution Ordering

Let θ and τ be substitutions. We say that θ is more general than τ if $\tau = \theta\eta$ for some substitution η.

A Substitution Ordering

Let θ and τ be substitutions. We say that θ is more general than τ if $\tau = \theta\eta$ for some substitution η.

Example

- $\{x/y\}$ is more general than $\{x/a, y/a\}$

A Substitution Ordering

Let θ and τ be substitutions. We say that θ is more general than τ if $\tau = \theta\eta$ for some substitution η.

Example

- $\{x/y\}$ is more general than $\{x/a, y/a\}$ (via $\{x/y\}\{y/a\} = \{x/a, y/a\}$)

A Substitution Ordering

Let θ and τ be substitutions. We say that θ is more general than τ if $\tau = \theta\eta$ for some substitution η.

Example

- $\{x/y\}$ is more general than $\{x/a, y/a\}$
  (via $\{x/y\}\{y/a\} = \{x/a, y/a\}$)
- $\{x/y\}$ is not more general than $\tau = \{x/a\}$

A Substitution Ordering

Let θ and τ be substitutions. We say that θ is more general than τ if $\tau = \theta\eta$ for some substitution η.

Example

- $\{x/y\}$ is more general than $\{x/a, y/a\}$
  (via $\{x/y\}\{y/a\} = \{x/a, y/a\}$)
- $\{x/y\}$ is not more general than $\tau = \{x/a\}$
  (Assume $\{x/a\} = \{x/y\}\eta$ for some η. Since $x/a \in \tau = \{x/y\}\eta$, we have $y/a \in \eta$, which implies $y \in Dom(\tau)$. A contradition.)

Unifiers

Definition

- A substitution θ is called a unifier of terms s and t if $s\theta = t\theta$.

Unifiers

Definition

- A substitution θ is called a unifier of terms s and t if $s\theta = t\theta$.
- Terms s and t are called unifiable if $s\theta = t\theta$ for some θ.

Unifiers

Definition

- A substitution θ is called a unifier of terms s and t if $s\theta = t\theta$.
- Terms s and t are called unifiable if $s\theta = t\theta$ for some θ.
- θ is called a most general unifier (mgu) of terms s and t if

Unifiers

Definition

- A substitution θ is called a unifier of terms s and t if $s\theta = t\theta$.
- Terms s and t are called unifiable if $s\theta = t\theta$ for some θ.
- θ is called a most general unifier (mgu) of terms s and t if
  - it is a unifier of s and t,

Unifiers

Definition

- A substitution θ is called a unifier of terms s and t if $s\theta = t\theta$.
- Terms s and t are called unifiable if $s\theta = t\theta$ for some θ.
- θ is called a most general unifier (mgu) of terms s and t if
  - it is a unifier of s and t,
  - it is more general than all unifiers of s and t.

Unifiers

Definition

- A substitution θ is called a unifier of terms s and t if $s\theta = t\theta$.
- Terms s and t are called unifiable if $s\theta = t\theta$ for some θ.
- θ is called a most general unifier (mgu) of terms s and t if
  - it is a unifier of s and t,
  - it is more general than all unifiers of s and t.

Unifiers

Definition

- A substitution θ is called a unifier of terms s and t if $s\theta = t\theta$.
- Terms s and t are called unifiable if $s\theta = t\theta$ for some θ.
- θ is called a most general unifier (mgu) of terms s and t if
  - it is a unifier of s and t,
  - it is more general than all unifiers of s and t.

Example

Consider terms $s = f(g(x, a), z)$ and $t = f(y, b)$. Then

Unifiers

Definition

- A substitution θ is called a unifier of terms s and t if $s\theta = t\theta$.
- Terms s and t are called unifiable if $s\theta = t\theta$ for some θ.
- θ is called a most general unifier (mgu) of terms s and t if
  - it is a unifier of s and t,
  - it is more general than all unifiers of s and t.

Example

Consider terms $s = f(g(x, a), z)$ and $t = f(y, b)$. Then

- $\tau = \{x/c, y/g(c, a), z/b\}$ is a unifier of s and t,

Unifiers

Definition

- A substitution θ is called a unifier of terms s and t if $s\theta = t\theta$.
- Terms s and t are called unifiable if $s\theta = t\theta$ for some θ.
- θ is called a most general unifier (mgu) of terms s and t if
  - it is a unifier of s and t,
  - it is more general than all unifiers of s and t.

Example

Consider terms $s = f(g(x, a), z)$ and $t = f(y, b)$. Then

- $\tau = \{x/c, y/g(c, a), z/b\}$ is a unifier of s and t,
- so is $\theta = \{y/g(x, a), z/b\}$, which is more general, since

Unifiers

Definition

- A substitution θ is called a unifier of terms s and t if $s\theta = t\theta$.
- Terms s and t are called unifiable if $s\theta = t\theta$ for some θ.
- θ is called a most general unifier (mgu) of terms s and t if
  - it is a unifier of s and t,
  - it is more general than all unifiers of s and t.

Example

Consider terms $s = f(g(x, a), z)$ and $t = f(y, b)$. Then

- $\tau = \{x/c, y/g(c, a), z/b\}$ is a unifier of s and t,
- so is $\theta = \{y/g(x, a), z/b\}$, which is more general, since $\{x/c, y/g(c, a), z/b\} = \{y/g(x, a), z/b\}\{x/c\}$

Unifiers

Definition

- A substitution θ is called a unifier of terms s and t if $s\theta = t\theta$.
- Terms s and t are called unifiable if $s\theta = t\theta$ for some θ.
- θ is called a most general unifier (mgu) of terms s and t if
  - it is a unifier of s and t,
  - it is more general than all unifiers of s and t.

Example

Consider terms $s = f(g(x, a), z)$ and $t = f(y, b)$. Then

- $\tau = \{x/c, y/g(c, a), z/b\}$ is a unifier of s and t,
- so is $\theta = \{y/g(x, a), z/b\}$, which is more general, since $\{x/c, y/g(c, a), z/b\} = \{y/g(x, a), z/b\}\{x/c\}$ in fact, θ is an mgu

Non-Unifiers

Example

- $f(g(x, a), z)$ and $f(g(x, b), b)$ are not unifiable

Non-Unifiers

Example

- $f(g(x, a), z)$ and $f(g(x, b), b)$ are not unifiable (no θ can make $a\theta = b\theta$)

Non-Unifiers

Example

- $f(g(x, a), z)$ and $f(g(x, b), b)$ are not unifiable (no θ can make $a\theta = b\theta$)
- $g(x, a)$ and $g(f(x), a)$ are not unifiable

Non-Unifiers

Example

- $f(g(x, a), z)$ and $f(g(x, b), b)$ are not unifiable
  (no θ can make $a\theta = b\theta$)
- $g(x, a)$ and $g(f(x), a)$ are not unifiable
  ($x\theta$ is always a proper subterm of $f(x)\theta$)

Non-Unifiers

Example

- $f(g(x,a),z)$ and $f(g(x,b),b)$ are not unifiable
  (no θ can make $a\theta = b\theta$)
- $g(x,a)$ and $g(f(x),a)$ are not unifiable
  ($x\theta$ is always a proper subterm of $f(x)\theta$)

Recall the convention about symbols and variables. Here we had: $\Sigma_f = \{a/0, b/0, f/1, g/2\}$ (constants and function symbols) and $\mathcal{V} = \{x, y, z, \ldots\}$ (first-order variables).

Unifying Sets of Pairs of Terms

Let $E = \{s_1 \doteq t_1, \ldots, s_n \doteq t_n\}$ be a set of term equations.

Unifying Sets of Pairs of Terms

Let $E = \{s_1 \doteq t_1, \ldots, s_n \doteq t_n\}$ be a set of term equations.

Unifying Sets of Pairs of Terms

Let $E = \{s_1 \doteq t_1, \ldots, s_n \doteq t_n\}$ be a set of term equations.

- θ is a unifier of E if $s_i\theta = t_i\theta$ for $i = 1, \ldots, n$

Unifying Sets of Pairs of Terms

Let $E = \{s_1 \doteq t_1, \ldots, s_n \doteq t_n\}$ be a set of term equations.

- θ is a unifier of E if $s_i\theta = t_i\theta$ for $i = 1, \ldots, n$
- θ is a most general unifier (mgu) of E, if

Unifying Sets of Pairs of Terms

Let $E = \{s_1 \doteq t_1, \ldots, s_n \doteq t_n\}$ be a set of term equations.

- θ is a unifier of E if $s_i\theta = t_i\theta$ for $i = 1, \ldots, n$
- θ is a most general unifier (mgu) of E, if
  - it is a unifier of E and is more general than all unifiers of E

Unifying Sets of Pairs of Terms

Let $E = \{s_1 \doteq t_1, \ldots, s_n \doteq t_n\}$ be a set of term equations.

- θ is a unifier of E if $s_i\theta = t_i\theta$ for $i = 1, \ldots, n$
- θ is a most general unifier (mgu) of E, if
  - it is a unifier of E and is more general than all unifiers of E
- a set of equations E is a called solved, if it is of the form $\{x_1 \doteq t_1, \ldots, x_n \doteq t_n\}$ where
  - the variables x_i are distinct, and
  - none of the x_i's occurs in any $Var(t_j)$

Unifying Sets of Pairs of Terms

Let $E = \{s_1 \doteq t_1, \ldots, s_n \doteq t_n\}$ be a set of term equations.

- θ is a unifier of E if $s_i\theta = t_i\theta$ for $i = 1, \ldots, n$
- θ is a most general unifier (mgu) of E, if
  - it is a unifier of E and is more general than all unifiers of E
- a set of equations E is a called solved, if it is of the form $\{x_1 \doteq t_1, \ldots, x_n \doteq t_n\}$ where
  - the variables x_i are distinct, and
  - none of the x_i's occurs in any $Var(t_j)$

Lemma (Solved Form)

If $E = \{x_1 \doteq t_1, \ldots, x_n \doteq t_n\}$ is solved, then $\theta = \{x_1/t_1, \ldots, x_n/t_n\}$ is an mgu of E.

Unifying Sets of Pairs of Terms

Let $E = \{s_1 \doteq t_1, \ldots, s_n \doteq t_n\}$ be a set of term equations.

- θ is a unifier of E if $s_i\theta = t_i\theta$ for $i = 1, \ldots, n$
- θ is a most general unifier (mgu) of E, if
  - it is a unifier of E and is more general than all unifiers of E
- a set of equations E is a called solved, if it is of the form $\{x_1 \doteq t_1, \ldots, x_n \doteq t_n\}$ where
  - the variables x_i are distinct, and
  - none of the x_i's occurs in any $Var(t_j)$

Lemma (Solved Form)

If $E = \{x_1 \doteq t_1, \ldots, x_n \doteq t_n\}$ is solved, then $\theta = \{x_1/t_1, \ldots, x_n/t_n\}$ is an mgu of E. **Proof:**

- $x_i\theta = t_i$

Unifying Sets of Pairs of Terms

Let $E = \{s_1 \doteq t_1, \ldots, s_n \doteq t_n\}$ be a set of term equations.

- θ is a unifier of E if $s_i\theta = t_i\theta$ for $i = 1, \ldots, n$
- θ is a most general unifier (mgu) of E, if
  - it is a unifier of E and is more general than all unifiers of E
- a set of equations E is a called solved, if it is of the form $\{x_1 \doteq t_1, \ldots, x_n \doteq t_n\}$ where
  - the variables x_i are distinct, and
  - none of the x_i's occurs in any $Var(t_j)$

Lemma (Solved Form)

If $E = \{x_1 \doteq t_1, \ldots, x_n \doteq t_n\}$ is solved, then $\theta = \{x_1/t_1, \ldots, x_n/t_n\}$ is an mgu of E. **Proof:**

- $x_i\theta = t_i = t_i\theta$,

Unifying Sets of Pairs of Terms

Let $E = \{s_1 \doteq t_1, \ldots, s_n \doteq t_n\}$ be a set of term equations.

- θ is a unifier of E if $s_i\theta = t_i\theta$ for $i = 1, \ldots, n$
- θ is a most general unifier (mgu) of E, if
  - it is a unifier of E and is more general than all unifiers of E
- a set of equations E is a called solved, if it is of the form $\{x_1 \doteq t_1, \ldots, x_n \doteq t_n\}$ where
  - the variables x_i are distinct, and
  - none of the x_i's occurs in any $Var(t_j)$

Lemma (Solved Form)

If $E = \{x_1 \doteq t_1, \ldots, x_n \doteq t_n\}$ is solved, then $\theta = \{x_1/t_1, \ldots, x_n/t_n\}$ is an mgu of E. **Proof:**

- $x_i\theta = t_i = t_i\theta$,
- let η be a unifier of E:
  - $x_i\eta = t_i\eta = x_i\theta\eta$ for $i = 1, \ldots, n$, and

Unifying Sets of Pairs of Terms

Let $E = \{s_1 \doteq t_1, \ldots, s_n \doteq t_n\}$ be a set of term equations.

- θ is a unifier of E if $s_i\theta = t_i\theta$ for $i = 1, \ldots, n$
- θ is a most general unifier (mgu) of E, if
  - it is a unifier of E and is more general than all unifiers of E
- a set of equations E is a called solved, if it is of the form $\{x_1 \doteq t_1, \ldots, x_n \doteq t_n\}$ where
  - the variables x_i are distinct, and
  - none of the x_i's occurs in any $Var(t_j)$

Lemma (Solved Form)

If $E = \{x_1 \doteq t_1, \ldots, x_n \doteq t_n\}$ is solved, then $\theta = \{x_1/t_1, \ldots, x_n/t_n\}$ is an mgu of E. **Proof:**

- $x_i\theta = t_i = t_i\theta$,
- let η be a unifier of E:
  - $x_i\eta = t_i\eta = x_i\theta\eta$ for $i = 1, \ldots, n$, and
  - $x\eta = x\theta\eta$ for any $x \notin \{x_1, \ldots, x_n\}$.

Unifying Sets of Pairs of Terms

Let $E = \{s_1 \doteq t_1, \ldots, s_n \doteq t_n\}$ be a set of term equations.

- θ is a unifier of E if $s_i\theta = t_i\theta$ for $i = 1, \ldots, n$
- θ is a most general unifier (mgu) of E, if
  - it is a unifier of E and is more general than all unifiers of E
- a set of equations E is a called solved, if it is of the form $\{x_1 \doteq t_1, \ldots, x_n \doteq t_n\}$ where
  - the variables x_i are distinct, and
  - none of the x_i's occurs in any $Var(t_j)$

Lemma (Solved Form)

If $E = \{x_1 \doteq t_1, \ldots, x_n \doteq t_n\}$ is solved, then $\theta = \{x_1/t_1, \ldots, x_n/t_n\}$ is an mgu of E. **Proof:**

- $x_i\theta = t_i = t_i\theta$,
- let η be a unifier of E:
  - $x_i\eta = t_i\eta = x_i\theta\eta$ for $i = 1, \ldots, n$, and
  - $x\eta = x\theta\eta$ for any $x \notin \{x_1, \ldots, x_n\}$.

  Thus $\eta = \theta\eta$.

Unifying Sets of Pairs of Terms

Let $E = \{s_1 \doteq t_1, \ldots, s_n \doteq t_n\}$ be a set of term equations.

- θ is a unifier of E if $s_i\theta = t_i\theta$ for $i = 1, \ldots, n$
- θ is a most general unifier (mgu) of E, if
  - it is a unifier of E and is more general than all unifiers of E
- a set of equations E is a called solved, if it is of the form $\{x_1 \doteq t_1, \ldots, x_n \doteq t_n\}$ where
  - the variables x_i are distinct, and
  - none of the x_i's occurs in any $Var(t_j)$

Lemma (Solved Form)

If $E = \{x_1 \doteq t_1, \ldots, x_n \doteq t_n\}$ is solved, then $\theta = \{x_1/t_1, \ldots, x_n/t_n\}$ is an mgu of E. **Proof:**

- $x_i\theta = t_i = t_i\theta$,
- let η be a unifier of E:
  - $x_i\eta = t_i\eta = x_i\theta\eta$ for $i = 1, \ldots, n$, and
  - $x\eta = x\theta\eta$ for any $x \notin \{x_1, \ldots, x_n\}$.

  Thus $\eta = \theta\eta$. (Such an mgu is, by the way, called strong.)

Martelli-Montanari Unification Algorithm

Martelli-Montanari Unification Algorithm

Let E be a set of term equations. As long as possible, choose an equation in E of a form below and perform the associated action:

Martelli-Montanari Unification Algorithm

Let E be a set of term equations. As long as possible, choose an equation in E of a form below and perform the associated action:

1. $f(s_1, \ldots, s_n) \doteq f(t_1, \ldots, t_n)$

Martelli-Montanari Unification Algorithm

Let E be a set of term equations. As long as possible, choose an equation in E of a form below and perform the associated action:

1. $f(s_1, \ldots, s_n) \doteq f(t_1, \ldots, t_n)$ where $n \geq 0$

Martelli-Montanari Unification Algorithm

Let E be a set of term equations. As long as possible, choose an equation in E of a form below and perform the associated action:

1. $f(s_1, \ldots, s_n) \doteq f(t_1, \ldots, t_n)$ where $n \geq 0$

   $\longrightarrow$ replace with $s_1 \doteq t_1, \ldots, s_n \doteq t_n$

Martelli-Montanari Unification Algorithm

Let E be a set of term equations. As long as possible, choose an equation in E of a form below and perform the associated action:

1. $f(s_1, \ldots, s_n) \doteq f(t_1, \ldots, t_n)$ where $n \geq 0$
   $\longrightarrow$ replace with $s_1 \doteq t_1, \ldots, s_n \doteq t_n$
2. $f(s_1, \ldots, s_n) \doteq g(t_1, \ldots, t_n)$ and $f \neq g$ (and $n \geq 0$)
   $\longrightarrow$ halt with failure

Martelli-Montanari Unification Algorithm

Let E be a set of term equations. As long as possible, choose an equation in E of a form below and perform the associated action:

1. $f(s_1, \ldots, s_n) \doteq f(t_1, \ldots, t_n)$ where $n \geq 0$

   $\longrightarrow$ replace with $s_1 \doteq t_1, \ldots, s_n \doteq t_n$

2. $f(s_1, \ldots, s_n) \doteq g(t_1, \ldots, t_n)$ and $f \neq g$ (and $n \geq 0$)

   $\longrightarrow$ halt with failure

3. $x \doteq x$

   $\longrightarrow$ delete the equation

Martelli-Montanari Unification Algorithm

Let E be a set of term equations. As long as possible, choose an equation in E of a form below and perform the associated action:

1. $f(s_1, \ldots, s_n) \doteq f(t_1, \ldots, t_n)$ where $n \geq 0$

   $\longrightarrow$ replace with $s_1 \doteq t_1, \ldots, s_n \doteq t_n$
2. $f(s_1, \ldots, s_n) \doteq g(t_1, \ldots, t_n)$ and $f \neq g$ (and $n \geq 0$)

   $\longrightarrow$ halt with failure
3. $x \doteq x$

   $\longrightarrow$ delete the equation
4. $t \doteq x$ where t is not a variable

   $\longrightarrow$ replace by $x \doteq t$

Martelli-Montanari Unification Algorithm

Let E be a set of term equations. As long as possible, choose an equation in E of a form below and perform the associated action:

1. $f(s_1, \ldots, s_n) \doteq f(t_1, \ldots, t_n)$ where $n \geq 0$
   $\longrightarrow$ replace with $s_1 \doteq t_1, \ldots, s_n \doteq t_n$
2. $f(s_1, \ldots, s_n) \doteq g(t_1, \ldots, t_n)$ and $f \neq g$ (and $n \geq 0$)
   $\longrightarrow$ halt with failure
3. $x \doteq x$
   $\longrightarrow$ delete the equation
4. $t \doteq x$ where t is not a variable
   $\longrightarrow$ replace by $x \doteq t$
5. $x \doteq t$ where $x \notin Var(t)$ and x occurs in elsewhere
   $\longrightarrow$ perform $\{x/t\}$ on all other pairs

Martelli-Montanari Unification Algorithm

Let E be a set of term equations. As long as possible, choose an equation in E of a form below and perform the associated action:

1. $f(s_1, \ldots, s_n) \doteq f(t_1, \ldots, t_n)$ where $n \geq 0$
   $\longrightarrow$ replace with $s_1 \doteq t_1, \ldots, s_n \doteq t_n$
2. $f(s_1, \ldots, s_n) \doteq g(t_1, \ldots, t_n)$ and $f \neq g$ (and $n \geq 0$)
   $\longrightarrow$ halt with failure
3. $x \doteq x$
   $\longrightarrow$ delete the equation
4. $t \doteq x$ where t is not a variable
   $\longrightarrow$ replace by $x \doteq t$
5. $x \doteq t$ where $x \notin Var(t)$ and x occurs in elsewhere
   $\longrightarrow$ perform $\{x/t\}$ on all other pairs
6. $x \doteq t$ where $x \in Var(t)$ and $x \neq t$
   $\longrightarrow$ halt with failure

Martelli-Montanari Unification Algorithm

Let E be a set of term equations. As long as possible, choose an equation in E of a form below and perform the associated action:

1. $f(s_1, \ldots, s_n) \doteq f(t_1, \ldots, t_n)$ where $n \geq 0$
   $\longrightarrow$ replace with $s_1 \doteq t_1, \ldots, s_n \doteq t_n$
2. $f(s_1, \ldots, s_n) \doteq g(t_1, \ldots, t_n)$ and $f \neq g$ (and $n \geq 0$)
   $\longrightarrow$ halt with failure
3. $x \doteq x$
   $\longrightarrow$ delete the equation
4. $t \doteq x$ where t is not a variable
   $\longrightarrow$ replace by $x \doteq t$
5. $x \doteq t$ where $x \notin Var(t)$ and x occurs in elsewhere
   $\longrightarrow$ perform $\{x/t\}$ on all other pairs
6. $x \doteq t$ where $x \in Var(t)$ and $x \neq t$
   $\longrightarrow$ halt with failure

Terminate with success when no other action can be performed.

Martelli-Montanari Unification Algorithm

Let E be a set of term equations. As long as possible, choose an equation in E of a form below and perform the associated action:

1. $f(s_1, \ldots, s_n) \doteq f(t_1, \ldots, t_n)$ where $n \geq 0$
   $\longrightarrow$ replace with $s_1 \doteq t_1, \ldots, s_n \doteq t_n$
2. $f(s_1, \ldots, s_n) \doteq g(t_1, \ldots, t_n)$ and $f \neq g$ (and $n \geq 0$)
   $\longrightarrow$ halt with failure
3. $x \doteq x$
   $\longrightarrow$ delete the equation
4. $t \doteq x$ where t is not a variable
   $\longrightarrow$ replace by $x \doteq t$
5. $x \doteq t$ where $x \notin Var(t)$ and x occurs in elsewhere
   $\longrightarrow$ perform $\{x/t\}$ on all other pairs
6. $x \doteq t$ where $x \in Var(t)$ and $x \neq t$
   $\longrightarrow$ halt with failure

Terminate with success when no other action can be performed.

(We call: 2. the clash-failure and 6. the "occurs check"-failure.)

Using Martelli-Montanari

Sets E_1 and E_2 are equivalent if they have the same set of unifiers.

Using Martelli-Montanari

Sets E_1 and E_2 are equivalent if they have the same set of unifiers.

To use the MM-algorithm to unify s and t, we use it on $E = \{s \doteq t\}$.

Using Martelli-Montanari

Sets E_1 and E_2 are equivalent if they have the same set of unifiers.

To use the MM-algorithm to unify s and t, we use it on $E = \{s \doteq t\}$.

Theorem (MM is strongly correct)

If the original set E has a unifier, MM successfully terminates with an equivalent solved set E'. Otherwise, it terminates with failure.

Using Martelli-Montanari

Sets E_1 and E_2 are equivalent if they have the same set of unifiers.

To use the MM-algorithm to unify s and t, we use it on $E = \{s \doteq t\}$.

Theorem (MM is strongly correct)

If the original set E has a unifier, MM successfully terminates with an equivalent solved set E'. Otherwise, it terminates with failure.

Proof:

- MM terminates

Using Martelli-Montanari

Sets E_1 and E_2 are equivalent if they have the same set of unifiers.

To use the MM-algorithm to unify s and t, we use it on $E = \{s \doteq t\}$.

Theorem (MM is strongly correct)

If the original set E has a unifier, MM successfully terminates with an equivalent solved set E'. Otherwise, it terminates with failure.

Proof:

- MM terminates
- each step preserves equivalence

Using Martelli-Montanari

Sets E_1 and E_2 are equivalent if they have the same set of unifiers.

To use the MM-algorithm to unify s and t, we use it on $E = \{s \doteq t\}$.

Theorem (MM is strongly correct)

If the original set E has a unifier, MM successfully terminates with an equivalent solved set E'. Otherwise, it terminates with failure.

Proof:

- MM terminates
- each step preserves equivalence
- on success, the final set is solved

Using Martelli-Montanari

Sets E_1 and E_2 are equivalent if they have the same set of unifiers.

To use the MM-algorithm to unify s and t, we use it on $E = \{s \doteq t\}$.

Theorem (MM is strongly correct)

If the original set E has a unifier, MM successfully terminates with an equivalent solved set E'. Otherwise, it terminates with failure.

Proof:

- MM terminates
- each step preserves equivalence
- on success, the final set is solved
- on failure, there is no unifier

Unifiers may be Exponential

$$f(x_1) \doteq f(g(x_0, x_0))$$
$$\theta_1 = \{x_1 / g(x_0, x_0)\}$$

Unifiers may be Exponential

$$f(x_1) \doteq f(g(x_0, x_0))$$
$$\theta_1 = \{x_1 / g(x_0, x_0)\}$$

$$f(x_1, x_2) \doteq f(g(x_0, x_0), g(x_1, x_1))$$
$$\theta_2 = \theta_1 \cup \{x_2 / g(g(x_0, x_0), g(x_0, x_0))\}$$

Unifiers may be Exponential

$$f(x_1) \doteq f(g(x_0, x_0))$$
$$\theta_1 = \{x_1 / g(x_0, x_0)\}$$

$$f(x_1, x_2) \doteq f(g(x_0, x_0), g(x_1, x_1))$$
$$\theta_2 = \theta_1 \cup \{x_2 / g(g(x_0, x_0), g(x_0, x_0))\}$$

$$f(x_1, x_2, x_3) \doteq f(g(x_0, x_0), g(x_1, x_1), g(x_2, x_2))$$
$$\theta_3 = \theta_2 \cup \{x_3 / g(g(g(x_0, x_0), g(x_0, x_0)), g(g(x_0, x_0), g(x_0, x_0)))\}$$
$$\vdots$$

A First-order Resolution Calculus in Full

Consists of two inference rules:

- Binary resolution, denoted by BR:

$$\frac{L_1 \vee C_1 \quad \neg L_2 \vee C_2}{(C_1 \vee C_2)\sigma} \text{ (BR)}.$$

- Factoring, denoted by Fact:

$$\frac{L_1 \vee L_2 \vee C}{(L_1 \vee C)\sigma} \text{ (Fact)}.$$

where, in both cases, σ is the most general unifier of L_1 and L_2.

A First-order Resolution Calculus in Full

Consists of two inference rules:

- Binary resolution, denoted by BR:

$$\frac{L_1 \vee C_1 \quad \neg L_2 \vee C_2}{(C_1 \vee C_2)\sigma} \text{ (BR)}.$$

- Factoring, denoted by Fact:

$$\frac{L_1 \vee L_2 \vee C}{(L_1 \vee C)\sigma} \text{ (Fact)}.$$

where, in both cases, σ is the most general unifier of L_1 and L_2.

Note: For completeness, we always need to make sure that the premises are variable-disjoint!

A First-order Resolution Calculus in Full

Consists of two inference rules:

- Binary resolution, denoted by BR:

$$\frac{L_1 \vee C_1 \quad \neg L_2 \vee C_2}{(C_1 \vee C_2)\sigma} \text{ (BR)}.$$

- Factoring, denoted by Fact:

$$\frac{L_1 \vee L_2 \vee C}{(L_1 \vee C)\sigma} \text{ (Fact)}.$$

where, in both cases, σ is the most general unifier of L_1 and L_2.

Note: For completeness, we always need to make sure that the premises are variable-disjoint! Done by a so called renaming apart.