Automated Termination Analysis

Jürgen Giesl

LuFG Informatik 2, RWTH Aachen University, Germany

VTSA '12, Saarbrücken, Germany

Overview

I. Termination of Term Rewriting

1. Termination of Term Rewrite Systems
2. Non-Termination of Term Rewrite Systems
3. Complexity of Term Rewrite Systems
4. Termination of Integer Term Rewrite Systems

II. Termination of Programs

1. Termination of Functional Programs (Haskell)
2. Termination of Logic Programs (Prolog)
3. Termination of Imperative Programs (Java)

Overview

I. Termination of Term Rewriting

1. Termination of Term Rewrite Systems (TCS '00, JAR '06)
2. Non-Termination of Term Rewrite Systems
3. Complexity of Term Rewrite Systems
4. Termination of Integer Term Rewrite Systems

II. Termination of Programs

1. Termination of Functional Programs (Haskell)
2. Termination of Logic Programs (Prolog)
3. Termination of Imperative Programs (Java)

Termination Analysis for TRSs

$$\mathcal{R}: \quad \begin{array}{rcl} \mathsf{plus}(x, 0) & \to & x \\ \mathsf{plus}(x, \mathsf{s}(y)) & \to & \mathsf{s}(\mathsf{plus}(x, y)) \end{array}$$

$\mathcal{R}$ is *terminating* iff there is no infinite evaluation $t_1 \to_{\mathcal{R}} t_2 \to_{\mathcal{R}} \ldots$

Computation of "2 + 1":

$$\begin{array}{rl} \mathsf{plus}(\mathsf{s}(\mathsf{s}(0)), \mathsf{s}(0)) & \to_{\mathcal{R}} \mathsf{s}(\mathsf{plus}(\mathsf{s}(\mathsf{s}(0)), 0)) \\ & \to_{\mathcal{R}} \mathsf{s}(\mathsf{s}(\mathsf{s}(0))) \end{array}$$

Termination Analysis for TRSs

$$\mathcal{R}: \quad \begin{array}{rcl} \mathsf{plus}(x, 0) & \to & x \\ \mathsf{plus}(x, \mathsf{s}(y)) & \to & \mathsf{s}(\mathsf{plus}(x, y)) \end{array}$$

$\mathcal{R}$ is *terminating* iff there is no infinite evaluation $t_1 \to_{\mathcal{R}} t_2 \to_{\mathcal{R}} \ldots$

- easier / more general than for functional programs
- termination technique for TRSs can be adapted to strategies, types, higher-order functions, ...
- suitable for automation
- **But:** halting problem is undecidable!
  $\Rightarrow$ automated termination proofs do not always succeed

Outline

1. Classical Techniques

- Lexicographic Path Order (with Status) *(Kamin & Lévy, 80)*
- Recursive Path Order (with Status) *(Dershowitz, 82)*
- Polynomial Order *(Lankford, 79)*

2. Dependency Pairs *(Arts & Giesl et al, 96 – today)*

- Proving Termination
- Proving Innermost Termination
- Integrating Other Termination Techniques

3. Other Recent Techniques

- Semantic Labeling *(Zantema, 95)*
- Match-Bounds *(Geser, Hofbauer, Waldmann, Zantema, 03 – today)*

1. Classical Techniques

$$\mathcal{R}: \quad \begin{array}{rcl} \mathsf{plus}(x, 0) & \to & x \\ \mathsf{plus}(x, \mathsf{s}(y)) & \to & \mathsf{s}(\mathsf{plus}(x, y)) \end{array}$$

$\mathcal{R}$ is *terminating* iff there is no infinite evaluation $t_1 \to_{\mathcal{R}} t_2 \to_{\mathcal{R}} \ldots$

Goal: Find order $\succ$ such that $l \succ r$ for all rules $l \to r \in \mathcal{R}$

- $\succ$ is well-founded: no infinite sequence $t_1 \succ t_2 \succ \ldots$
- $\succ$ is monotonic: if $t_1 \succ t_2$ then $f(\ldots t_1 \ldots) \succ f(\ldots t_2 \ldots)$
  if $\mathsf{plus}(x, 0) \succ x$ then $\mathsf{s}(\mathsf{plus}(x, 0)) \succ \mathsf{s}(x)$
- $\succ$ is stable: if $t_1 \succ t_2$ then $\sigma(t_1) \succ \sigma(t_2)$
  if $\mathsf{plus}(x, 0) \succ x$ then $\mathsf{plus}(0, 0) \succ 0$

Lexicographic Path Order (LPO)

$$\mathcal{R}: \quad \begin{array}{rcl} \mathsf{plus}(x,0) & \rightarrow & x \\ \mathsf{plus}(x,\mathsf{s}(y)) & \rightarrow & \mathsf{s}(\mathsf{plus}(x,y)) \end{array}$$

LPO is an order where $s \succ t$ iff

- $s = f(s_1 \ldots s_n)$, $s_i \succeq t$ for some i

  $\mathsf{plus}(x,0) \succ x$ since $x \succeq x$

- $s = f(\ldots)$, $t = g(t_1 \ldots t_n)$, $f > g$, and $s \succ t_i$ for all i (e.g. $\mathsf{plus} > \mathsf{s}$)

  $\mathsf{plus}(x,\mathsf{s}(y)) \succ \mathsf{s}(\mathsf{plus}(x,y))$ if $\mathsf{plus}(x,\mathsf{s}(y)) \succ \mathsf{plus}(x,y)$

- $s = f(s_1 \ldots s_{j-1} s_j \ldots s_n)$, $t = f(s_1 \ldots s_{j-1} t_j \ldots t_n)$, $s_j \succ t_j$, $s \succ t_i$ for $i > j$

  $\mathsf{plus}(x,\mathsf{s}(y)) \succ \mathsf{plus}(x,y)$ since $\mathsf{s}(y) \succ y$

  $\mathsf{plus}(\mathsf{s}(x),y) \succ \mathsf{plus}(x,\mathsf{s}(y))$ since $\mathsf{s}(x) \succ x$, $\mathsf{plus}(\mathsf{s}(x),y) \succ \mathsf{s}(y)$

LPO with Status (LPOS)

$$\mathcal{R}: \quad \begin{array}{rcl} \mathsf{plus}(x, 0) & \rightarrow & x \\ \mathsf{plus}(x, \mathsf{s}(y)) & \rightarrow & \mathsf{plus}(\mathsf{s}(x), y) \end{array}$$

LPO does lexicographic comparison from left to right:

- $s = f(s_1 ... s_{j-1} s_j ... s_n)$, $t = f(s_1 ... s_{j-1} t_j ... t_n)$, $s_j \succ t_j$, $s \succ t_i$ for $i > j$
  $\mathsf{plus}(x, \mathsf{s}(y)) \not\succ \mathsf{plus}(\mathsf{s}(x), y)$

We need lexicographic comparison from right to left:

- $s = f(s_1 ... s_j s_{j+1} ... s_n)$, $t = f(t_1 ... t_j s_{j+1} ... s_n)$, $s_j \succ t_j$, $s \succ t_i$ for $i < j$
  $\mathsf{plus}(x, \mathsf{s}(y)) \succ \mathsf{plus}(\mathsf{s}(x), y)$ since $\mathsf{s}(y) \succ y$, $\mathsf{plus}(x, \mathsf{s}(y)) \succ \mathsf{s}(x)$

LPO with Status: status(f) = permutation of 1, . . . , arity(f)
determines order in lexicographic comparison

Recursive (Multiset) Path Order (RPO)

$$\mathcal{R}: \quad \begin{array}{rcl} \mathsf{plus}(x, 0) & \to & x \\ \mathsf{plus}(x, \mathsf{s}(y)) & \to & \mathsf{s}(\mathsf{plus}(y, x)) \end{array}$$

RPO is an order where $s \succ t$ iff

- $s = f(s_1 \ldots s_n)$, $s_i \succeq t$ for some i
  $\mathsf{plus}(x, 0) \succ x$ since $x \succeq x$
- $s = f(\ldots)$, $t = g(t_1 \ldots t_n)$, $f > g$, and $s \succ t_i$ for all i (e.g. $\mathsf{plus} > \mathsf{s}$)
  $\mathsf{plus}(x, \mathsf{s}(y)) \succ \mathsf{s}(\mathsf{plus}(y, x))$ if $\mathsf{plus}(x, \mathsf{s}(y)) \succ \mathsf{plus}(y, x)$
- $s = f(s_1 ... s_n)$, $t = f(t_1 ... t_n)$, $\{s_1, ..., s_n\} \succ_{mul} \{t_1, ..., t_n\}$
  $\mathsf{plus}(x, \mathsf{s}(y)) \succ \mathsf{plus}(y, x)$ since $\{x, \mathsf{s}(y)\} \succ_{mul} \{y, x\}$

RPO with Status (RPOS)

$$\begin{aligned} \mathsf{plus}_1(0, y) &\to y \\ \mathsf{plus}_1(\mathsf{s}(x), y) &\to \mathsf{plus}_1(x, \mathsf{s}(y)) \end{aligned}$$

$$\begin{aligned} \mathsf{plus}_2(x, 0) &\to x \\ \mathsf{plus}_2(x, \mathsf{s}(y)) &\to \mathsf{plus}_2(\mathsf{s}(x), y) \end{aligned}$$

$$\begin{aligned} \mathsf{plus}_3(x, 0) &\to x \\ \mathsf{plus}_3(x, \mathsf{s}(y)) &\to \mathsf{s}(\mathsf{plus}_3(y, x)) \end{aligned}$$

RPO with Status: status(f) = permutation of 1, . . . , arity(f) or "*mul*"

status(plus_1) = (1,2)
status(plus_2) = (2,1)
status(plus_3) = *mul*

Polynomial Order

$$\mathcal{P}ol(\,\mathsf{plus}(0,y)\,) \;>\; \mathcal{P}ol(\,y\,)$$
$$\mathcal{P}ol(\,\mathsf{plus}(\mathsf{s}(x),y)\,) \;>\; \mathcal{P}ol(\,\mathsf{s}(\mathsf{plus}(x,y))\,)$$

$\mathcal{R}$ is *terminating* iff there is no infinite evaluation $t_1 \to_{\mathcal{R}} t_2 \to_{\mathcal{R}} \ldots$

Goal: Find order $\succ$ such that $l \succ r$ for all rules $l \to r \in \mathcal{R}$

Polynomial Order: $l \succ r$ iff $\mathcal{P}ol(l) > \mathcal{P}ol(r)$

$$\mathcal{P}ol(0) = 1$$
$$\mathcal{P}ol(\mathsf{s}(t)) = 1 + \mathcal{P}ol(t)$$
$$\mathcal{P}ol(\mathsf{plus}(t_1,t_2)) = 2\,\mathcal{P}ol(t_1) + \mathcal{P}ol(t_2)$$

1. Classical Techniques

- **Goal:** Find order $\succ$ such that $l \succ r$ for all rules $l \to r \in \mathcal{R}$
  - Lexicographic Path Order (Kamin & Levy 1980)
  - Recursive Path Order (Dershowitz 1982)
  - Recursive Path Order with Status
  - Polynomial Orders (Lankford 1979)
- all these orders are *simplification orders*: $f(\ldots t \ldots) \succ t$
- too restrictive for many important TRSs
- Dependency Pairs
  - Original method: Arts & Giesl (1996 - 2002)
  - New refinements: Giesl, Thiemann, Schneider-Kamp (since 2003)
    Middeldorp, Hirokawa (since 2001)

Dependency Pairs for Termination

$$
\begin{aligned}
\mathsf{minus}(x, \mathsf{0}) &\rightarrow x \\
\mathsf{minus}(\mathsf{s}(x), \mathsf{s}(y)) &\rightarrow \mathsf{minus}(x, y) \\
\mathsf{div}(\mathsf{0}, \mathsf{s}(y)) &\rightarrow \mathsf{0} \\
\mathsf{div}(\mathsf{s}(x), \mathsf{s}(y)) &\rightarrow \mathsf{s}(\mathsf{div}(\mathsf{minus}(x, y), \mathsf{s}(y)))
\end{aligned}
$$

- Standard Approach:
  Compare left- and right-hand sides of rules

- Problem:
  Automated techniques use *simplification orders*
  $\Rightarrow$ Failure!

- *Dependency Pair Approach:*
  Examine only those subterms which are responsible
  for starting new reductions

Dependency Pairs for Termination

$$
\begin{aligned}
\mathsf{minus}(x, 0) &\rightarrow x \\
\mathsf{minus}(\mathsf{s}(x), \mathsf{s}(y)) &\rightarrow \mathsf{minus}(x, y) \\
\mathsf{div}(0, \mathsf{s}(y)) &\rightarrow 0 \\
\mathsf{div}(\mathsf{s}(x), \mathsf{s}(y)) &\rightarrow \mathsf{s}(\mathsf{div}(\mathsf{minus}(x, y), \mathsf{s}(y)))
\end{aligned}
$$

- Defined Symbols: minus, div
  Constructors: 0, s

- **Definition**
  If $f(s_1, \ldots, s_n) \rightarrow C[g(t_1, \ldots, t_m)]$ is a rule and g is defined, then $\mathsf{F}(s_1, \ldots, s_n) \rightarrow \mathsf{G}(t_1, \ldots, t_m)$ is a *dependency pair*

$$
\begin{aligned}
\mathsf{M}(\mathsf{s}(x), \mathsf{s}(y)) &\rightarrow \mathsf{M}(x, y) \\
\mathsf{D}(\mathsf{s}(x), \mathsf{s}(y)) &\rightarrow \mathsf{M}(x, y) \\
\mathsf{D}(\mathsf{s}(x), \mathsf{s}(y)) &\rightarrow \mathsf{D}(\mathsf{minus}(x, y), \mathsf{s}(y))
\end{aligned}
$$

$$\begin{aligned} \mathsf{M}(\mathsf{s}(x),\mathsf{s}(y)) &\to \mathsf{M}(x,y) \\ \mathsf{D}(\mathsf{s}(x),\mathsf{s}(y)) &\to \mathsf{M}(x,y) \\ \mathsf{D}(\mathsf{s}(x),\mathsf{s}(y)) &\to \mathsf{D}(\mathsf{minus}(x,y),\mathsf{s}(y)) \end{aligned}$$

$$\begin{aligned} \mathsf{minus}(x,0) &\to x \\ \mathsf{minus}(\mathsf{s}(x),\mathsf{s}(y)) &\to \mathsf{minus}(x,y) \\ \mathsf{div}(0,\mathsf{s}(y)) &\to 0 \\ \mathsf{div}(\mathsf{s}(x),\mathsf{s}(y)) &\to \mathsf{s}(\mathsf{div}(\mathsf{minus}(x,y),\mathsf{s}(y))) \end{aligned}$$

- **Definition**
  A sequence of dependency pairs $s_1 \to t_1,\ s_2 \to t_2,\ s_3 \to t_3, \ldots$
  is a *chain* iff there exists a substitution σ such that

$$t_1\sigma \to^* s_2\sigma, \quad t_2\sigma \to^* s_3\sigma, \quad \ldots$$

$\mathsf{D}(\mathsf{s}(x_1),\mathsf{s}(y_1)) \to \mathsf{D}(\mathsf{minus}(x_1,y_1),\mathsf{s}(y_1)),\ \mathsf{D}(\mathsf{s}(x_2),\mathsf{s}(y_2)) \to \mathsf{D}(\mathsf{minus}(x_2,y_2),\mathsf{s}(y_2))$

$\mathsf{D}(\mathsf{s}(\mathsf{s}(0)),\mathsf{s}(0)) \to \mathsf{D}(\mathsf{minus}(\mathsf{s}(0),0),\mathsf{s}(0)) \qquad \mathsf{D}(\mathsf{s}(0),\mathsf{s}(0)) \to \mathsf{D}(\mathsf{minus}(0,0),\mathsf{s}(0))$

with $\sigma = [x_1/\mathsf{s}(0),\ y_1/0,\ x_2/0,\ y_2/0]$

- **Theorem**
  A TRS terminates iff there is no infinite chain.

Dependency Pair Framework

- Apply the general idea of problem solving for termination analysis
  - transform problems into simpler sub-problems repeatedly until all problems are solved
- What *objects* do we work on, i.e., what are the "*problems*"?
  - ~~TRSs $\mathcal{R}$~~ not powerful enough
  - ~~DPs $\mathcal{P}$~~ not expressive enough
  - DP problems $(\mathcal{P}, \mathcal{R})$
- What *techniques* do we use for transformation?
  - DP processors: $Proc(\,(\mathcal{P}, \mathcal{R})\,) = \{(\mathcal{P}_1, \mathcal{R}_1), \ldots, (\mathcal{P}_n, \mathcal{R}_n)\}$
- When is a problem *solved*?
  - $(\mathcal{P}, \mathcal{R})$ is *finite* iff there is no infinite $(\mathcal{P}, \mathcal{R})$-chain

Dependency Pair Framework

- **Basic Idea**
  - examine *DP problems* $(\mathcal{P}, \mathcal{R})$
  - a DP problem $(\mathcal{P}, \mathcal{R})$ is *finite* iff
    there is no infinite $(\mathcal{P}, \mathcal{R})$-chain

- **Definition**
  A sequence of pairs $s_1 \to t_1,\ s_2 \to t_2,\ s_3 \to t_3, \ldots$ from $\mathcal{P}$
  is a $(\mathcal{P}, \mathcal{R})$-*chain* iff there exists a substitution σ such that

$$t_1\sigma \to_{\mathcal{R}}^* s_2\sigma, \quad t_2\sigma \to_{\mathcal{R}}^* s_3\sigma, \quad \ldots$$

- **Theorem**
  A TRS terminates iff there is no infinite $(DP(\mathcal{R}), \mathcal{R})$-chain.

Dependency Pair Framework

- **Procedure**

1. Start with the *initial* DP problem $(DP(\mathcal{R}), \mathcal{R})$.
2. Transform a remaining DP problem by a sound processor.
3. If result is **“no”** and all processors were complete, return **“no”**.
   If there is no remaining DP problem, then return **“yes”**.
   Otherwise go to 2.

DP processor: $Proc(\,(\mathcal{P}, \mathcal{R})\,) = \{(\mathcal{P}_1, \mathcal{R}_1), \ldots, (\mathcal{P}_n, \mathcal{R}_n)\}$ or **“no”**

$Proc$ is *sound:* if all $(\mathcal{P}_i, \mathcal{R}_i)$ are finite, then $(\mathcal{P}, \mathcal{R})$ is finite
$Proc$ is *complete:* if some $(\mathcal{P}_i, \mathcal{R}_i)$ is infinite or $Proc((\mathcal{P}, \mathcal{R}))=$**“no”**, then $(\mathcal{P}, \mathcal{R})$ is infinite

- **Theorem**
  A TRS terminates iff $(DP(\mathcal{R}), \mathcal{R})$ is finite.

Dependency Pair Framework

Procedure

1. Start with the *initial* DP problem $(DP(\mathcal{R}), \mathcal{R})$.
2. Transform a remaining DP problem by a sound processor.
3. If result is **"no"** and all processors were complete, return **"no"**.
   If there is no remaining DP problem, then return **"yes"**.
   Otherwise go to 2.

Remaining Lecture on Dependency Pairs

I. DP Processors for Proving Termination

II. DP Processors for Proving Innermost Termination

III. DP Processors from Other Termination Techniques

I. DP Processors for Proving Termination

- **Dependency Graph Processor**
- **Reduction Pair Processor**

  Processors only modify $\mathcal{P}$: $Proc(\,(\mathcal{P},\mathcal{R})\,) = \{(\mathcal{P}_1,\mathcal{R}),\ldots,(\mathcal{P}_n,\mathcal{R})\}$

- **Rule Removal Processor**

  Processor modifies $\mathcal{P}$ and $\mathcal{R}$: $Proc(\,(\mathcal{P},\mathcal{R})\,) = \{(\mathcal{P}_1,\mathcal{R}_1)\}$

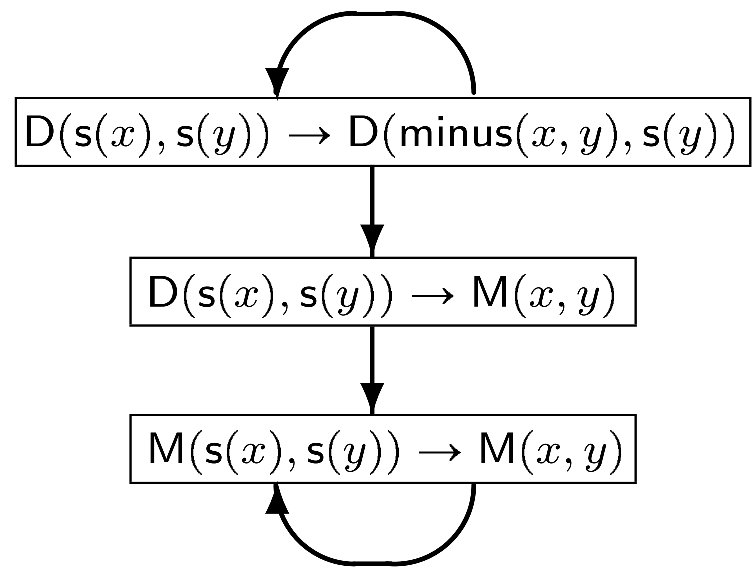
$\mathcal{P}$: $\mathsf{M}(\mathsf{s}(x),\mathsf{s}(y)) \to \mathsf{M}(x,y)$
$\mathsf{D}(\mathsf{s}(x),\mathsf{s}(y)) \to \mathsf{M}(x,y)$
$\mathsf{D}(\mathsf{s}(x),\mathsf{s}(y)) \to \mathsf{D}(\mathsf{minus}(x,y),\mathsf{s}(y))$

$\mathcal{R}$: $\mathsf{minus}(x,0) \to x$
$\mathsf{minus}(\mathsf{s}(x),\mathsf{s}(y)) \to \mathsf{minus}(x,y)$
$\mathsf{div}(0,\mathsf{s}(y)) \to 0$
$\mathsf{div}(\mathsf{s}(x),\mathsf{s}(y)) \to \mathsf{s}(\mathsf{div}(\mathsf{minus}(x,y),\mathsf{s}(y)))$

- **Dependency Graph Processor** (sound & complete)

  $$Proc(\,(\mathcal{P},\mathcal{R})\,) = \{\,(\mathcal{P}_1,\mathcal{R}),\ldots,(\mathcal{P}_n,\mathcal{R})\,\}$$

  where $\mathcal{P}_1,\ldots,\mathcal{P}_n$
  are the SCCs of the
  $(\mathcal{P},\mathcal{R})$-dependency graph



- $(\mathcal{P},\mathcal{R})$-Dependency Graph
  - directed graph whose nodes are the pairs of $\mathcal{P}$
  - arc from $s \to t$ to $v \to w$ iff $s \to t,\ v \to w$ is a $(\mathcal{P},\mathcal{R})$-chain

$\mathcal{P}_1 : \mathsf{M}(\mathsf{s}(x), \mathsf{s}(y)) \to \mathsf{M}(x, y)$

$\mathcal{P}_2 : \mathsf{D}(\mathsf{s}(x), \mathsf{s}(y)) \to \mathsf{D}(\mathsf{minus}(x, y), \mathsf{s}(y))$

$\mathcal{R}$:

$$\begin{aligned} \mathsf{minus}(x, 0) &\to x \\ \mathsf{minus}(\mathsf{s}(x), \mathsf{s}(y)) &\to \mathsf{minus}(x, y) \\ \mathsf{div}(0, \mathsf{s}(y)) &\to 0 \\ \mathsf{div}(\mathsf{s}(x), \mathsf{s}(y)) &\to \mathsf{s}(\mathsf{div}(\mathsf{minus}(x, y), \mathsf{s}(y))) \end{aligned}$$

- $(\gtrsim, \succ)$ is a *reduction pair* iff
  - $\succ$ is stable and well founded
  - $\gtrsim$ is stable and monotonic
  - $\succ$ and $\gtrsim$ are compatible ($\succ \circ \gtrsim \subseteq \succ$ or $\gtrsim \circ \succ \subseteq \succ$)

$$s_1 \to t_1 \, , \quad s_2 \to t_2 \, , \quad s_3 \to t_3 \, , \ldots$$

$$s_1\sigma \;{}_{(}\gtrsim_{)}\; t_1\sigma \gtrsim s_2\sigma \;{}_{(}\gtrsim_{)}\; t_2\sigma \gtrsim s_3\sigma \;{}_{(}\gtrsim_{)}\; t_3\sigma \gtrsim \ldots$$

$\mathcal{P}_1: \mathsf{M}(\mathsf{s}(x), \mathsf{s}(y)) \to \mathsf{M}(x, y)$

$\mathcal{P}_2: \mathsf{D}(\mathsf{s}(x), \mathsf{s}(y)) \to \mathsf{D}(\mathsf{minus}(x, y), \mathsf{s}(y))$

$$\mathcal{R}: \begin{aligned} \mathsf{minus}(x, 0) &\to x \\ \mathsf{minus}(\mathsf{s}(x), \mathsf{s}(y)) &\to \mathsf{minus}(x, y) \\ \mathsf{div}(0, \mathsf{s}(y)) &\to 0 \\ \mathsf{div}(\mathsf{s}(x), \mathsf{s}(y)) &\to \mathsf{s}(\mathsf{div}(\mathsf{minus}(x, y), \mathsf{s}(y))) \end{aligned}$$

- **Reduction Pair Processor** (sound & complete)

  $Proc(\,(\mathcal{P}, \mathcal{R})\,) = \{\, (\,\mathcal{P} \backslash \mathcal{P}_{\succ},\, \mathcal{R}\,)\,\}$ if

  - $l \gtrsim r$ for all rules $l \to r$ in $\mathcal{R}$ $\qquad \mathcal{R}_{\gtrsim} = \mathcal{R}$
  - $s \succ t$ or $s \gtrsim t$ for all $s \to t$ in $\mathcal{P}$ $\qquad \mathcal{P}_{\succ} \cup \mathcal{P}_{\gtrsim} = \mathcal{P}$

- Resulting Inequalities for $(\mathcal{P}_1, \mathcal{R})$:

$\mathcal{P}_1: \mathsf{M}(\mathsf{s}(x), \mathsf{s}(y)) \; {}_{(}\!\gtrsim\!{}_{)} \; \mathsf{M}(x, y)$

$$\mathcal{R}: \begin{aligned} \mathsf{minus}(x, 0) &\gtrsim x \\ \mathsf{minus}(\mathsf{s}(x), \mathsf{s}(y)) &\gtrsim \mathsf{minus}(x, y) \\ \mathsf{div}(0, \mathsf{s}(y)) &\gtrsim 0 \\ \mathsf{div}(\mathsf{s}(x), \mathsf{s}(y)) &\gtrsim \mathsf{s}(\mathsf{div}(\mathsf{minus}(x, y), \mathsf{s}(y))) \end{aligned}$$

$$\begin{aligned} \mathcal{P}ol(\mathsf{s}(t)) &= 1 + \mathcal{P}ol(t) \\ \mathcal{P}ol(f(t_1, t_2)) &= \mathcal{P}ol(t_1) \end{aligned}$$

$\mathcal{P}_1: \mathsf{M}(\mathsf{s}(x), \mathsf{s}(y)) \to \mathsf{M}(x, y)$

$\mathcal{P}_2: \mathsf{D}(\mathsf{s}(x), \mathsf{s}(y)) \to \mathsf{D}(\mathsf{minus}(x, y), \mathsf{s}(y))$

$\mathcal{R}:$

$$\mathsf{minus}(x, 0) \to x$$
$$\mathsf{minus}(\mathsf{s}(x), \mathsf{s}(y)) \to \mathsf{minus}(x, y)$$
$$\mathsf{div}(0, \mathsf{s}(y)) \to 0$$
$$\mathsf{div}(\mathsf{s}(x), \mathsf{s}(y)) \to \mathsf{s}(\mathsf{div}(\mathsf{minus}(x, y), \mathsf{s}(y)))$$

- **Reduction Pair Processor** (sound & complete)

  $Proc(\,(\mathcal{P}, \mathcal{R})\,) = \{\, (\,\mathcal{P} \backslash \mathcal{P}_{\succ},\, \mathcal{R}\,)\,\}$ if

  - $l \gtrsim r$ for all rules $l \to r$ in $\mathcal{R}$ — $\mathcal{R}_{\gtrsim} = \mathcal{R}$
  - $s \succ t$ or $s \gtrsim t$ for all $s \to t$ in $\mathcal{P}$ — $\mathcal{P}_{\succ} \cup \mathcal{P}_{\gtrsim} = \mathcal{P}$

- Resulting Inequalities for $(\mathcal{P}_2, \mathcal{R})$:

  $\mathcal{P}_2: \mathsf{D}(\mathsf{s}(x), \mathsf{s}(y)) \;_{(}\gtrsim_{)}\; \mathsf{D}(\mathsf{minus}(x, y), \mathsf{s}(y))$

$$\mathsf{minus}(0, \mathsf{s}(y)) \gtrsim 0$$
$$\mathsf{minus}(\mathsf{s}(x), \mathsf{s}(y)) \gtrsim \mathsf{minus}(x, y)$$
$$\mathsf{div}(0, \mathsf{s}(y)) \gtrsim 0$$
$$\mathsf{div}(\mathsf{s}(x), \mathsf{s}(y)) \gtrsim \mathsf{s}(\mathsf{div}(\mathsf{minus}(x, y), \mathsf{s}(y)))$$

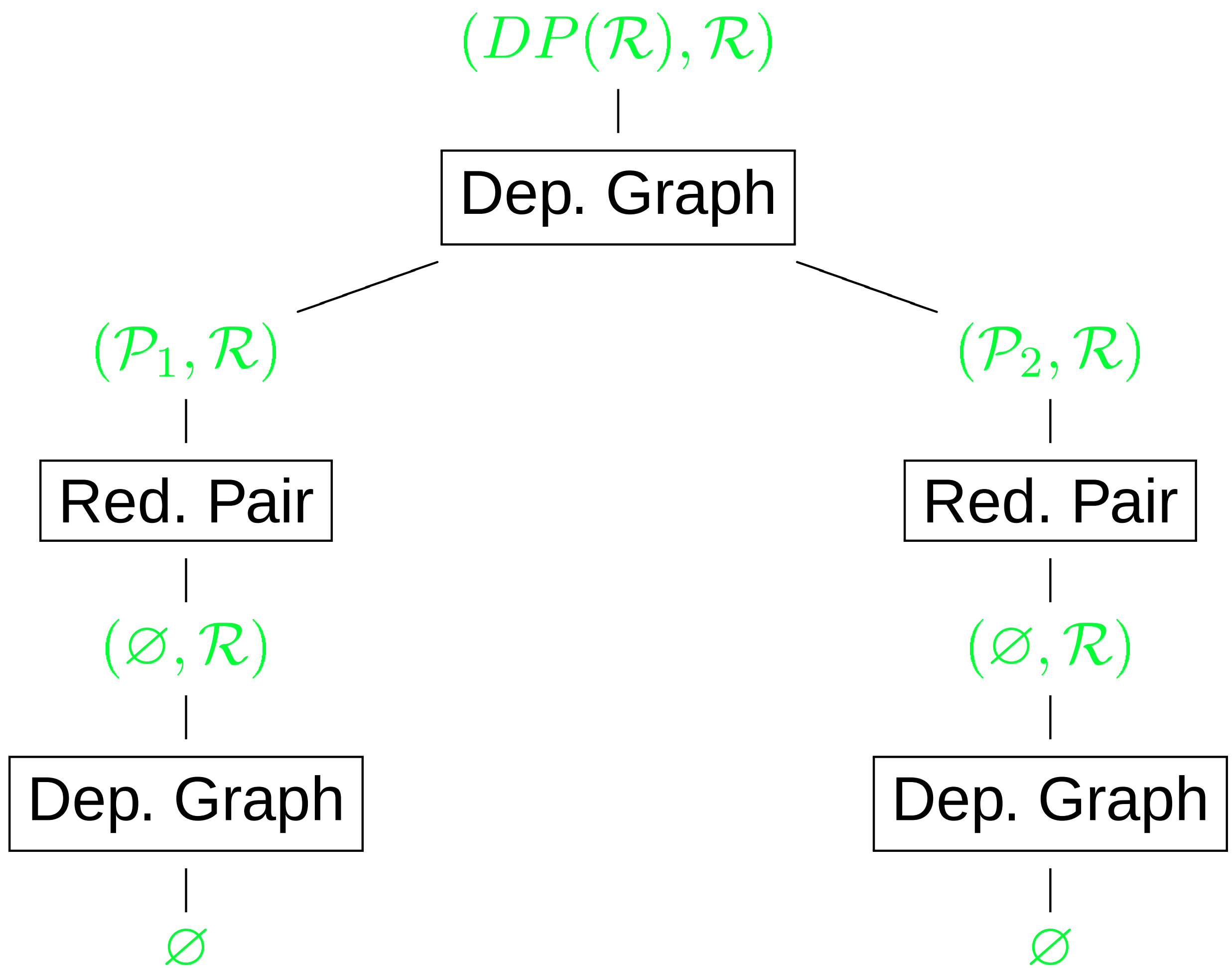
$$\mathcal{P}ol(\mathsf{s}(t)) = \mathcal{P}ol(t) + 1$$
$$\mathcal{P}ol(f(t_1, t_2)) = \mathcal{P}ol(t_1)$$

$\mathcal{P}_1 : \mathsf{M}(\mathsf{s}(x), \mathsf{s}(y)) \to \mathsf{M}(x, y)$

$\mathcal{P}_2 : \mathsf{D}(\mathsf{s}(x), \mathsf{s}(y)) \to \mathsf{D}(\mathsf{minus}(x, y), \mathsf{s}(y))$

$$\mathcal{R} : \begin{aligned} \mathsf{minus}(x, 0) &\to x \\ \mathsf{minus}(\mathsf{s}(x), \mathsf{s}(y)) &\to \mathsf{minus}(x, y) \\ \mathsf{div}(0, \mathsf{s}(y)) &\to 0 \\ \mathsf{div}(\mathsf{s}(x), \mathsf{s}(y)) &\to \mathsf{s}(\mathsf{div}(\mathsf{minus}(x, y), \mathsf{s}(y))) \end{aligned}$$

Termination is proved automatically!



$\mathcal{P}: \mathsf{M}(\mathsf{s}(x), \mathsf{s}(y)) \to \mathsf{M}(\mathsf{p}(\mathsf{s}(x)), \mathsf{p}(\mathsf{s}(y)))$

$$\mathcal{R}: \begin{aligned} \mathsf{p}(\mathsf{s}(x)) &\to x \\ \mathsf{minus}(x, 0) &\to x \\ \mathsf{minus}(\mathsf{s}(x), \mathsf{s}(y)) &\to \mathsf{minus}(\mathsf{p}(\mathsf{s}(x)), \mathsf{p}(\mathsf{s}(y))) \end{aligned}$$

DP problem for minus hard to solve automatically!

$\mathcal{P}:\ \mathsf{M}(\mathsf{s}(x), \mathsf{s}(y)) \to \mathsf{M}(\mathsf{p}(\mathsf{s}(x)), \mathsf{p}(\mathsf{s}(y)))$ $\quad \mathcal{R}:$

$$\begin{aligned} \mathsf{p}(\mathsf{s}(x)) &\to x \\ \mathsf{minus}(x, 0) &\to x \\ \mathsf{minus}(\mathsf{s}(x), \mathsf{s}(y)) &\to \mathsf{minus}(\mathsf{p}(\mathsf{s}(x)), \mathsf{p}(\mathsf{s}(y))) \end{aligned}$$

- **Rule Removal Processor** (sound & complete)

  $Proc(\,(\mathcal{P}, \mathcal{R})\,) = \{\,(\mathcal{P} \backslash \mathcal{P}_{\succ},\ \mathcal{R} \backslash \mathcal{R}_{\succ})\,\}$ if

  - $l \succ r$ or $l \succsim r$ for all $l \to r$ in $\mathcal{R}$ $\qquad \mathcal{R}_{\succ} \cup \mathcal{R}_{\succsim} = \mathcal{R}$
  - $s \succ t$ or $s \succsim t$ for all $s \to t$ in $\mathcal{P}$ $\qquad \mathcal{P}_{\succ} \cup \mathcal{P}_{\succsim} = \mathcal{P}$
  - $\succ$ is monotonic

- **Automation** $\qquad$ count number of s-symbols

  try $\quad \mathcal{P}ol(f(t_1, \ldots, t_n)) = \mathcal{P}ol(t_1) + \ldots + \mathcal{P}ol(t_n)$

  or $\quad \mathcal{P}ol(f(t_1, \ldots, t_n)) = 1 + \mathcal{P}ol(t_1) + \ldots + \mathcal{P}ol(t_n)$

$\mathcal{P}:\ \mathsf{M}(\mathsf{s}(x), \mathsf{s}(y)) \to \mathsf{M}(\mathsf{p}(\mathsf{s}(x)), \mathsf{p}(\mathsf{s}(y)))$ $\mathcal{R} \backslash \mathcal{R}_{\succ}:$

$$\mathsf{minus}(x, 0) \to x$$
$$\mathsf{minus}(\mathsf{s}(x), \mathsf{s}(y)) \to \mathsf{minus}(\mathsf{p}(\mathsf{s}(x)), \mathsf{p}(\mathsf{s}(y)))$$

- **Rule Removal Processor** (sound & complete)

  $Proc(\,(\mathcal{P}, \mathcal{R})\,) = \{\,(\mathcal{P} \backslash \mathcal{P}_{\succ}, \mathcal{R} \backslash \mathcal{R}_{\succ})\,\}$ if

  - $l \succ r$ or $l \succsim r$ for all $l \to r$ in $\mathcal{R}$ $\quad \mathcal{R}_{\succ} \cup \mathcal{R}_{\succsim} = \mathcal{R}$
  - $s \succ t$ or $s \succsim t$ for all $s \to t$ in $\mathcal{P}$ $\quad \mathcal{P}_{\succ} \cup \mathcal{P}_{\succsim} = \mathcal{P}$
  - $\succ$ is monotonic

- $(\mathcal{P}, \mathcal{R} \backslash \mathcal{R}_{\succ})$ is transformed into $\varnothing$ by the Dep. Graph Processor

  Termination is proved automatically!

II. DP Processors for Proving Innermost Termination

- **Component for the *evaluation strategy***

  $(\mathcal{P}, \mathcal{R}, e)$ with $e \in \{\mathbf{t}, \mathbf{i}\}$ for **t**ermination or **i**nnermost termination

$$\begin{array}{rcl} \mathsf{f}(\mathsf{g}(x), \mathsf{s}(0), y) & \rightarrow & \mathsf{f}(y, y, \mathsf{g}(x)) \\ \mathsf{g}(\mathsf{s}(x)) & \rightarrow & \mathsf{s}(\mathsf{g}(x)) \\ \mathsf{g}(0) & \rightarrow & 0 \end{array}$$

- Infinite (non-innermost) reduction:

$$\begin{array}{rl} \mathsf{f}(\mathrm{gs0}, \mathrm{s0}, \mathrm{gs0}) \rightarrow & \mathsf{f}(\mathrm{gs0}, \mathrm{gs0}, \mathrm{gs0}) \\ \rightarrow & \mathsf{f}(\mathrm{gs0}, \mathrm{sg0}, \mathrm{gs0}) \\ \rightarrow & \mathsf{f}(\mathrm{gs0}, \mathrm{s0}, \mathrm{gs0}) \rightarrow \ldots \end{array}$$

Dependency Pair Framework

- **Basic Idea**
  - examine *DP problems* $(\mathcal{P}, \mathcal{R}, e)$
  - a DP problem $(\mathcal{P}, \mathcal{R}, e)$ is *finite* iff there is no infinite $(\mathcal{P}, \mathcal{R}, e)$-chain
  - termination techniques should operate on DP problems:

    DP processor: $Proc(\,(\mathcal{P}, \mathcal{R}, e)\,) = \{(\mathcal{P}_1, \mathcal{R}_1, e_1), \ldots, (\mathcal{P}_n, \mathcal{R}_n, e_n)\}$

- **Theorem**
  A TRS $\mathcal{R}$ terminates (innermost) iff $(DP(\mathcal{R}), \mathcal{R}, e)$ is finite.

$\mathcal{P}:$

$$\mathsf{G}(\mathsf{s}(x)) \rightarrow \mathsf{G}(x)$$

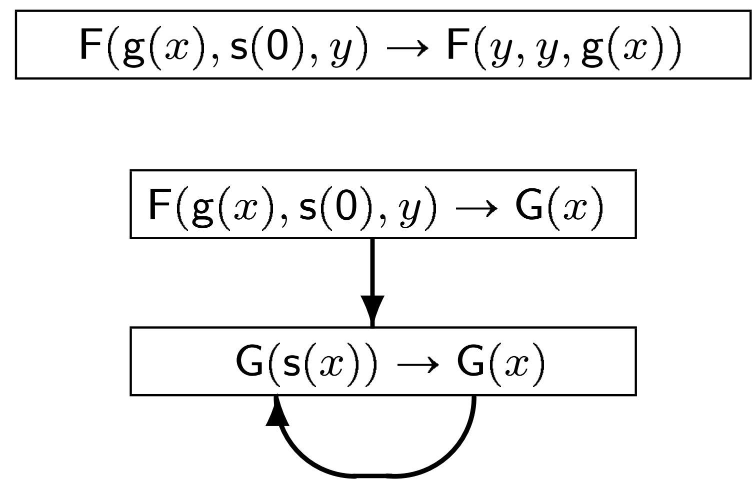
$\mathcal{R}:$

$$\begin{array}{rcl} \mathsf{f}(\mathsf{g}(x), \mathsf{s}(0), y) & \rightarrow & \mathsf{f}(y, y, \mathsf{g}(x)) \\ \mathsf{g}(\mathsf{s}(x)) & \rightarrow & \mathsf{s}(\mathsf{g}(x)) \\ \mathsf{g}(0) & \rightarrow & 0 \end{array}$$

- **Dependency Graph Processor** (sound & complete)

$$Proc(\,(\mathcal{P}, \mathcal{R}, e)\,) = \{\,(\mathcal{P}_1, \mathcal{R}, e), \ldots, (\mathcal{P}_n, \mathcal{R}, e)\,\}$$

where $\mathcal{P}_1, \ldots, \mathcal{P}_n$ are the SCCs of the $(\mathcal{P}, \mathcal{R}, e)$-dependency graph



- $(\mathcal{P}, \mathcal{R}, e)$-Dependency Graph
  - directed graph whose nodes are the pairs of $\mathcal{P}$
  - arc from $s \rightarrow t$ to $v \rightarrow w$ iff $s \rightarrow t,\ v \rightarrow w$ is a $(\mathcal{P}, \mathcal{R}, e)$-chain

$$\mathcal{P}: \quad \mathsf{G}(\mathsf{s}(x)) \to \mathsf{G}(x)$$

$$\mathcal{R}: \begin{array}{rcl} \mathsf{f}(\mathsf{g}(x), \mathsf{s}(0), y) & \to & \mathsf{f}(y, y, \mathsf{g}(x)) \\ \mathsf{g}(\mathsf{s}(x)) & \to & \mathsf{s}(\mathsf{g}(x)) \\ \mathsf{g}(0) & \to & 0 \end{array}$$

- **Usable Rule Processor** (sound)

  $$Proc(\,(\mathcal{P}, \mathcal{R}, \mathbf{i})\,) = \{\,(\,\mathcal{P},\ \mathcal{U}(\mathcal{P}, \mathcal{R}),\ \mathbf{i}\,)\,\}$$

- $\mathcal{U}(\mathcal{P}, \mathcal{R}) = \varnothing$

$\mathcal{P}:$ $\quad \mathsf{G}(\mathsf{s}(x)) \rightarrow \mathsf{G}(x)$ $\quad \mathcal{U}(\mathcal{P},\mathcal{R}):$

- **Usable Rule Processor** (sound)

  $Proc(\,(\mathcal{P},\mathcal{R},\mathbf{i})\,) = \{\,(\,\mathcal{P},\ \mathcal{U}(\mathcal{P},\mathcal{R}),\ \mathbf{i}\,)\,\}$

- example is trivial with Reduction Pair Processor: $\mathsf{G}(\mathsf{s}(x)) \succ \mathsf{G}(x)$

  Innermost termination is proved automatically!

- Completeness of processor can be achieved:

  sophisticated representation of the evaluation strategy, not just flag e

$$\mathcal{P}: \quad \mathsf{M}(\mathsf{s}(x), \mathsf{s}(y)) \to \mathsf{M}(x, y)$$
$$\mathsf{D}(\mathsf{s}(x), \mathsf{s}(y)) \to \mathsf{M}(x, y)$$
$$\mathsf{D}(\mathsf{s}(x), \mathsf{s}(y)) \to \mathsf{D}(\mathsf{minus}(x, y), \mathsf{s}(y))$$

$$\mathcal{R}: \quad \mathsf{minus}(x, 0) \to x$$
$$\mathsf{minus}(\mathsf{s}(x), \mathsf{s}(y)) \to \mathsf{minus}(x, y)$$
$$\mathsf{div}(0, \mathsf{s}(y)) \to 0$$
$$\mathsf{div}(\mathsf{s}(x), \mathsf{s}(y)) \to \mathsf{s}(\mathsf{div}(\mathsf{minus}(x, y), \mathsf{s}(y)))$$

- **Advantage of $e = \mathbf{i}$:**

  innermost termination is *easier* to prove than termination

  $\Rightarrow$ $Proc$ often more powerful if $e = \mathbf{i}$

  $\Rightarrow$ prove innermost termination instead of termination, if possible

- **Modular Non-Overlap Check Processor** (sound & complete)

  $Proc(\,(\mathcal{P}, \mathcal{R}, \mathbf{t})\,) = \{\,(\mathcal{P}, \mathcal{R}, \mathbf{i})\,\}$ if

  - $\mathcal{R}$ has no critical pairs with $\mathcal{P}$
  - $\mathcal{R}$ is locally confluent

$\mathcal{P}$: $\mathsf{M}(\mathsf{s}(x), \mathsf{s}(y)) \rightarrow \mathsf{M}(x, y)$
$\mathsf{D}(\mathsf{s}(x), \mathsf{s}(y)) \rightarrow \mathsf{M}(x, y)$
$\mathsf{D}(\mathsf{s}(x), \mathsf{s}(y)) \rightarrow \mathsf{D}(\mathsf{minus}(x, y), \mathsf{s}(y))$

$\mathcal{R}$: $\mathsf{minus}(x, 0) \rightarrow x$
$\mathsf{minus}(\mathsf{s}(x), \mathsf{s}(y)) \rightarrow \mathsf{minus}(x, y)$
$\mathsf{div}(0, \mathsf{s}(y)) \rightarrow 0$
$\mathsf{div}(\mathsf{s}(x), \mathsf{s}(y)) \rightarrow \mathsf{s}(\mathsf{div}(\mathsf{minus}(x, y), \mathsf{s}(y)))$

- **Example:**

  $(\mathcal{P}, \mathcal{R}, \mathbf{t})$ is replaced by $(\mathcal{P}, \mathcal{R}, \mathbf{i})$

  it suffices to prove innermost termination

- **Modular Non-Overlap Check Processor** (sound & complete)

  $Proc(\,(\mathcal{P}, \mathcal{R}, \mathbf{t})\,) = \{\,(\mathcal{P}, \mathcal{R}, \mathbf{i})\,\}$ if

  - $\mathcal{R}$ has no critical pairs with $\mathcal{P}$
  - $\mathcal{R}$ is locally confluent

III. DP Processors from Other Techniques

- **Termination Technique**

  TT maps TRSs to TRSs

  TT is is *sound:* if termination of $TT(\mathcal{R})$ implies termination of $\mathcal{R}$

  TT is *complete:* if termination of $\mathcal{R}$ implies termination of $TT(\mathcal{R})$

- Termination techniques can be transformational or *conventional*:

$$TT(\mathcal{R}) = \begin{cases} \varnothing, & \text{if termination of } \mathcal{R} \text{ can be proved} \\ \mathcal{R}, & \text{otherwise} \end{cases}$$

III. DP Processors from Other Techniques

- **Advantages**
  - different techniques can be used for different sub-problems
  - combines benefits of different methods and of dependency pair techniques
  - termination techniques with restricted applicability can be used, even if they are not applicable to the whole TRS

- **Termination Technique Processor** (sound & complete)

$$Proc(\,(\mathcal{P}, \mathcal{R}, e)\,) = \{\,(DP(\mathcal{R}'), \mathcal{R}', \mathbf{t})\,\}$$

where $\mathcal{R}' = TT(\mathcal{P} \cup \mathcal{R})$

III. DP Processors from Other Techniques

- **Example**: String Reversal
  - only applicable on *string rewrite systems (SRS)*
    arity$(f) = 1$ for all f
  - $TT(\mathcal{R}) = \mathcal{R}^{-1} \quad = \{l^{-1} \to r^{-1} \mid l \to r \in \mathcal{R}\}$ (sound & complete)
  - $\mathcal{R}$: $\mathsf{a}(\mathsf{b}(\mathsf{b}(x))) \to \mathsf{b}(\mathsf{a}(x))$
    $\mathcal{R}^{-1}$: $\mathsf{b}(\mathsf{b}(\mathsf{a}(x))) \to \mathsf{a}(\mathsf{b}(x))$

- **Termination Technique Processor** (sound & complete)

  $Proc(\,(\mathcal{P}, \mathcal{R}, e)\,) = \{\,(DP(\mathcal{R}'), \mathcal{R}', \mathbf{t})\,\}$

  where $\mathcal{R}' = TT(\mathcal{P} \cup \mathcal{R})$

Challenging Example

$$\begin{array}{rcl} \mathsf{p}(\mathsf{s}(0)) & \to & 0 \\ \mathsf{p}(\mathsf{s}(\mathsf{s}(x))) & \to & \mathsf{s}(\mathsf{p}(\mathsf{s}(x))) \\ \mathsf{fact}(0) & \to & \mathsf{s}(0) \\ \mathsf{fact}(\mathsf{s}(x)) & \to & \mathsf{times}(\mathsf{s}(x), \mathsf{fact}(\mathsf{p}(\mathsf{s}(x)))) \\ \mathsf{times}(\ldots) & \to & \ldots \\ \mathsf{plus}(\ldots) & \to & \ldots \end{array}$$

- Dependency Graph Processor: 4 DP problems for p, fact, times, plus
- DP problem for fact hard to solve automatically!
- String reversal not applicable: no SRS

$\mathcal{P}$: $\mathsf{FACT}(\mathsf{s}(x)) \rightarrow \mathsf{FACT}(\mathsf{p}(\mathsf{s}(x)))$

$\mathcal{R}$:

$$\begin{aligned} \mathsf{p}(\mathsf{s}(0)) &\rightarrow 0 \\ \mathsf{p}(\mathsf{s}(\mathsf{s}(x))) &\rightarrow \mathsf{s}(\mathsf{p}(\mathsf{s}(x))) \\ \mathsf{fact}(\ldots) &\rightarrow \ldots \\ \mathsf{times}(\ldots) &\rightarrow \ldots \\ \mathsf{plus}(\ldots) &\rightarrow \ldots \end{aligned}$$

- **Modular Non-Overlap Check Processor** (sound & complete)

  $Proc(\,(\mathcal{P},\mathcal{R},\mathbf{t})\,) = \{\,(\mathcal{P},\mathcal{R},\mathbf{i})\,\}$ if

  - $\mathcal{R}$ has no critical pairs with $\mathcal{P}$
  - $\mathcal{R}$ is locally confluent

- $(\mathcal{P},\mathcal{R},\mathbf{t})$ is replaced by $(\mathcal{P},\mathcal{R},\mathbf{i})$

  it suffices to prove innermost termination

$\mathcal{P}: \quad \mathsf{FACT}(\mathsf{s}(x)) \;\to\; \mathsf{FACT}(\mathsf{p}(\mathsf{s}(x)))$

$\mathcal{U}(\mathcal{P},\mathcal{R}):$

$$\begin{aligned} \mathsf{p}(\mathsf{s}(0)) &\to 0 \\ \mathsf{p}(\mathsf{s}(\mathsf{s}(x))) &\to \mathsf{s}(\mathsf{p}(\mathsf{s}(x))) \end{aligned}$$

- **Usable Rule Processor** (sound)

  $$Proc(\,(\mathcal{P},\mathcal{R},\mathbf{i})\,) = \{\,(\,\mathcal{P},\;\mathcal{U}(\mathcal{P},\mathcal{R}),\;\mathbf{i}\,)\,\}$$

- $\mathcal{U}(\mathcal{P},\mathcal{R}) \;=\; \{\; \mathsf{p}(\mathsf{s}(0)) \to 0, \quad \mathsf{p}(\mathsf{s}(\mathsf{s}(x))) \to \mathsf{s}(\mathsf{p}(\mathsf{s}(x)))\;\}$

$\mathcal{P}:\quad \mathsf{FACT}(\mathsf{s}(x)) \;\to\; \mathsf{FACT}(\mathsf{p}(\mathsf{s}(x)))$

$\mathcal{R}:\quad \mathsf{p}(\mathsf{s}(0)) \to 0$

$\mathsf{p}(\mathsf{s}(\mathsf{s}(x))) \to \mathsf{s}(\mathsf{p}(\mathsf{s}(x)))$

- **Rule Removal Processor** (sound & complete)

  $Proc(\,(\mathcal{P},\mathcal{R},e)\,) = \{\,(\,\mathcal{P}\backslash\mathcal{P}_{\succ},\ \mathcal{R}\backslash\mathcal{R}_{\succ},\ e\,)\,\}$ if

  - $l \succ r$ or $l \succsim r$ for all $l \to r$ in $\mathcal{R}$ — $\mathcal{R}_{\succ} \cup \mathcal{R}_{\succsim} = \mathcal{R}$
  - $s \succ t$ or $s \succsim t$ for all $s \to t$ in $\mathcal{P}$ — $\mathcal{P}_{\succ} \cup \mathcal{P}_{\succsim} = \mathcal{P}$
  - $\succ$ is monotonic

- **Automation** — count number of s-symbols

  try $\mathcal{P}ol(f(t_1,\ldots,t_n)) = \mathcal{P}ol(t_1) + \ldots + \mathcal{P}ol(t_n)$

  or $\mathcal{P}ol(f(t_1,\ldots,t_n)) = \mathcal{P}ol(t_1) + \ldots + \mathcal{P}ol(t_n) + 1$

$\mathcal{P}:\quad \mathsf{FACT}(\mathsf{s}(x)) \;\to\; \mathsf{FACT}(\mathsf{p}(\mathsf{s}(x)))$ $\qquad \mathcal{R}:\quad \mathsf{p}(\mathsf{s}(\mathsf{s}(x))) \;\to\; \mathsf{s}(\mathsf{p}(\mathsf{s}(x)))$

- **Termination Technique Processor** (sound & complete)

  $$Proc(\,(\mathcal{P},\mathcal{R},e)\,) = \{\,(DP(\mathcal{R}'),\mathcal{R}',\mathbf{t})\,\}$$

  where $\mathcal{R}' = \mathcal{P}^{-1} \cup \mathcal{R}^{-1}$

- String reversal applicable: arity$(f) = 1$ for all f

$DP(\mathcal{R}')$: $\mathsf{S}(\mathsf{FACT}(x)) \to \mathsf{S}(\mathsf{p}(\mathsf{FACT}(x)))$
$\mathsf{S}(\mathsf{s}(\mathsf{p}(x))) \to \mathsf{S}(\mathsf{p}(\mathsf{s}(x)))$
$\mathsf{S}(\mathsf{s}(\mathsf{p}(x))) \to \mathsf{S}(x)$

$\mathcal{R}'$: $\mathsf{s}(\mathsf{FACT}(x)) \to \mathsf{s}(\mathsf{p}(\mathsf{FACT}(x)))$
$\mathsf{s}(\mathsf{s}(\mathsf{p}(x))) \to \mathsf{s}(\mathsf{p}(\mathsf{s}(x)))$

- **Termination Technique Processor** (sound & complete)

$Proc(\,(\mathcal{P},\mathcal{R},e)\,) = \{\,(DP(\mathcal{R}'),\mathcal{R}',\mathbf{t})\,\}$

where $\mathcal{R}' = \mathcal{P}^{-1} \cup \mathcal{R}^{-1}$

$\mathcal{P}^{-1}$: $\mathsf{s}(\mathsf{FACT}(x)) \quad \to \quad \mathsf{s}(\mathsf{p}(\mathsf{FACT}(x)))$

$\mathcal{R}^{-1}$: $\mathsf{s}(\mathsf{s}(\mathsf{p}(x))) \quad \to \quad \mathsf{s}(\mathsf{p}(\mathsf{s}(x)))$

$\mathcal{P}$:

$$\mathsf{S}(\mathsf{s}(\mathsf{p}(x))) \to \mathsf{S}(x)$$

$\mathcal{R}$: $\mathsf{s}(\mathsf{FACT}(x)) \to \mathsf{s}(\mathsf{p}(\mathsf{FACT}(x)))$

$\mathsf{s}(\mathsf{s}(\mathsf{p}(x))) \to \mathsf{s}(\mathsf{p}(\mathsf{s}(x)))$

- **Dependency Graph Processor** (sound & complete)

$$Proc(\,(\mathcal{P}, \mathcal{R}, e)\,) = \{\,(\mathcal{P}_1, \mathcal{R}, e), \ldots, (\mathcal{P}_n, \mathcal{R}, e)\,\}$$

where $\mathcal{P}_1, \ldots, \mathcal{P}_n$ are the SCCs of the $(\mathcal{P}, \mathcal{R}, e)$-dependency graph

$(DP(\mathcal{R}), \mathcal{R})$

Dep. Graph

$\ldots$ $(\mathcal{P}, \mathcal{R}, \mathbf{t})$ $\ldots$ $\ldots$

Mod. NO Check

$(\mathcal{P}, \mathcal{R}, \mathbf{i})$

Inn. Usable Rules

$(\mathcal{P}, \mathcal{U}(\mathcal{P}, \mathcal{R}), \mathbf{i})$

Rule Removal

$(\mathcal{P}, \{\mathsf{p}(\mathsf{s}(\mathsf{s}(x))) \to \mathsf{s}(\mathsf{p}(\mathsf{s}(x)))\}, \mathbf{i})$

TT String Reversal

$(DP(\mathcal{R}'), \mathcal{R}', \mathbf{t})$

Dep. Graph

$(\{\mathsf{S}(\mathsf{s}(\mathsf{p}(x))) \to \mathsf{S}(x)\}, \mathcal{R}', \mathbf{t})$

Rule Removal

$(\varnothing, \mathcal{R}', \mathbf{t})$

Dep. Graph

$\varnothing$

Termination is proved automatically!

Semantic Labeling

$$\mathcal{R}: \qquad \mathsf{f}(\mathsf{f}(x)) \to \mathsf{f}(\mathsf{g}(\mathsf{f}(x)))$$

- **Choose model for TRS $\mathcal{R}$**
  - carrier set $M = \{0, 1\}$
  - for every n-ary symbol f choose interpretation $f_M : M^n \to M$

$$\mathsf{f}_M(x) = 0 \qquad \mathsf{g}_M(x) = x + 1$$

For every variable assignment $\alpha : \mathcal{V} \to M$

$$\alpha(\mathsf{f}(\mathsf{f}(x))) = 0 = \alpha(\mathsf{f}(\mathsf{g}(\mathsf{f}(x))))$$

$\Rightarrow$ is a model!

Semantic Labeling

$$\mathcal{R}: \quad \mathsf{f}(\mathsf{f}(x)) \rightarrow \mathsf{f}(\mathsf{g}(\mathsf{f}(x)))$$

- **Choose model for TRS $\mathcal{R}$**
  - carrier set $M = \{0, 1\}$
  - for every n-ary symbol f choose interpretation $f_M : M^n \rightarrow M$

$$\mathsf{f}_M(x) = 0 \qquad \mathsf{g}_M(x) = x + 1$$

- **Label every symbol by interpretation of its argument(s)**

$$\begin{array}{lcl} \mathsf{f}_0(\mathsf{f}_0(x)) & \rightarrow & \mathsf{f}_1(\mathsf{g}_0(\mathsf{f}_0(x))) \\ \mathsf{f}_0(\mathsf{f}_1(x)) & \rightarrow & \mathsf{f}_1(\mathsf{g}_0(\mathsf{f}_1(x))) \end{array}$$

- **$\mathcal{R}$ terminates iff labeled TRS $\overline{\mathcal{R}}$ terminates**

termination can be proved by LPO if f_0 has highest precedence

Semantic Labeling

$$\begin{aligned}
\mathsf{p}\ (\mathsf{s}\ (0)) &\to 0 \\
\mathsf{p}\quad (\mathsf{s}\quad (\mathsf{s}\ (x))) &\to \mathsf{s}\ (\mathsf{p}\quad (\mathsf{s}\ (x))) \\
\mathsf{fact}\ (0) &\to \mathsf{s}\ (0) \\
\mathsf{fact}\quad (\mathsf{s}\ (x)) &\to \mathsf{times}\qquad (\mathsf{s}\ (x), \mathsf{fact}\ (\mathsf{p}\quad (\mathsf{s}\ (x))))
\end{aligned}$$

- **Choose model for TRS $\mathcal{R}$**
  - carrier set $M = \mathbb{N}$
  - for every n-ary symbol f choose interpretation $f_M : M^n \to M$

Semantic Labeling

$$\begin{aligned}
\mathsf{p}_1(\mathsf{s}_0(0)) &\to 0 \\
\mathsf{p}_{n+2}(\mathsf{s}_{n+1}(\mathsf{s}_n(x))) &\to \mathsf{s}_n(\mathsf{p}_{n+1}(\mathsf{s}_n(x))) \\
\mathsf{fact}_0(0) &\to \mathsf{s}_0(0) \\
\mathsf{fact}_{n+1}(\mathsf{s}_n(x)) &\to \mathsf{times}_{(n+1,n+1)}(\mathsf{s}_n(x), \mathsf{fact}_n(\mathsf{p}_{n+1}(\mathsf{s}_n(x))))
\end{aligned}$$

- **Choose model for TRS $\mathcal{R}$**

$$\begin{aligned}
0_M(x) &= 0 \\
\mathsf{s}_M(x) &= x+1 \\
\mathsf{p}_M(x) &= \begin{cases} 0, & \text{if } x = 0 \\ x-1, & \text{if } x > 0 \end{cases} \\
\mathsf{fact}_M(x) &= x+1 \\
\mathsf{times}_M(x,y) &= y+1
\end{aligned}$$

- **Label every symbol by interpretation of its argument(s)**
- **$\mathcal{R}$ terminates iff labeled TRS $\overline{\mathcal{R}}$ terminates**

  termination can be proved by LPO if $\mathsf{fact}_{n+1} > \mathsf{fact}_n$

Match–Bounds for String Rewriting

$$\mathcal{R}: \quad \mathsf{a}(\mathsf{b}(\mathsf{a}(x))) \to \mathsf{a}(\mathsf{b}(\mathsf{b}(\mathsf{b}(\mathsf{a}(x)))))$$

- **match($\mathcal{R}$)**
  - label symbols of lhs by arbitrary natural numbers
  - label all symbols of rhs by *1 + minimum label of lhs*

$$\mathsf{a}_0(\mathsf{b}_0(\mathsf{a}_0(x))) \to \mathsf{a}_1(\mathsf{b}_1(\mathsf{b}_1(\mathsf{b}_1(\mathsf{a}_1(x)))))$$
$$\mathsf{a}_0(\mathsf{b}_1(\mathsf{a}_5(x))) \to \mathsf{a}_1(\mathsf{b}_1(\mathsf{b}_1(\mathsf{b}_1(\mathsf{a}_1(x)))))$$
$$\mathsf{a}_3(\mathsf{b}_1(\mathsf{a}_5(x))) \to \mathsf{a}_2(\mathsf{b}_2(\mathsf{b}_2(\mathsf{b}_2(\mathsf{a}_2(x)))))$$
$$\vdots$$

Match–Bounds for String Rewriting

$$\mathcal{R}: \quad \mathsf{a}(\mathsf{b}(\mathsf{a}(x))) \rightarrow \mathsf{a}(\mathsf{b}(\mathsf{b}(\mathsf{b}(\mathsf{a}(x)))))$$

- **Thm 1:** If all symbols in t are labeled with 0 and for all $t \rightarrow^*_{\mathbf{match}(\mathcal{R})} s$, the labels in s are $\leqslant n$ (*match-bound*), then t is terminating w.r.t. $\mathcal{R}$.

- **Right-Forward Closures** (Dershowitz, 81):

$$\mathcal{R}_\# = \mathcal{R} \cup \{l_1\, \# \rightarrow r \mid l \rightarrow r \in \mathcal{R},\ l = l_1\, l_2,\ l_i \neq \varepsilon\}$$

$$\mathcal{R}_\#: \quad \begin{aligned} \mathsf{a}(\mathsf{b}(\mathsf{a}(x))) &\rightarrow \mathsf{a}(\mathsf{b}(\mathsf{b}(\mathsf{b}(\mathsf{a}(x))))) \\ \mathsf{a}(\mathsf{b}(\#(x))) &\rightarrow \mathsf{a}(\mathsf{b}(\mathsf{b}(\mathsf{b}(\mathsf{a}(x))))) \\ \mathsf{a}(\#(x)) &\rightarrow \mathsf{a}(\mathsf{b}(\mathsf{b}(\mathsf{b}(\mathsf{a}(x))))) \end{aligned}$$

Match–Bounds for String Rewriting

$$\mathcal{R}: \quad a(b(a(x))) \rightarrow a(b(b(b(a(x)))))$$

- **Thm 1:** If all symbols in t are labeled with 0 and for all $t \rightarrow^*_{\mathbf{match}(\mathcal{R})} s$, the labels in s are $\leqslant n$ (*match-bound*), then t is terminating w.r.t. $\mathcal{R}$.

- **Thm 2:** If $r\ \#^k$ terminates w.r.t. $\mathcal{R}_\#$ for all rhs's r and all $k \in \mathbb{N}$, then $\mathcal{R}$ is terminating.

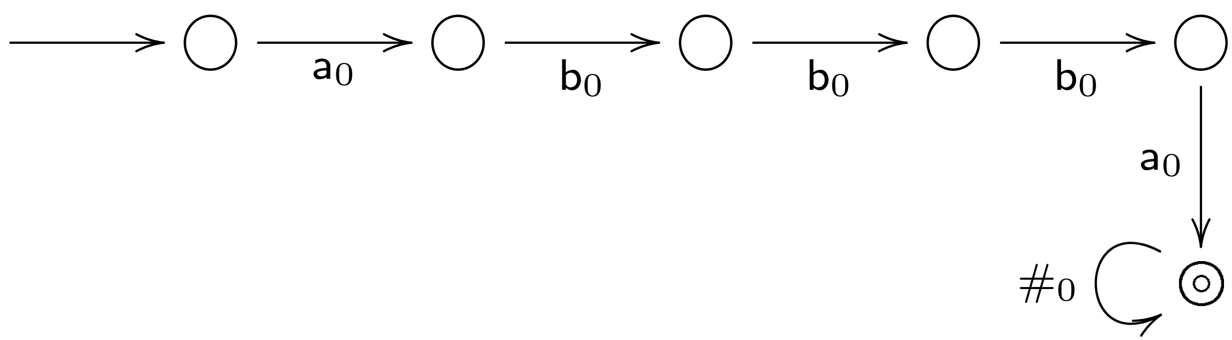
- **Thm 3:** If all symbols in $r\ \#^k$ are labeled with 0 and for all $r\ \#^k \rightarrow^*_{\mathbf{match}(\mathcal{R}_\#)} s$, labels in s are $\leqslant n$ (*match-bound*), then $\mathcal{R}$ is terminating.

Construct finite automaton accepting $\{s \mid r\ \#^k \rightarrow^*_{\mathbf{match}(\mathcal{R}_\#)} s\}$

$$\mathcal{R}_\#: \quad \begin{aligned} a(b(a(x))) &\rightarrow a(b(b(b(a(x))))) \\ a(b(\#(x))) &\rightarrow a(b(b(b(a(x))))) \\ a(\#(x)) &\rightarrow a(b(b(b(a(x))))) \end{aligned}$$

Match–Bounds for String Rewriting

- If there is path from q_1 to q_2 with lhs of $\mathbf{match}(\mathcal{R}_\#)$, then check if there is path from q_1 to q_2 with corresponding rhs.
  - If rhs = $a\ w$ and there is path from q_1' to q_2 with w, then add edge from q_1 to q_1' with a.
  - Otherwise, add new path from q_1 to q_2 with $a\ w$.

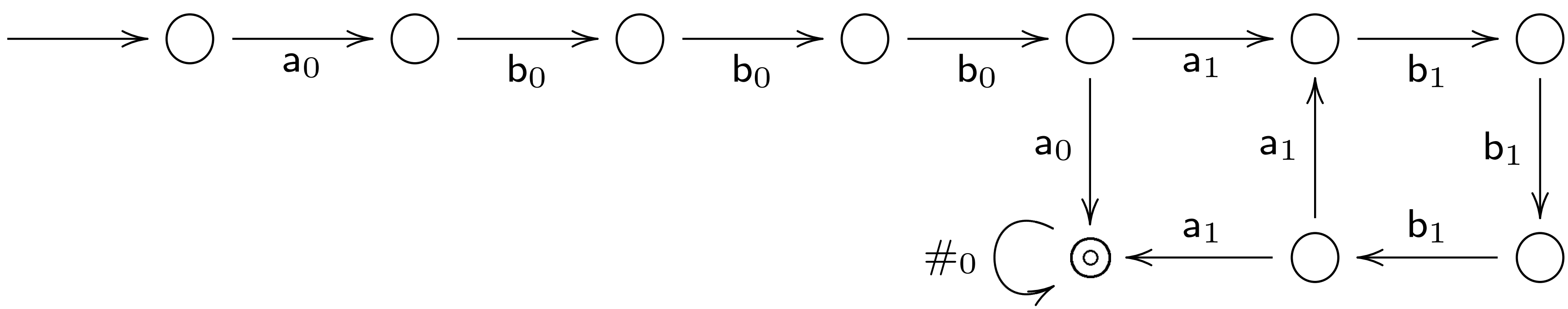


Construct finite automaton accepting $\{s \mid r\ \#^k \to^*_{\mathbf{match}(\mathcal{R}_\#)} s\}$

$$\mathcal{R}_\# : \quad \begin{aligned} \mathsf{a}(\mathsf{b}(\mathsf{a}(x))) &\to \mathsf{a}(\mathsf{b}(\mathsf{b}(\mathsf{b}(\mathsf{a}(x))))) \\ \mathsf{a}(\mathsf{b}(\#(x))) &\to \mathsf{a}(\mathsf{b}(\mathsf{b}(\mathsf{b}(\mathsf{a}(x))))) \\ \mathsf{a}(\#(x)) &\to \mathsf{a}(\mathsf{b}(\mathsf{b}(\mathsf{b}(\mathsf{a}(x))))) \end{aligned}$$

Match–Bounds for String Rewriting

- If there is path from q_1 to q_2 with lhs of $\mathbf{match}(\mathcal{R}_\#)$, then check if there is path from q_1 to q_2 with corresponding rhs.
  - If rhs = $a\ w$ and there is path from q_1' to q_2 with w, then add edge from q_1 to q_1' with a.
  - Otherwise, add new path from q_1 to q_2 with $a\ w$.



Construct finite automaton accepting $\{s \mid r\ \#^k \to^*_{\mathbf{match}(\mathcal{R}_\#)} s\}$

$$
\mathcal{R}_\# : \quad
\begin{aligned}
\mathsf{a}(\mathsf{b}(\mathsf{a}(x))) &\to \mathsf{a}(\mathsf{b}(\mathsf{b}(\mathsf{b}(\mathsf{a}(x)))))\\
\mathsf{a}(\mathsf{b}(\#(x))) &\to \mathsf{a}(\mathsf{b}(\mathsf{b}(\mathsf{b}(\mathsf{a}(x)))))\\
\mathsf{a}(\#(x)) &\to \mathsf{a}(\mathsf{b}(\mathsf{b}(\mathsf{b}(\mathsf{a}(x)))))
\end{aligned}
$$

Implementation

- **Procedure**

1. Start with the *initial* DP problem $(DP(\mathcal{R}), \mathcal{R})$.
2. Transform a remaining DP problem by a sound processor.
3. If result is **"no"** and all processors were complete, return **"no"**. If there is no remaining DP problem, then return **"yes"**. Otherwise go to 2.

- **Strategy**

  - decide which DP processor to use in Step 2
  - use fast processors first
  - only use slower more powerful processors on DP problems that cannot be solved by fast processors

Strategy of AProVE

Use the first applicable processor from the following list:

1. Dependency Graph Processor
2. Modular Non-Overlap Check Processor
3. Usable Rule Processor
4. A–Transformation Processor
5. Size–Change Processor
6. DP Transformation Processors (in “safe” cases)
7. Rule Removal Processor
8. Reduction Pair Processor: linear polynomials over $\{0, 1\}$
9. Termination Technique Processor: Match–Bounds
10. Reduction Pair Processor: LPO with strict precedence
11. DP Transformation Processors (up to a certain limit)
12. Reduction Pair Processor: linear polynomials over $\{-1, 0, 1\}$
13. Non-Termination Processor
14. Reduction Pair Processor: non-linear polynomials over $\{0, 1\}$
15. Reduction Pair Processor: LPO with non-strict precedence
16. Termination Technique Processor: String Reversal
17. Forward–Instantiation Processor
18. Termination Technique Processor: Semantic Labeling

Termination of Term Rewriting

- DP Framework combines many different termination techniques
  - termination techniques also help for disproving termination
  - non-termination techniques also help for proving termination
  - also possible for innermost termination

- Improvements
  - development of new DP processors
  - development of strategies how to apply DP processors
  - more efficient algorithms (e.g., using SAT-solvers)

- **AProVE**: implements the DP framework
  - winner of *Internat. Termination Competition '04 - '12*
    both for *termination* and *non-termination* of TRSs

`http://aprove.informatik.rwth-aachen.de/`

Overview

I. Termination of Term Rewriting

1. Termination of Term Rewrite Systems
2. Non-Termination of Term Rewrite Systems (FroCoS '05, IJCAR '12)
3. Complexity of Term Rewrite Systems
4. Termination of Integer Term Rewrite Systems

II. Termination of Programs

1. Termination of Functional Programs (Haskell)
2. Termination of Logic Programs (Prolog)
3. Termination of Imperative Programs (Java)

DP Processors for Disproving Termination

$$\begin{array}{rcl} \mathsf{minus}(x, \mathsf{0}) & \to & x \\ \mathsf{minus}(\mathsf{s}(x), \mathsf{s}(y)) & \to & \mathsf{minus}(x, y) \\ \mathsf{div}(\mathsf{0}, y) & \to & \mathsf{0} \\ \mathsf{div}(\mathsf{s}(x), y) & \to & \mathsf{s}(\mathsf{div}(\mathsf{minus}(\mathsf{s}(x), y), y)) \end{array}$$

$\mathcal{R}$ is *looping*:

$$\mathsf{div}(\mathsf{s}(x), \mathsf{0}) \to_{\mathcal{R}} \mathsf{s}(\mathsf{div}(\mathsf{minus}(\mathsf{s}(x), \mathsf{0}), \mathsf{0})) \to_{\mathcal{R}} \mathsf{s}(\mathsf{div}(\mathsf{s}(x), \mathsf{0})) \to_{\mathcal{R}} \ldots$$

Definition (Looping TRS)

A TRS $\mathcal{R}$ is *looping* if $s \to_{\mathcal{R}}^{+} C[s\mu]$ for some term s

$$\mathcal{P}: \quad \mathsf{M}(\mathsf{s}(x),\mathsf{s}(y)) \to \mathsf{M}(x,y)$$
$$\mathsf{D}(\mathsf{s}(x),y) \to \mathsf{M}(\mathsf{s}(x),y)$$
$$\mathsf{D}(\mathsf{s}(x),y) \to \mathsf{D}(\mathsf{minus}(\mathsf{s}(x),y),y)$$

$$\mathcal{R}: \quad \mathsf{minus}(x,\mathsf{0}) \to x$$
$$\mathsf{minus}(\mathsf{s}(x),\mathsf{s}(y)) \to \mathsf{minus}(x,y)$$
$$\mathsf{div}(\mathsf{0},y) \to \mathsf{0}$$
$$\mathsf{div}(\mathsf{s}(x),y) \to \mathsf{s}(\mathsf{div}(\mathsf{minus}(\mathsf{s}(x),y),y))$$

Definition (Looping DP Problem)

$(\mathcal{P},\mathcal{R})$ is *looping* if there is a chain $s_1 \to t_1,\ s_2 \to t_2, \ldots, s_k \to t_k$ with

- $t_1\sigma \to_{\mathcal{R}}^* s_2\sigma, \quad t_2\sigma \to_{\mathcal{R}}^* s_3\sigma, \quad \ldots$
- $s_1\sigma$ matches $s_k\sigma, \quad k > 1$ $\qquad (s_1\sigma\,\mu = s_k\sigma)$

Theorem

A TRS $\mathcal{R}$ is *looping* iff $(DP(\mathcal{R}),\mathcal{R})$ is *looping*

Definition (Looping TRS)

A TRS $\mathcal{R}$ is *looping* if $s \to_{\mathcal{R}}^+ C[s\mu]$ for some term s

$$\mathcal{P}: \begin{array}{rcl} \mathsf{M}(\mathsf{s}(x),\mathsf{s}(y)) & \to & \mathsf{M}(x,y) \\ \mathsf{D}(\mathsf{s}(x),y) & \to & \mathsf{M}(\mathsf{s}(x),y) \\ \mathsf{D}(\mathsf{s}(x),y) & \to & \mathsf{D}(\mathsf{minus}(\mathsf{s}(x),y),y) \end{array}$$

$$\mathcal{R}: \begin{array}{rcl} \mathsf{minus}(x,0) & \to & x \\ \mathsf{minus}(\mathsf{s}(x),\mathsf{s}(y)) & \to & \mathsf{minus}(x,y) \\ \mathsf{div}(0,y) & \to & 0 \\ \mathsf{div}(\mathsf{s}(x),y) & \to & \mathsf{s}(\mathsf{div}(\mathsf{minus}(\mathsf{s}(x),y),y)) \end{array}$$

Definition (Looping DP Problem)

$(\mathcal{P},\mathcal{R})$ is *looping* if there is a chain $s_1 \to t_1,\ s_2 \to t_2, \ldots, s_k \to t_k$ with

- $t_1\sigma \to_{\mathcal{R}}^* s_2\sigma, \quad t_2\sigma \to_{\mathcal{R}}^* s_3\sigma, \quad \ldots$
- $s_1\sigma$ matches $s_k\sigma, \quad k > 1$ $\qquad (s_1\sigma\,\mu = s_k\sigma)$

Example

$\mathsf{D}(\mathsf{s}(x_1),\mathsf{s}(y_1)) \to \mathsf{D}(\mathsf{minus}(\mathsf{s}(x_1),y_1),\mathsf{s}(y_1)),\ \mathsf{D}(\mathsf{s}(x_2),\mathsf{s}(y_2)) \to \ldots$

$\mathsf{D}(\mathsf{s}(x_1),\mathsf{s}(0)) \to \mathsf{D}(\mathsf{minus}(\mathsf{s}(x_1),0),\mathsf{s}(0)) \qquad \mathsf{D}(\mathsf{s}(x_1),\mathsf{s}(0)) \to \ldots$

loops with $\sigma = [y_1/0,\ x_2/x_1,\ y_2/0]$, $\mu =$identity

$$\mathcal{P}: \quad \mathsf{M}(\mathsf{s}(x), \mathsf{s}(y)) \to \mathsf{M}(x, y)$$
$$\mathsf{D}(\mathsf{s}(x), y) \to \mathsf{M}(\mathsf{s}(x), y)$$
$$\mathsf{D}(\mathsf{s}(x), y) \to \mathsf{D}(\mathsf{minus}(\mathsf{s}(x), y), y)$$

$$\mathcal{R}: \quad \mathsf{minus}(x, 0) \to x$$
$$\mathsf{minus}(\mathsf{s}(x), \mathsf{s}(y)) \to \mathsf{minus}(x, y)$$
$$\mathsf{div}(0, y) \to 0$$
$$\mathsf{div}(\mathsf{s}(x), y) \to \mathsf{s}(\mathsf{div}(\mathsf{minus}(\mathsf{s}(x), y), y))$$

Definition (Looping DP Problem)

$(\mathcal{P}, \mathcal{R})$ is *looping* if there is a chain $s_1 \to t_1, \; s_2 \to t_2, \ldots, s_k \to t_k$ with

- $t_1\sigma \to_{\mathcal{R}}^* s_2\sigma, \quad t_2\sigma \to_{\mathcal{R}}^* s_3\sigma, \quad \ldots$
- $s_1\sigma$ matches $s_k\sigma, \quad k > 1$ $\qquad (s_1\sigma\,\mu = s_k\sigma)$

Loopingness $\Rightarrow$ Infiniteness

Non-Termination Processor (sound & complete)

$Proc(\,(\mathcal{P}, \mathcal{R})\,) =$ **"no"** $\qquad$ if $(\mathcal{P}, \mathcal{R})$ is looping

$$\mathcal{P}: \quad \mathsf{M}(\mathsf{s}(x),\mathsf{s}(y)) \to \mathsf{M}(x,y)$$
$$\mathsf{D}(\mathsf{s}(x),y) \to \mathsf{M}(\mathsf{s}(x),y)$$
$$\mathsf{D}(\mathsf{s}(x),y) \to \mathsf{D}(\mathsf{minus}(\mathsf{s}(x),y),y)$$

$$\mathcal{R}: \quad \mathsf{minus}(x,\mathsf{0}) \to x$$
$$\mathsf{minus}(\mathsf{s}(x),\mathsf{s}(y)) \to \mathsf{minus}(x,y)$$
$$\mathsf{div}(\mathsf{0},y) \to \mathsf{0}$$
$$\mathsf{div}(\mathsf{s}(x),y) \to \mathsf{s}(\mathsf{div}(\mathsf{minus}(\mathsf{s}(x),y),y))$$

Definition (Looping DP Problem)

$(\mathcal{P},\mathcal{R})$ is *looping* if there is a chain $s_1 \to t_1,\; s_2 \to t_2, \ldots, s_k \to t_k$ with

- $t_1\sigma \to_{\mathcal{R}}^* s_2\sigma, \quad t_2\sigma \to_{\mathcal{R}}^* s_3\sigma, \quad \ldots$
- $s_1\sigma$ matches $s_k\sigma, \quad k > 1$ $\qquad (s_1\sigma\,\mu = s_k\sigma)$

Advantages of Loop Detection in DP Framework

- no need to search for context $C[\ldots]$
- other processors remove terminating parts:
  Termination techniques help for disproving termination

$$\mathcal{P}: \quad \mathsf{M}(\mathsf{s}(x), \mathsf{s}(y)) \to \mathsf{M}(x, y)$$
$$\mathsf{D}(\mathsf{s}(x), y) \to \mathsf{M}(\mathsf{s}(x), y)$$
$$\mathsf{D}(\mathsf{s}(x), y) \to \mathsf{D}(\mathsf{minus}(\mathsf{s}(x), y), y)$$

$$\mathcal{R}: \quad \mathsf{minus}(x, 0) \to x$$
$$\mathsf{minus}(\mathsf{s}(x), \mathsf{s}(y)) \to \mathsf{minus}(x, y)$$
$$\mathsf{div}(0, y) \to 0$$
$$\mathsf{div}(\mathsf{s}(x), y) \to \mathsf{s}(\mathsf{div}(\mathsf{minus}(\mathsf{s}(x), y), y))$$

Dependency Graph Processor results in

- $\mathcal{P}_1: \quad \mathsf{M}(\mathsf{s}(x), \mathsf{s}(y)) \to \mathsf{M}(x, y) \qquad \mathcal{R}: \quad \ldots$

  easy to prove

- $\mathcal{P}_2: \quad \mathsf{D}(\mathsf{s}(x), y) \to \mathsf{D}(\mathsf{minus}(\mathsf{s}(x), y), y) \qquad \mathcal{R}: \quad \ldots$

  potentially infinite

Termination techniques help to identify non-terminating parts!

$\mathcal{P}:\ \mathsf{D}(\mathsf{s}(x), y) \to \mathsf{D}(\mathsf{minus}(\mathsf{s}(x), y), y)$

$\mathcal{R}:$

$$\begin{aligned} \mathsf{minus}(x, 0) &\to x \\ \mathsf{minus}(\mathsf{s}(x), \mathsf{s}(y)) &\to \mathsf{minus}(x, y) \\ \mathsf{div}(0, y) &\to 0 \\ \mathsf{div}(\mathsf{s}(x), y) &\to \mathsf{s}(\mathsf{div}(\mathsf{minus}(\mathsf{s}(x), y), y)) \end{aligned}$$

Detect loops by narrowing:

- start with rhs t of a dependency pair $s \to t$
- narrow t repeatedly: $s \to t \quad \rightsquigarrow^*_\sigma \quad s'$
- check whether $s\sigma$ matches s'

$$\mathsf{D}(\mathsf{s}(x), y) \to \mathsf{D}(\mathsf{minus}(\mathsf{s}(x), y), y) \quad \rightsquigarrow_{\sigma=[y/0]} \quad \mathsf{D}(\mathsf{s}(x), 0)$$

$$\underbrace{\mathsf{D}(\mathsf{s}(x), y)\ \sigma}_{\mathsf{D}(\mathsf{s}(x),0)} \quad \text{matches} \quad \mathsf{D}(\mathsf{s}(x), 0)$$

$\mathcal{P}:\quad \mathsf{F}(x,y) \to \mathsf{G}(x,y)$
$\mathsf{G}(\mathsf{s}(x),y) \to \mathsf{F}(y,y)$

$\mathcal{R}:\quad \mathsf{f}(x,y) \to \mathsf{g}(x,y)$
$\mathsf{g}(\mathsf{s}(x),y) \to \mathsf{f}(y,y)$

Detect loops by narrowing:

- start with rhs t of a dependency pair $s \to t$
- narrow t repeatedly: $s \to t \quad \rightsquigarrow^*_\sigma \quad s'$
- check whether $s\sigma$ and s' semi-unify

$\mathsf{F}(x,y) \to \mathsf{G}(x,y) \quad \rightsquigarrow_{\sigma=[x/\mathsf{s}(x)]} \quad \mathsf{F}(y,y)$

$\underbrace{\mathsf{F}(x,y)\ \sigma}_{\mathsf{F}(\mathsf{s}(x),y)}$ unifies with $\mathsf{F}(y,y)$

$\mathcal{P}:\ \mathsf{F}(0,1,x) \to \mathsf{F}(x,x,x)$

$\mathcal{R}:\ \mathsf{f}(0,1,x) \to \mathsf{f}(x,x,x)$
$\mathsf{g}(y,z) \to y$
$\mathsf{g}(y,z) \to z$

Detect loops by narrowing:

- start with rhs t of a dependency pair $s \to t$
- narrow t repeatedly: $s \to t \quad \rightsquigarrow_{\sigma}^{*} \quad s'$
- check whether $s\sigma$ and s' semi-unify

$\mathsf{F}(0,1,x) \to \mathsf{F}(x,x,x)$ cannot be narrowed!

$\mathcal{P}:\ \mathsf{F}(0,1,x) \to \mathsf{F}(x,x,x)$

$$\mathcal{R}:\ \begin{aligned} \mathsf{f}(0,1,x) &\to \mathsf{f}(x,x,x) \\ \mathsf{g}(y,z) &\to y \\ \mathsf{g}(y,z) &\to z \end{aligned}$$

$$\mathsf{F}(\mathsf{g}(0,z),\mathsf{g}(y,1),x) \quad \leftsquigarrow \quad \mathsf{F}(\mathsf{g}(0,z),\underline{1},x) \quad \leftsquigarrow \quad \mathsf{F}(\underline{0},1,x) \to \mathsf{F}(x,x,x)$$

$\mathsf{F}(\mathsf{g}(0,z),\mathsf{g}(y,1),x)$ unifies with $\mathsf{F}(x,x,x)$

Detect loops by backward narrowing:

- start with lhs s of a dependency pair $s \to t$
- narrow s with reversed rules: $t' \leftsquigarrow^*_\sigma s \to t$
- check whether t' and $t\sigma$ semi-unify

Heuristic for loop detection of $(\mathcal{P}, \mathcal{R})$ in AProVE:

- If $\mathcal{P} \cup \mathcal{R}$ left-linear: backward narrowing
- If $\mathcal{P} \cup \mathcal{R}$ right-linear: forward narrowing
- If $\mathcal{P} \cup \mathcal{R}$ not left- or right-linear: backward narrowing into variables

To obtain finite search space:

each rule may only applied n times for narrowing

Looping vs. Non-Looping Non-Termination

- Most existing approaches detect loops

$$s \to_{\mathcal{R}}^{n} C[\, s\,\mu\,] \to_{\mathcal{R}}^{n} C[\, C\mu[\, s\,\mu^2\,]\,] \to_{\mathcal{R}}^{n} \ldots$$

- cannot capture non-periodic infinite rewrite sequences

$$\begin{aligned}
\mathsf{f}(\mathsf{tt}, x, y) &\to \mathsf{f}(\mathsf{gt}(x, y),\ \mathsf{dbl}(x),\ \mathsf{s}(y)) \\
\mathsf{gt}(\mathsf{s}(x), 0) &\to \mathsf{tt} \\
\mathsf{gt}(0, y) &\to \mathsf{ff} \\
\mathsf{gt}(\mathsf{s}(x), \mathsf{s}(y)) &\to \mathsf{gt}(x, y)
\end{aligned}
\qquad
\begin{aligned}
\mathsf{dbl}(x) &\to \mathsf{mul}(\mathsf{s}^2(0), x) \\
\mathsf{mul}(x, 0) &\to 0 \\
\mathsf{mul}(x, \mathsf{s}(y)) &\to \mathsf{plus}(\mathsf{mul}(x, y), x) \\
\mathsf{plus}(x, 0) &\to x \\
\mathsf{plus}(x, \mathsf{s}(y)) &\to \mathsf{plus}(\mathsf{s}(x), y)
\end{aligned}$$

$$\begin{aligned}
&\mathsf{f}(\mathsf{tt}, \mathsf{s}^n(0), \mathsf{s}^m(0)) && \to_{\mathcal{R}} \\
&\mathsf{f}(\mathsf{gt}(\mathsf{s}^n(0), \mathsf{s}^m(0)), \mathsf{dbl}(\mathsf{s}^n(0)), \mathsf{s}^{m+1}(0)) && \to_{\mathcal{R}}^{m+1} \\
&\mathsf{f}(\mathsf{tt}, \mathsf{dbl}(\mathsf{s}^n(0)), \mathsf{s}^{m+1}(0)) && \to_{\mathcal{R}} \\
&\mathsf{f}(\mathsf{tt}, \mathsf{mul}(\mathsf{s}^2(0), \mathsf{s}^n(0)), \mathsf{s}^{m+1}(0)) && \to_{\mathcal{R}}^{4\cdot n} \\
&\mathsf{f}(\mathsf{tt}, \mathsf{s}^{2\cdot n}(\mathsf{mul}(\mathsf{s}^2(0), 0)), \mathsf{s}^{m+1}(0)) && \to_{\mathcal{R}} \\
&\mathsf{f}(\mathsf{tt}, \mathsf{s}^{2\cdot n}(0), \mathsf{s}^{m+1}(0)) && \to_{\mathcal{R}} \ldots
\end{aligned}$$

```
while (gt(x,y)) {
  x = dbl(x);
  y = y + 1;     }
```

non-terminating, but not looping

Looping vs. Non-Looping Non-Termination

- Most existing approaches detect loops

$$t \to_{\mathcal{R}}^{n} C[\, s\,\mu \,] \to_{\mathcal{R}}^{n} C[\, C\mu[\, s\,\mu^2 \,]\,] \to_{\mathcal{R}}^{n} \ldots$$

Method for Loop Detection

1. Let $\mathcal{S} := \mathcal{R}$.
2. Check if some $s \to t \in \mathcal{S}$ is a loop. If yes: stop with "non-termination".
3. Modify some $s \to t \in \mathcal{S}$ by narrowing to obtain $s' \to_{\mathcal{R}}^{+} t'$.
4. Let $\mathcal{S} := \mathcal{S} \cup \{s' \to t'\}$ and go to Step 2.

Method for Non-Looping Non-Termination

1. Let $\mathcal{S}$ be a set of pattern rules $p \hookrightarrow q$ corresponding to $\mathcal{R}$.
2. Check if some $p \hookrightarrow q \in \mathcal{S}$ is obviously non-terminating.
   If yes: stop with "non-termination".
3. Modify some $p \hookrightarrow q \in \mathcal{S}$ by narrowing to obtain $p' \hookrightarrow q'$.
4. Let $\mathcal{S} := \mathcal{S} \cup \{p' \hookrightarrow q'\}$ and go to Step 2.

Pattern Terms and Pattern Rules

Pattern Term

Pattern term p : $n \mapsto t\,\sigma^n\,\mu$

represents $\{\underbrace{t\,\mu}_{p(0)}, \underbrace{t\,\sigma\,\mu}_{p(1)}, \underbrace{t\,\sigma^2\,\mu}_{p(2)}, \underbrace{t\,\sigma^3\,\mu}_{p(3)}, \ldots\}$.

- base term t
- pumping substitution σ
- closing substitution μ

Pattern Terms and Pattern Rules

Pattern Term

Pattern term p : $t\,\sigma^n\,\mu$

represents $\{\underbrace{t\,\mu}_{p(0)},\ \underbrace{t\,\sigma\,\mu}_{p(1)},\ \underbrace{t\,\sigma^2\,\mu}_{p(2)},\ \underbrace{t\,\sigma^3\,\mu}_{p(3)},\ \ldots\}$.

- base term t
- pumping substitution σ
- closing substitution μ

Pattern Rule

Pattern rule: $p \hookrightarrow q$

where p, q are pattern terms

$p \hookrightarrow q$ is *correct* w.r.t. TRS $\mathcal{R}$

if $\forall n \in \mathbb{N}: \quad p(n) \to_{\mathcal{R}}^{+} q(n)$

Example: $p = \mathsf{gt}(\mathsf{s}(x), \mathsf{s}(y))\ \ [x/\mathsf{s}(x), y/\mathsf{s}(y)]^n\ \ [x/\mathsf{s}(x), y/0]$

represents $\{\underbrace{\mathsf{gt}(\mathsf{s}^2(x), \mathsf{s}(0))}_{p(0)},\ \underbrace{\mathsf{gt}(\mathsf{s}^3(x), \mathsf{s}^2(0))}_{p(1)},\ \underbrace{\mathsf{gt}(\mathsf{s}^4(x), \mathsf{s}^3(0))}_{p(2)}, \ldots\}$

Example: $\mathsf{gt}(\mathsf{s}(x), \mathsf{s}(y))\ \ [x/\mathsf{s}(x), y/\mathsf{s}(y)]^n\ \ [x/\mathsf{s}(x), y/0] \quad \hookrightarrow \quad \mathsf{tt}\ \varnothing^n\ \varnothing$

correct, since $\forall n \in \mathbb{N}: \quad \mathsf{gt}(\mathsf{s}^{n+2}(x), \mathsf{s}^{n+1}(0)) \to_{\mathcal{R}}^{+} \mathsf{tt}$

Proving Non-Termination of TRSs Automatically

Method for Non-Looping Non-Termination

1. Let $\mathcal{S}$ be a set of pattern rules corresponding to $\mathcal{R}$.
2. Check if some $p \hookrightarrow q \in \mathcal{S}$ is obviously non-terminating.
   If yes: stop with "non-termination".
3. Modify some $p \hookrightarrow q \in \mathcal{S}$ by narrowing to obtain $p' \hookrightarrow q'$.
4. Let $\mathcal{S} := \mathcal{S} \cup \{p' \hookrightarrow q'\}$ and go to Step 2.

Contributions

- **pattern rules** $p \hookrightarrow q$
  to represent *sets* of rewrite sequences $\{p(n) \to_{\mathcal{R}}^{+} q(n) \mid n \in \mathbb{N}\}$
- **inference rules** to deduce new correct pattern rules

Inference Rules

$$\frac{p_1 \hookrightarrow q_1 \quad \dots \quad p_k \hookrightarrow q_k}{p \hookrightarrow q}$$

If $p_1 \hookrightarrow q_1, \dots, p_k \hookrightarrow q_k$ are correct w.r.t. $\mathcal{R}$ then $p \hookrightarrow q$ is also correct w.r.t. $\mathcal{R}$

$$\begin{aligned}
\mathsf{f}(\mathsf{tt}, x, y) &\to \mathsf{f}(\mathsf{gt}(x, y), \mathsf{dbl}(x), \mathsf{s}(y)) \\
\mathsf{gt}(\mathsf{s}(x), 0) &\to \mathsf{tt} \\
\mathsf{gt}(0, y) &\to \mathsf{ff} \\
\mathsf{gt}(\mathsf{s}(x), \mathsf{s}(y)) &\to \mathsf{gt}(x, y) \\
\mathsf{dbl}(x) &\to \mathsf{mul}(\mathsf{s}^2(0), x) \\
\mathsf{mul}(x, 0) &\to 0 \\
\mathsf{mul}(x, \mathsf{s}(y)) &\to \mathsf{plus}(\mathsf{mul}(x, y), x) \\
\mathsf{plus}(x, 0) &\to x \\
\mathsf{plus}(x, \mathsf{s}(y)) &\to \mathsf{plus}(\mathsf{s}(x), y)
\end{aligned}$$

(1) Pattern Rule from TRS

$$\frac{}{\ell\, \varnothing^n\, \varnothing \;\hookrightarrow\; r\, \varnothing^n\, \varnothing} \quad \text{if } \ell \to r \in \mathcal{R}$$

$$\mathsf{gt}(\mathsf{s}(x), \mathsf{s}(y)) \;\; \varnothing^n \;\; \varnothing \quad \hookrightarrow \quad \mathsf{gt}(x, y) \;\; \varnothing^n \;\; \varnothing$$

Inference Rules

(2) Pattern Creation

$$\frac{s\,\varnothing^n\,\varnothing \;\hookrightarrow\; t\,\varnothing^n\,\varnothing}{s\,\sigma^n\,\varnothing \;\hookrightarrow\; t\,\theta^n\,\varnothing}$$

if $s\theta = t\sigma$, and
θ commutes with σ

θ and σ *commute*
if $\theta\,\sigma = \sigma\,\theta$

$$\begin{array}{lllllll}
s\ \sigma^n & \rightarrow^+_{\mathcal{R}} & t\ \sigma^n & = & s\ \theta\,\sigma^{n-1} & = & \\
s\ \sigma^{n-1}\,\theta & \rightarrow^+_{\mathcal{R}} & t\ \sigma^{n-1}\,\theta & = & s\ \theta\,\sigma^{n-2}\,\theta & = & \\
s\ \sigma^{n-2}\,\theta^2 & \rightarrow^+_{\mathcal{R}} & t\ \sigma^{n-2}\,\theta^2 & = & s\ \theta\,\sigma^{n-3}\,\theta^2 & \rightarrow^+_{\mathcal{R}} \ldots \rightarrow^+_{\mathcal{R}} & t\ \theta^n
\end{array}$$

Inference Rules

(2) Pattern Creation

$$\frac{s \, \varnothing^n \, \varnothing \;\hookrightarrow\; t \, \varnothing^n \, \varnothing}{s \, \sigma^n \, \varnothing \;\hookrightarrow\; t \, \theta^n \, \varnothing}$$

if $s\theta = t\sigma$, and
θ commutes with σ

θ and σ *commute*
if $\theta\,\sigma = \sigma\,\theta$

$$\underbrace{\mathsf{gt}(\mathsf{s}(x), \mathsf{s}(y))}_{s} \;\; \varnothing^n \;\; \varnothing \quad \hookrightarrow \quad \underbrace{\mathsf{gt}(x, y)}_{t} \;\; \varnothing^n \;\; \varnothing$$

$$\Downarrow$$

$$\mathsf{gt}(\mathsf{s}(x), \mathsf{s}(y)) \;\; [x/\mathsf{s}(x), y/\mathsf{s}(y)]^n \;\; \varnothing \quad \hookrightarrow \quad \mathsf{gt}(x, y) \;\; \varnothing^n \;\; \varnothing$$

since $\quad s \underbrace{\varnothing}_{\theta} \quad = \quad t \underbrace{[x/\mathsf{s}(x), y/\mathsf{s}(y)]}_{\sigma}$

Inference Rules

(3) Equivalence

$$\frac{p \hookrightarrow q}{p' \hookrightarrow q'}$$ if p is equivalent to p' and q is equivalent to q'

p and p' are *equivalent* if
$\forall n \in \mathbb{N}$: $\quad p(n) = p'(n)$

Goal: narrow $\mathsf{gt}(\mathsf{s}(x), \mathsf{s}(y))\ [x/\mathsf{s}(x), y/\mathsf{s}(y)]^n\ \varnothing \quad \hookrightarrow \quad \mathsf{gt}(x, y)\ \varnothing^n\ \varnothing$
with $\mathsf{gt}(\mathsf{s}(x), 0)\ \varnothing^n\ \varnothing \quad \hookrightarrow \quad \mathsf{tt}\ \varnothing^n\ \varnothing$

Problem: rules have different pumping and closing substitutions

Strategy:

1. Instantiate base terms.
   (Base term of 1st rhs should *contain* base term of 2nd lhs.)
2. Make all 4 pumping substitutions equal.
3. Make all 4 closing substitutions equal.

Inference Rules

(3) Equivalence

$$\frac{p \hookrightarrow q}{p' \hookrightarrow q'}$$ if p is equivalent to p' and q is equivalent to q'

Criteria for Equivalence

- renaming of *domain variables*

$$\begin{array}{rccc} \mathsf{gt}(\mathsf{s}(x),\mathsf{s}(y)) & [x/\mathsf{s}(x), y/\mathsf{s}(y)]^n & \varnothing & \hookrightarrow \\ \mathsf{gt}(x,y) & \varnothing^n & \varnothing & \end{array}$$

$$\Downarrow$$

$$\begin{array}{rccc} \mathsf{gt}(\mathsf{s}(x'),\mathsf{s}(y')) & [x'/\mathsf{s}(x'),\ y'/\mathsf{s}(y')]^n & [x'/x,\ y'/y] & \hookrightarrow \\ \mathsf{gt}(x,y) & \varnothing^n & \varnothing & \end{array}$$

Inference Rules

(3) Equivalence

$$\frac{p \hookrightarrow q}{p' \hookrightarrow q'}$$ if p is equivalent to p' and q is equivalent to q'

Criteria for Equivalence

- renaming of *domain variables*
- modifying substitutions of *irrelevant variables*

$$\begin{array}{rccc} \mathsf{gt}(\mathsf{s}(x),\mathsf{s}(y)) & [x/\mathsf{s}(x), y/\mathsf{s}(y)]^n & \varnothing & \hookrightarrow \\ \mathsf{gt}(x,y) & \varnothing^n & \varnothing & \end{array}$$

$$\Downarrow$$

$$\begin{array}{rccc} \mathsf{gt}(\mathsf{s}(x'),\mathsf{s}(y')) & [x'/\mathsf{s}(x'),\ y'/\mathsf{s}(y')]^n & [x'/x,\ y'/y] & \hookrightarrow \\ \mathsf{gt}(x,y) & [x'/\mathsf{s}(x'),\ y'/\mathsf{s}(y')]^n & [x'/x,\ y'/y] & \end{array}$$

Inference Rules

(4) Instantiation

$$\frac{s\,\delta^n\,\tau \;\hookrightarrow\; t\,\sigma^n\,\mu}{(s\,\rho)\;\delta^n_\rho\;\tau_\rho \;\hookrightarrow\; (t\,\rho)\;\sigma^n_\rho\;\mu_\rho}$$ if $\mathcal{V}(\rho) \cap (\mathrm{dom}(\delta) \cup \mathrm{dom}(\tau) \cup \mathrm{dom}(\sigma) \cup \mathrm{dom}(\mu)) = \varnothing$

$$\sigma_\rho = [x/s\rho \mid x/s \in \sigma] = (\sigma\,\rho)|_{\mathrm{dom}(\sigma)}$$

$$\rho = [x/\mathsf{s}(x),\, y/0]$$

narrow $\mathsf{gt}(\mathsf{s}(x'),\mathsf{s}(y'))\ [x'/\mathsf{s}(x'),\, y'/\mathsf{s}(y')]^n\ \ [x'/x,\, y'/y] \hookrightarrow$
$\mathsf{gt}(x,y)\ [x'/\mathsf{s}(x'),\, y'/\mathsf{s}(y')]^n\ \ [x'/x,\, y'/y]$

with $\mathsf{gt}(\mathsf{s}(x),0)\ \ \varnothing^n\ \ \varnothing \hookrightarrow$
$\mathsf{tt}\ \ \varnothing^n\ \ \varnothing$

Strategy:

1. Instantiate base terms.
   (Base term of 1st rhs should *contain* base term of 2nd lhs.)
2. Make all 4 pumping substitutions equal.
3. Make all 4 closing substitutions equal.

Inference Rules

(4) Instantiation

$$\frac{s\,\delta^n\,\tau \;\hookrightarrow\; t\,\sigma^n\,\mu}{(s\,\rho)\;\delta_\rho^n\;\tau_\rho \;\hookrightarrow\; (t\,\rho)\;\sigma_\rho^n\;\mu_\rho}$$

if $\mathcal{V}(\rho) \cap (\mathrm{dom}(\delta) \cup \mathrm{dom}(\tau) \cup \mathrm{dom}(\sigma) \cup \mathrm{dom}(\mu)) = \varnothing$

$$\begin{aligned}\sigma_\rho &= [x/s\rho \mid x/s \in \sigma] \\ &= (\,\sigma\,\rho\,)|_{\mathrm{dom}(\sigma)} \\ \rho &= [x/\mathsf{s}(x),\, y/0]\end{aligned}$$

narrow $\mathsf{gt}(\mathsf{s}(x'), \mathsf{s}(y'))\; [x'/\mathsf{s}(x'),\, y'/\mathsf{s}(y')]^n\; [x'/\mathsf{s}(x),\, y'/0] \;\hookrightarrow$
$\mathsf{gt}(\mathsf{s}(x), 0)\; [x'/\mathsf{s}(x'),\, y'/\mathsf{s}(y')]^n\; [x'/\mathsf{s}(x),\, y'/0]$

with $\mathsf{gt}(\mathsf{s}(x), 0)\quad \varnothing^n \quad \varnothing \;\hookrightarrow$
$\mathsf{tt}\quad \varnothing^n \quad \varnothing$

Strategy:

1. Instantiate base terms.
   (Base term of 1st rhs should *contain* base term of 2nd lhs.)
2. Make all 4 pumping substitutions equal.
3. Make all 4 closing substitutions equal.

Inference Rules

(4) Instantiation

$$\frac{s\,\delta^n\,\tau \;\hookrightarrow\; t\,\sigma^n\,\mu}{(s\,\rho)\;\delta^n_\rho\;\tau_\rho \;\hookrightarrow\; (t\,\rho)\;\sigma^n_\rho\;\mu_\rho}$$ if $\mathcal{V}(\rho) \cap (\mathrm{dom}(\delta) \cup \mathrm{dom}(\tau) \cup \mathrm{dom}(\sigma) \cup \mathrm{dom}(\mu)) = \varnothing$

$$\sigma_\rho = [x/s\rho \mid x/s \in \sigma] = (\sigma\,\rho)|_{\mathrm{dom}(\sigma)}$$

$$\rho = [x/\mathsf{s}(x),\, y/0]$$

narrow $\mathsf{gt}(\mathsf{s}(x'),\mathsf{s}(y'))\;[x'/\mathsf{s}(x'),\, y'/\mathsf{s}(y')]^n\;[x'/\mathsf{s}(x),\, y'/0] \hookrightarrow$
$\mathsf{gt}(\mathsf{s}(x),0)\;[x'/\mathsf{s}(x'),\, y'/\mathsf{s}(y')]^n\;[x'/\mathsf{s}(x),\, y'/0]$

with $\mathsf{gt}(\mathsf{s}(x),0)\;[x'/\mathsf{s}(x'),\, y'/\mathsf{s}(y')]^n\;\varnothing \hookrightarrow$
$\mathsf{tt}\;[x'/\mathsf{s}(x'),\, y'/\mathsf{s}(y')]^n\;\varnothing$

Strategy:

1. Instantiate base terms.
   (Base term of 1st rhs should *contain* base term of 2nd lhs.)
2. Make all 4 pumping substitutions equal.
3. Make all 4 closing substitutions equal.

Inference Rules

(4) Instantiation

$$\frac{s\,\delta^n\,\tau \;\hookrightarrow\; t\,\sigma^n\,\mu}{(s\,\rho)\;\delta^n_\rho\;\tau_\rho \;\hookrightarrow\; (t\,\rho)\;\sigma^n_\rho\;\mu_\rho}$$

if $\mathcal{V}(\rho) \cap (\mathrm{dom}(\delta) \cup \mathrm{dom}(\tau) \cup \mathrm{dom}(\sigma) \cup \mathrm{dom}(\mu)) = \varnothing$

$$\sigma_\rho = [x/s\rho \mid x/s \in \sigma] = (\sigma\,\rho)|_{\mathrm{dom}(\sigma)}$$

$$\rho = [x/\mathsf{s}(x),\ y/0]$$

narrow $\mathsf{gt}(\mathsf{s}(x'),\mathsf{s}(y'))\ [x'/\mathsf{s}(x'),\ y'/\mathsf{s}(y')]^n\ [x'/\mathsf{s}(x),\ y'/0] \hookrightarrow$
$\mathsf{gt}(\mathsf{s}(x),0)\ [x'/\mathsf{s}(x'),\ y'/\mathsf{s}(y')]^n\ [x'/\mathsf{s}(x),\ y'/0]$

with $\mathsf{gt}(\mathsf{s}(x),0)\ [x'/\mathsf{s}(x'),\ y'/\mathsf{s}(y')]^n\ [x'/\mathsf{s}(x),\ y'/0] \hookrightarrow$
$\mathsf{tt}\ [x'/\mathsf{s}(x'),\ y'/\mathsf{s}(y')]^n\ [x'/\mathsf{s}(x),\ y'/0]$

Strategy:

1. Instantiate base terms.
   (Base term of 1st rhs should *contain* base term of 2nd lhs.)
2. Make all 4 pumping substitutions equal.
3. Make all 4 closing substitutions equal.

Inference Rules

(5) Narrowing

$$\frac{s\,\sigma^n\,\mu \hookrightarrow t\,\sigma^n\,\mu \qquad u\,\sigma^n\,\mu \hookrightarrow v\,\sigma^n\,\mu}{s\,\sigma^n\,\mu \hookrightarrow t[v]_\pi\,\sigma^n\,\mu} \quad \text{if } t|_\pi = u$$

narrow $\quad \mathsf{gt}(\mathsf{s}(x'),\mathsf{s}(y'))\ [x'/\mathsf{s}(x'),\ y'/\mathsf{s}(y')]^n\ [x'/\mathsf{s}(x),\ y'/0] \hookrightarrow$
$\mathsf{gt}(\mathsf{s}(x),0)\ [x'/\mathsf{s}(x'),\ y'/\mathsf{s}(y')]^n\ [x'/\mathsf{s}(x),\ y'/0]$

with $\quad \mathsf{gt}(\mathsf{s}(x),0)\ [x'/\mathsf{s}(x'),\ y'/\mathsf{s}(y')]^n\ [x'/\mathsf{s}(x),\ y'/0] \hookrightarrow$
$\mathsf{tt}\ [x'/\mathsf{s}(x'),\ y'/\mathsf{s}(y')]^n\ [x'/\mathsf{s}(x),\ y'/0]$

$$\Downarrow$$

$\mathsf{gt}(\mathsf{s}(x'),\mathsf{s}(y'))\ [x'/\mathsf{s}(x'),\ y'/\mathsf{s}(y')]^n\ [x'/\mathsf{s}(x),\ y'/0] \hookrightarrow$
$\mathsf{tt}\ [x'/\mathsf{s}(x'),\ y'/\mathsf{s}(y')]^n\ [x'/\mathsf{s}(x),\ y'/0]$

Inference Rules

(4) Instantiation

$$\frac{s\,\delta^n\,\tau \;\hookrightarrow\; t\,\sigma^n\,\mu}{(s\,\rho)\;\delta^n_\rho\;\tau_\rho \;\hookrightarrow\; (t\,\rho)\;\sigma^n_\rho\;\mu_\rho}$$ if $\mathcal{V}(\rho) \cap (\mathrm{dom}(\delta) \cup \mathrm{dom}(\tau) \cup \mathrm{dom}(\sigma) \cup \mathrm{dom}(\mu)) = \varnothing$

$$\sigma_\rho = [x/s\rho \mid x/s \in \sigma] = (\sigma\,\rho)|_{\mathrm{dom}(\sigma)}$$

$$\rho = [x/\mathsf{s}(x'),\; y/\mathsf{s}(y')]$$

narrow $\mathsf{f}(\mathsf{tt}, x, y)$ $\varnothing^n$ $\varnothing$ $\hookrightarrow$
$\mathsf{f}(\mathsf{gt}(x, y), \mathsf{dbl}(x), \mathsf{s}(y))$ $\varnothing^n$ $\varnothing$

with $\mathsf{gt}(\mathsf{s}(x'), \mathsf{s}(y'))$ $[x'/\mathsf{s}(x'),\; y'/\mathsf{s}(y')]^n$ $[x'/\mathsf{s}(x),\; y'/0]$ $\hookrightarrow$
tt $[x'/\mathsf{s}(x'),\; y'/\mathsf{s}(y')]^n$ $[x'/\mathsf{s}(x),\; y'/0]$

Strategy:

1. Instantiate base terms.
   (Base term of 1st rhs should *contain* base term of 2nd lhs.)
2. Make all 4 pumping substitutions equal.
3. Make all 4 closing substitutions equal.

Inference Rules

(4) Instantiation

$$\frac{s\,\delta^n\,\tau \;\hookrightarrow\; t\,\sigma^n\,\mu}{(s\,\rho)\;\delta^n_\rho\;\tau_\rho \;\hookrightarrow\; (t\,\rho)\;\sigma^n_\rho\;\mu_\rho}$$ if $\mathcal{V}(\rho) \cap (\mathrm{dom}(\delta) \cup \mathrm{dom}(\tau) \cup \mathrm{dom}(\sigma) \cup \mathrm{dom}(\mu)) = \varnothing$

$$\sigma_\rho = [x/s\rho \mid x/s \in \sigma] = (\,\sigma\,\rho\,)|_{\mathrm{dom}(\sigma)}$$

$$\rho = [x/\mathsf{s}(x'),\; y/\mathsf{s}(y')]$$

narrow $\mathsf{f}(\mathsf{tt}, \mathsf{s}(x'), \mathsf{s}(y'))$ $\varnothing^n$ $\varnothing$ $\hookrightarrow$
$\mathsf{f}(\mathsf{gt}(\mathsf{s}(x'), \mathsf{s}(y')), \mathsf{dbl}(\mathsf{s}(x')), \mathsf{s}^2(y'))$ $\varnothing^n$ $\varnothing$

with $\mathsf{gt}(\mathsf{s}(x'), \mathsf{s}(y'))$ $[x'/\mathsf{s}(x'),\; y'/\mathsf{s}(y')]^n$ $[x'/\mathsf{s}(x),\; y'/0]$ $\hookrightarrow$
tt $[x'/\mathsf{s}(x'),\; y'/\mathsf{s}(y')]^n$ $[x'/\mathsf{s}(x),\; y'/0]$

Strategy:

1. Instantiate base terms.
   (Base term of 1st rhs should *contain* base term of 2nd lhs.)
2. Make all 4 pumping substitutions equal.
3. Make all 4 closing substitutions equal.

Inference Rules

(6) Instantiating σ

$$\frac{s\,\delta^n\,\tau \;\hookrightarrow\; t\,\sigma^n\,\mu}{s\,(\delta\,\rho)^n\,\tau \;\hookrightarrow\; t\,(\sigma\,\rho)^n\,\mu}$$ if ρ commutes with δ, τ, σ, and μ

$\rho = [x'/\mathsf{s}(x'),\ y'/\mathsf{s}(y')]$

narrow $\mathsf{f}(\mathsf{tt},\mathsf{s}(x'),\mathsf{s}(y'))\ [x'/\mathsf{s}(x'),\ y'/\mathsf{s}(y')]^n\ \varnothing \hookrightarrow$
$\mathsf{f}(\mathsf{gt}(\mathsf{s}(x'),\mathsf{s}(y')),\mathsf{dbl}(\mathsf{s}(x')),\mathsf{s}^2(y'))\ [x'/\mathsf{s}(x'),\ y'/\mathsf{s}(y')]^n\ \varnothing$

with $\mathsf{gt}(\mathsf{s}(x'),\mathsf{s}(y'))\ [x'/\mathsf{s}(x'),\ y'/\mathsf{s}(y')]^n\ [x'/\mathsf{s}(x),\ y'/0] \hookrightarrow$
$\mathsf{tt}\ [x'/\mathsf{s}(x'),\ y'/\mathsf{s}(y')]^n\ [x'/\mathsf{s}(x),\ y'/0]$

Strategy:

1. Instantiate base terms.
(Base term of 1st rhs should *contain* base term of 2nd lhs.)
2. Make all 4 pumping substitutions equal.
3. Make all 4 closing substitutions equal.

Inference Rules

(7) Instantiating μ

$$\frac{s\,\delta^n\,\tau \;\hookrightarrow\; t\,\sigma^n\,\mu}{s\,\delta^n\,(\tau\,\rho) \;\hookrightarrow\; t\,\sigma^n\,(\mu\,\rho)}$$

$\rho = [x'/\mathsf{s}(x'),\; y'/0]$

narrow

$$\mathsf{f}(\mathsf{tt}, \mathsf{s}(x'), \mathsf{s}(y'))\; [x'/\mathsf{s}(x'),\; y'/\mathsf{s}(y')]^n\; [x'/\mathsf{s}(x),\; y'/0] \;\hookrightarrow$$
$$\mathsf{f}(\mathsf{gt}(\mathsf{s}(x'), \mathsf{s}(y')), \mathsf{dbl}(\mathsf{s}(x')), \mathsf{s}^2(y'))\; [x'/\mathsf{s}(x'),\; y'/\mathsf{s}(y')]^n\; [x'/\mathsf{s}(x),\; y'/0]$$

with

$$\mathsf{gt}(\mathsf{s}(x'), \mathsf{s}(y'))\; [x'/\mathsf{s}(x'),\; y'/\mathsf{s}(y')]^n\; [x'/\mathsf{s}(x),\; y'/0] \;\hookrightarrow$$
$$\mathsf{tt}\; [x'/\mathsf{s}(x'),\; y'/\mathsf{s}(y')]^n\; [x'/\mathsf{s}(x),\; y'/0]$$

Strategy:

1. Instantiate base terms.
   (Base term of 1st rhs should *contain* base term of 2nd lhs.)
2. Make all 4 pumping substitutions equal.
3. Make all 4 closing substitutions equal.

Inference Rules

(5) Narrowing

$$\frac{s\,\sigma^n\,\mu \;\hookrightarrow\; t\,\sigma^n\,\mu \qquad u\,\sigma^n\,\mu \;\hookrightarrow\; v\,\sigma^n\,\mu}{s\,\sigma^n\,\mu \;\hookrightarrow\; t[v]_\pi\,\sigma^n\,\mu} \quad \text{if } t|_\pi = u$$

narrow

$$\mathsf{f}(\mathsf{tt}, \mathsf{s}(x'), \mathsf{s}(y'))\; [x'/\mathsf{s}(x'),\, y'/\mathsf{s}(y')]^n\; [x'/\mathsf{s}(x),\, y'/0] \;\hookrightarrow$$
$$\mathsf{f}(\mathsf{gt}(\mathsf{s}(x'), \mathsf{s}(y')), \mathsf{dbl}(\mathsf{s}(x')), \mathsf{s}^2(y'))\; [x'/\mathsf{s}(x'),\, y'/\mathsf{s}(y')]^n\; [x'/\mathsf{s}(x),\, y'/0]$$

with

$$\mathsf{gt}(\mathsf{s}(x'), \mathsf{s}(y'))\; [x'/\mathsf{s}(x'),\, y'/\mathsf{s}(y')]^n\; [x'/\mathsf{s}(x),\, y'/0] \;\hookrightarrow$$
$$\mathsf{tt}\; [x'/\mathsf{s}(x'),\, y'/\mathsf{s}(y')]^n\; [x'/\mathsf{s}(x),\, y'/0]$$

$$\Downarrow$$

$$\mathsf{f}(\mathsf{tt}, \mathsf{s}(x'), \mathsf{s}(y'))\; [x'/\mathsf{s}(x'),\, y'/\mathsf{s}(y')]^n\; [x'/\mathsf{s}(x),\, y'/0] \;\hookrightarrow$$
$$\mathsf{f}(\mathsf{tt}, \mathsf{dbl}(\mathsf{s}(x')), \mathsf{s}^2(y'))\; [x'/\mathsf{s}(x'),\, y'/\mathsf{s}(y')]^n\; [x'/\mathsf{s}(x),\, y'/0]$$

Proving Non-Termination of TRSs Automatically

Method for Non-Looping Non-Termination

1. Let $\mathcal{S}$ be a set of pattern rules corresponding to $\mathcal{R}$.
2. Check if some $p \hookrightarrow q \in \mathcal{S}$ is obviously non-terminating.
   If yes: stop with "non-termination".
3. Modify some $p \hookrightarrow q \in \mathcal{S}$ by narrowing to obtain $p' \hookrightarrow q'$.
4. Let $\mathcal{S} := \mathcal{S} \cup \{p' \hookrightarrow q'\}$ and go to Step 2.

Contributions

- **pattern rules** $p \hookrightarrow q$
  to represent *sets* of rewrite sequences $\{p(n) \to_{\mathcal{R}}^{+} q(n) \mid n \in \mathbb{N}\}$
- **inference rules** to deduce new correct pattern rules
- **criterion for obvious non-termination** of pattern rules

Non-Termination Criterion

Criterion for Non-Termination of $\mathcal{R}$

- $s\,\delta^n\,\tau \hookrightarrow t\,\sigma^n\,\mu$ correct w.r.t. $\mathcal{R}$
- $s\,\delta^a = t|_\pi$ for some $a \in \mathbb{N}$
- $\sigma = \delta^b\,\delta'$, $\mu = \tau\,\tau'$ for some δ', τ' and some $b \in \mathbb{N}$ where δ' commutes with δ and τ

$$
\begin{array}{lllllll}
s & \delta^n & \tau & \to_{\mathcal{R}}^{+} & t & & \sigma^n & \mu \\
 & & & \trianglerighteq & t|_\pi & & \sigma^n & \mu \\
 & & & = & s & \delta^a & \sigma^n & \mu \\
 & & & = & s & \delta^a & (\delta^b\,\delta')^n & (\tau\,\tau') \\
 & & & = & s & \delta^{a+b\cdot n} & & \tau\,\delta'^n\,\tau'
\end{array}
$$

Thus: $s\,\delta^n\,\tau$ rewrites to an instance of $s\,\delta^{a+b\cdot n}\,\tau$

Non-Termination Criterion

Criterion for Non-Termination of $\mathcal{R}$

- $s\,\delta^n\,\tau \hookrightarrow t\,\sigma^n\,\mu$ correct w.r.t. $\mathcal{R}$
- $s\,\delta^a = t|_\pi$ for some $a \in \mathbb{N}$
- $\sigma = \delta^b\,\delta'$, $\mu = \tau\,\tau'$ for some δ', τ' and some $b \in \mathbb{N}$ where δ' commutes with δ and τ

$$\mathsf{f}(\mathsf{tt}, \mathsf{s}^2(x'), \mathsf{s}(y'))\,[x'/\mathsf{s}(x'),\ y'/\mathsf{s}(y')]^n\ [y'/0] \hookrightarrow$$
$$\mathsf{f}(\mathsf{tt}, \mathsf{s}^3(x'), \mathsf{s}^2(y'))\,[x'/\mathsf{s}^2(x'),\ y'/\mathsf{s}(y')]^n\ [x'/\mathsf{s}(\mathsf{mul}(\mathsf{s}^2(0), x')),\ y'/0]$$

Thus: $\mathsf{f}(\mathsf{tt}, \mathsf{s}^{n+2}(x'), \mathsf{s}^{n+1}(0)) \to_{\mathcal{R}}^{+} \mathsf{f}(\mathsf{tt}, \mathsf{s}^{2\cdot n+4}(\mathsf{mul}(\mathsf{s}^2(0), x')), \mathsf{s}^{n+2}(0))$

Non-Termination Criterion

Criterion for Non-Termination of $\mathcal{R}$

- $s\,\delta^n\,\tau \hookrightarrow t\,\sigma^n\,\mu$ correct w.r.t. $\mathcal{R}$
- $s\,\delta^a = t|_\pi$ for some $a \in \mathbb{N}$
- $\sigma = \delta^b\,\delta'$, $\mu = \tau\,\tau'$ for some δ', τ' and some $b \in \mathbb{N}$ where δ' commutes with δ and τ

$$\mathsf{f}(\mathsf{tt}, \mathsf{s}^2(x'), \mathsf{s}(y'))\ [x'/\mathsf{s}(x'),\ y'/\mathsf{s}(y')]^n\ [y'/0] \hookrightarrow$$
$$\mathsf{f}(\mathsf{tt}, \mathsf{s}^3(x'), \mathsf{s}^2(y'))\ [x'/\mathsf{s}^2(x'),\ y'/\mathsf{s}(y')]^n\ [x'/\mathsf{s}(\mathsf{mul}(\mathsf{s}^2(0), x')),\ y'/0]$$

- $\underbrace{\mathsf{f}(\mathsf{tt}, \mathsf{s}^2(x'), \mathsf{s}(y'))}_{s}\ \underbrace{[x'/\mathsf{s}(x'),\ y'/\mathsf{s}(y')]}_{\delta} = \underbrace{\mathsf{f}(\mathsf{tt}, \mathsf{s}^3(x'), \mathsf{s}^2(y'))}_{t}$

- $\underbrace{[x'/\mathsf{s}^2(x'),\ y'/\mathsf{s}(y')]}_{\sigma} = \underbrace{[x'/\mathsf{s}(x'),\ y'/\mathsf{s}(y')]}_{\delta}\ \underbrace{[x'/\mathsf{s}(x')]}_{\delta'}$

  $\underbrace{[x'/\mathsf{s}(\mathsf{mul}(\mathsf{s}^2(0), x')),\ y'/0]}_{\mu} = \underbrace{[y'/0]}_{\tau}\ \underbrace{[x'/\mathsf{s}(\mathsf{mul}(\mathsf{s}^2(0), x'))]}_{\tau'}$

Proving Non-Termination of TRSs Automatically

Method for Non-Looping Non-Termination

1. Let $\mathcal{S}$ be a set of pattern rules corresponding to $\mathcal{R}$.
2. Check if some $p \hookrightarrow q \in \mathcal{S}$ is obviously non-terminating.
   If yes: stop with "non-termination".
3. Modify some $p \hookrightarrow q \in \mathcal{S}$ by narrowing to obtain $p' \hookrightarrow q'$.
4. Let $\mathcal{S} := \mathcal{S} \cup \{p' \hookrightarrow q'\}$ and go to Step 2.

Contributions

- **pattern rules** $p \hookrightarrow q$
  to represent *sets* of rewrite sequences $\{p(n) \to_{\mathcal{R}}^{+} q(n) \mid n \in \mathbb{N}\}$
- **inference rules** to deduce new correct pattern rules
- **criterion for obvious non-termination** of pattern rules
- **strategy** to apply the inference rules and the non-termination criterion
- **implementation** and evaluation in **AProVE**

Proving Non-Termination of TRSs Automatically

- DP framework proves and disproves termination of TRSs
- approach detects also non-looping non-terminating TRSs by combining
  - techniques for looping non-termination of TRSs
  - techniques for non-looping non-termination of SRSs (Oppelt '08)

Contributions

- **pattern rules** $p \hookrightarrow q$
  to represent *sets* of rewrite sequences $\{p(n) \to_{\mathcal{R}}^{+} q(n) \mid n \in \mathbb{N}\}$
- **inference rules** to deduce new correct pattern rules
- **criterion for obvious non-termination** of pattern rules
- **strategy** to apply the inference rules and the non-termination criterion
- **implementation** and evaluation in **AProVE** (also as **DP processor**)

Overview

I. Termination of Term Rewriting

1. Termination of Term Rewrite Systems
2. Non-Termination of Term Rewrite Systems
3. Complexity of Term Rewrite Systems (CADE '11)
4. Termination of Integer Term Rewrite Systems

II. Termination of Programs

1. Termination of Functional Programs (Haskell)
2. Termination of Logic Programs (Prolog)
3. Termination of Imperative Programs (Java)

Termination Analysis of TRSs

- useful for termination of programs (Java, Haskell, Prolog, ...)
- Dependency Pair Framework
  - modular combination of different techniques
  - automatable

Complexity Analysis of TRSs

- should be useful for of programs ⇒ *Innermost Runtime Complexity*
- adapt Dependency Pair Framework
  - Hirokawa & Moser (IJCAR '08, LPAR '08)
    - first adaption of DPs for complexity
    - not modular
  - Zankl & Korp (RTA '10)
    - modular approach based on relative rewriting
    - for *Derivational Complexity*
      (cannot exploit strength of DPs for innermost rewriting)
  - new approach: *direct* adaption of DP framework
    - modular combination of different techniques
    - automated and more powerful than previous approaches

Innermost Runtime Complexity

$$\mathcal{R}: \quad \begin{array}{rcl} \mathsf{double}(0) & \to & 0 \\ \mathsf{double}(\mathsf{s}(x)) & \to & \mathsf{s}(\mathsf{s}(\mathsf{double}(x))) \end{array}$$

- **Derivation Height** $\mathsf{dh}(t)$**:** length of longest $\stackrel{\mathsf{i}}{\to}_{\mathcal{R}}$-sequence with t
  - $\mathsf{dh}(\,\mathsf{double}(\mathsf{s}^k(0))\,) = k + 1$
  - $\mathsf{dh}(\,\mathsf{double}^k(\mathsf{s}(0))\,) \approx 2^k$

- **Basic Terms** $f(t_1, \ldots, t_n)$

  f defined symbol (double), $t_1, \ldots, t_n$ no defined symbols (s, 0)

- **Complexity** $\iota_{\mathcal{R}}$ **of TRS** $\mathcal{R}$**:**
  length of longest $\stackrel{\mathsf{i}}{\to}_{\mathcal{R}}$-sequence with basic term t where $|t| \leq n$

  $\iota_{\mathcal{R}} = \mathcal{P}ol_0$ iff length $\in \mathcal{O}(1)$ $\qquad$ $\iota_{\mathcal{R}} = \mathcal{P}ol_1$ iff length $\in \mathcal{O}(n)$

  $\iota_{\mathcal{R}} = \mathcal{P}ol_2$ iff length $\in \mathcal{O}(n^2)$ $\qquad$ ...

- Example: $\iota_{\mathcal{R}} = \mathcal{P}ol_1$

Dependency Tuples

$$\begin{aligned} \mathsf{m}(x,y) &\to \mathsf{if}(\mathsf{gt}(x,y),x,y) \\ \mathsf{if}(\mathsf{true},x,y) &\to \mathsf{s}(\mathsf{m}(\mathsf{p}(x),y)) \\ \mathsf{if}(\mathsf{false},x,y) &\to 0 \end{aligned} \qquad \begin{aligned} \mathsf{gt}(0,k) &\to \mathsf{false} \\ \mathsf{gt}(\mathsf{s}(n),0) &\to \mathsf{true} \\ \mathsf{gt}(\mathsf{s}(n),\mathsf{s}(k)) &\to \mathsf{gt}(n,k) \end{aligned} \qquad \begin{aligned} \mathsf{p}(0) &\to 0 \\ \mathsf{p}(\mathsf{s}(n)) &\to n \end{aligned}$$

- **Termination Analysis: Dependency Pairs**

  compare lhs with subterms of rhs that start with defined symbol

$$\begin{aligned} \mathsf{m}^\sharp(x,y) &\to \mathsf{if}^\sharp(\mathsf{gt}(x,y),x,y) \\ \mathsf{m}^\sharp(x,y) &\to \mathsf{gt}^\sharp(x,y) \end{aligned} \qquad \begin{aligned} \mathsf{if}^\sharp(\mathsf{true},x,y) &\to \mathsf{m}^\sharp(\mathsf{p}(x),y) \\ \mathsf{if}^\sharp(\mathsf{true},x,y) &\to \mathsf{p}^\sharp(x) \\ \mathsf{gt}^\sharp(\mathsf{s}(n),\mathsf{s}(k)) &\to \mathsf{gt}^\sharp(n,k) \end{aligned}$$

- **Complexity Analysis: Dependency Tuples**

  compare lhs with *all* defined subterms of rhs *at once*

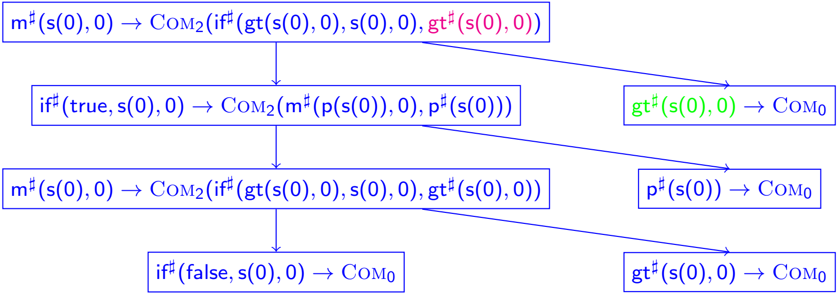
$$\begin{aligned} \mathsf{m}^\sharp(x,y) &\to \mathrm{COM}_2(\mathsf{if}^\sharp(\mathsf{gt}(x,y),x,y),\mathsf{gt}^\sharp(x,y)) \\ \mathsf{if}^\sharp(\mathsf{true},x,y) &\to \mathrm{COM}_2(\mathsf{m}^\sharp(\mathsf{p}(x),y),\mathsf{p}^\sharp(x)) \\ \mathsf{if}^\sharp(\mathsf{false},x,y) &\to \mathrm{COM}_0 \end{aligned} \qquad \begin{aligned} \mathsf{p}^\sharp(0) &\to \mathrm{COM}_0 \\ \mathsf{p}^\sharp(\mathsf{s}(n)) &\to \mathrm{COM}_0 \\ \mathsf{gt}^\sharp(0,k) &\to \mathrm{COM}_0 \\ \mathsf{gt}^\sharp(\mathsf{s}(n),0) &\to \mathrm{COM}_0 \\ \mathsf{gt}^\sharp(\mathsf{s}(n),\mathsf{s}(k)) &\to \mathrm{COM}_1(\mathsf{gt}^\sharp(n,k)) \end{aligned}$$

Chain Trees

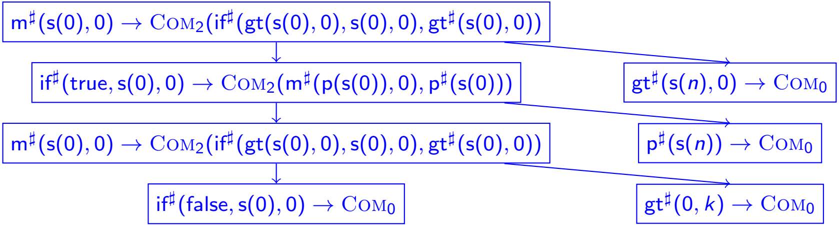
$$\mathrm{DT}(\mathcal{R}):\quad \begin{aligned} \mathsf{m}^\sharp(x,y) &\to \textsc{Com}_2(\mathsf{if}^\sharp(\mathsf{gt}(x,y),x,y),\mathsf{gt}^\sharp(x,y)) & \mathsf{p}^\sharp(0) &\to \textsc{Com}_0\\ \mathsf{if}^\sharp(\mathsf{true},x,y) &\to \textsc{Com}_2(\mathsf{m}^\sharp(\mathsf{p}(x),y),\mathsf{p}^\sharp(x)) & \mathsf{p}^\sharp(\mathsf{s}(n)) &\to \textsc{Com}_0\\ \mathsf{if}^\sharp(\mathsf{false},x,y) &\to \textsc{Com}_0 & \mathsf{gt}^\sharp(0,k) &\to \textsc{Com}_0\\ && \mathsf{gt}^\sharp(\mathsf{s}(n),0) &\to \textsc{Com}_0\\ && \mathsf{gt}^\sharp(\mathsf{s}(n),\mathsf{s}(k)) &\to \textsc{Com}_1(\mathsf{gt}^\sharp(n,k)) \end{aligned}$$

$(\mathcal{D},\mathcal{R})$-Chain Tree:

Edge $\sigma_1(u^\sharp \to \textsc{Com}_n(v_1^\sharp,\ldots,v_n^\sharp))$ to $\sigma_2(w^\sharp \to \textsc{Com}_m(\ldots))$ if $v_i^\sharp\sigma_1 \xrightarrow{\mathrm{i}}{}^*_{\mathcal{R}} w^\sharp\sigma_2$



Chain Trees and Complexity



$\iota_{\mathcal{R}}$: length of longest $\xrightarrow{\mathsf{i}}_{\mathcal{R}}$-sequence for $|t| \leq n$

$\iota_{\langle \mathcal{D},\mathcal{S},\mathcal{R}\rangle}$: maximal number of nodes from $\mathcal{S}$ in chain tree with root $t^\sharp \to \textsc{Com}(\ldots)$ for $|t| \leq n$

Theorem

If $\mathcal{D} = DT(\mathcal{R})$, then $\iota_{\mathcal{R}} \leq \iota_{\langle \mathcal{D},\mathcal{D},\mathcal{R}\rangle}$.

Chain Trees and Complexity

Theorem

If $\mathcal{D} = DT(\mathcal{R})$, then $\iota_{\mathcal{R}} \leq \iota_{\langle \mathcal{D},\mathcal{D},\mathcal{R}\rangle}$.

⇒Find out $\iota_{\langle \mathcal{D},\mathcal{S},\mathcal{R}\rangle}$

⇒Repeatedly replace DT problem $\langle \mathcal{D},\mathcal{S},\mathcal{R}\rangle$ by *simpler* $\langle \mathcal{D}',\mathcal{S}',\mathcal{R}'\rangle$, examine $\iota_{\langle \mathcal{D}',\mathcal{S}',\mathcal{R}'\rangle}$

⇒Start with canonical DT problem $\langle \mathrm{DT}(\mathcal{R}), \mathrm{DT}(\mathcal{R}), \mathcal{R}\rangle$

DT Processor: $Proc(P) = (c, P')$ $\quad P, P'$ DT problems, $c \in \{\mathcal{P}ol_0, \mathcal{P}ol_1, \ldots\}$
where $\iota_P \leq \max(c, \iota_{P'})$

Proof Chain: $P_0 \overset{c_1}{\rightsquigarrow} P_1 \overset{c_2}{\rightsquigarrow} P_2 \overset{c_3}{\rightsquigarrow} \ldots \overset{c_k}{\rightsquigarrow} P_k$ $\quad P_0 = \langle \mathrm{DT}(\mathcal{R}), \mathrm{DT}(\mathcal{R}), \mathcal{R}\rangle$ canonical
$\iota_{\mathcal{R}} \leq \iota_{P_0} \leq \max(c_1, c_2, \ldots, c_k)$ $\quad P_k = \langle \mathcal{D}_k, \varnothing, \mathcal{R}_k\rangle$ solved

Leaf Removal Processor

Dependency Graph: edge from DT $u \to v$ to $w \to t$ in dep. graph iff
edge from $\sigma_1(u \to v)$ to $\sigma_2(w \to t)$ in chain tree

Leaf Removal Processor: $\langle \mathcal{D}, \mathcal{S}, \mathcal{R} \rangle \overset{\mathcal{P}ol_0}{\rightsquigarrow} \langle \mathcal{D} \setminus \{w \to t\},\ \mathcal{S} \setminus \{w \to t\},\ \mathcal{R} \rangle$
if $w \to t$ is leaf in dependency graph

Example: $\langle \mathcal{D}, \mathcal{D}, \mathcal{R} \rangle \overset{\mathcal{P}ol_0}{\rightsquigarrow} \langle \mathcal{D}', \mathcal{D}', \mathcal{R} \rangle$

$\mathcal{R}$: $\mathsf{q}(0, \mathsf{s}(y), \mathsf{s}(z)) \to 0$, $\mathsf{q}(\mathsf{s}(x), \mathsf{s}(y), z) \to \mathsf{q}(x, y, z)$, $\mathsf{q}(x, 0, \mathsf{s}(z)) \to \mathsf{s}(\mathsf{q}(x, \mathsf{s}(z), \mathsf{s}(z)))$

$\mathcal{D}'$: $\mathsf{q}^\sharp(0, \mathsf{s}(y), \mathsf{s}(z)) \to \mathrm{Com}_0$

$\mathsf{q}^\sharp(\mathsf{s}(x), \mathsf{s}(y), z) \to \mathrm{Com}_1(\mathsf{q}^\sharp(x, y, z))$

$\mathsf{q}^\sharp(x, 0, \mathsf{s}(z)) \to \mathrm{Com}_1(\mathsf{q}^\sharp(x, \mathsf{s}(z), \mathsf{s}(z)))$

Usable Rules Processor

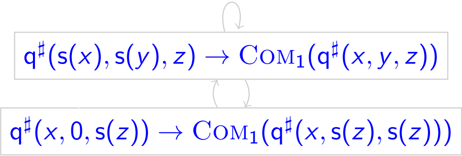
Usable Rules $\mathcal{U}_{\mathcal{R}}(\mathcal{D})$: rules from $\mathcal{R}$ that can reduce rhs of $\mathcal{D}$

Usable Rules Processor: $\langle \mathcal{D}, \mathcal{S}, \mathcal{R} \rangle \overset{\mathcal{P}ol_0}{\rightsquigarrow} \langle \mathcal{D},\ \mathcal{S},\ \mathcal{U}_{\mathcal{R}}(\mathcal{D}) \rangle$

Example: $\langle \mathcal{D}, \mathcal{D}, \mathcal{R} \rangle \overset{\mathcal{P}ol_0}{\rightsquigarrow} \langle \mathcal{D}', \mathcal{D}', \mathcal{R} \rangle \overset{\mathcal{P}ol_0}{\rightsquigarrow} \langle \mathcal{D}', \mathcal{D}', \varnothing \rangle$

$\mathcal{U}_{\mathcal{R}}(\mathcal{D}')$:

$\mathcal{D}'$:



Extended DT Problems

Extended DT Problem: $\langle \mathcal{D}, \mathcal{S}, \mathcal{K}, \mathcal{R} \rangle$

- when computing $\iota_{\langle \mathcal{D},\mathcal{S},\mathcal{R}\rangle}$, we already took $\iota_{\langle \mathcal{D},\mathcal{K},\mathcal{R}\rangle}$ into account
- $\iota_{\langle \mathcal{D},\mathcal{S},\mathcal{K},\mathcal{R}\rangle} = \begin{cases} \iota_{\langle \mathcal{D},\mathcal{S},\mathcal{R}\rangle}, & \text{if } \iota_{\langle \mathcal{D},\mathcal{S},\mathcal{R}\rangle} > \iota_{\langle \mathcal{D},\mathcal{K},\mathcal{R}\rangle} \\ \mathcal{P}ol_0, & \text{if } \iota_{\langle \mathcal{D},\mathcal{S},\mathcal{R}\rangle} \leq \iota_{\langle \mathcal{D},\mathcal{K},\mathcal{R}\rangle} \end{cases}$

Canonical Extended DT Problem: $\langle \mathsf{DT}(\mathcal{R}), \mathsf{DT}(\mathcal{R}), \varnothing, \mathcal{R} \rangle$

Example: $\langle \mathcal{D}, \mathcal{D}, \varnothing, \mathcal{R} \rangle \overset{\mathcal{P}ol_0}{\leadsto} \langle \mathcal{D}', \mathcal{D}', \varnothing, \mathcal{R} \rangle \qquad \overset{\mathcal{P}ol_0}{\leadsto} \langle \mathcal{D}', \mathcal{D}', \varnothing, \varnothing \rangle$

$\mathcal{D}'$:

$\mathsf{q}^\sharp(\mathsf{s}(x), \mathsf{s}(y), z) \to \textsc{Com}_1(\mathsf{q}^\sharp(x, y, z))$

$\mathsf{q}^\sharp(x, \mathsf{0}, \mathsf{s}(z)) \to \textsc{Com}_1(\mathsf{q}^\sharp(x, \mathsf{s}(z), \mathsf{s}(z)))$

Reduction Pair Processor

Termination: $\ell \succsim r$ for all DPs and rules, remove DPs with $\ell \succ r$

Complexity: $\ell \succsim r$ for all DTs and rules, move DTs with $\ell \succ r$ *from* $\mathcal{S}$ *to* $\mathcal{K}$

Reduction Pair Processor: $\langle \mathcal{D}, \mathcal{S}, \mathcal{K}, \mathcal{R} \rangle \overset{\mathcal{P}ol_m}{\leadsto} \langle \mathcal{D},\ \mathcal{S} \setminus \mathcal{D}_{\succ},\ \mathcal{K} \cup \mathcal{D}_{\succ},\ \mathcal{R} \rangle$ if

- $\mathcal{D} \subseteq \succsim \cup \succ, \quad \mathcal{R} \subseteq \succsim$
- m is the maximal degree of polynomials $[f^\sharp]$

Example: $\langle \mathcal{D}, \mathcal{D}, \varnothing, \mathcal{R} \rangle \overset{\mathcal{P}ol_0}{\leadsto} \langle \mathcal{D}', \mathcal{D}', \varnothing, \mathcal{R} \rangle \overset{\mathcal{P}ol_0}{\leadsto} \langle \mathcal{D}', \mathcal{D}', \varnothing, \varnothing \rangle$

Polynomial Order

$$[\mathrm{COM}_1](x) = x$$
$$[\mathsf{q}^\sharp](x, y, z) = x$$
$$[\mathsf{s}](x) = x + 1$$

(1) $\mathsf{q}^\sharp(\mathsf{s}(x), \mathsf{s}(y), z) \to \mathrm{COM}_1(\mathsf{q}^\sharp(x, y, z))$

(2) $\mathsf{q}^\sharp(x, 0, \mathsf{s}(z)) \to \mathrm{COM}_1(\mathsf{q}^\sharp(x, \mathsf{s}(z), \mathsf{s}(z)))$

Reduction Pair Processor

Termination: $\ell \gtrsim r$ for all DPs and rules, remove DPs with $\ell \succ r$

Complexity: $\ell \gtrsim r$ for all DTs and rules, move DTs with $\ell \succ r$ *from* $\mathcal{S}$ *to* $\mathcal{K}$

Reduction Pair Processor: $\langle \mathcal{D}, \mathcal{S}, \mathcal{K}, \mathcal{R} \rangle \overset{\mathcal{P}ol_m}{\rightsquigarrow} \langle \mathcal{D}, \mathcal{S} \setminus \mathcal{D}_\succ, \mathcal{K} \cup \mathcal{D}_\succ, \mathcal{R} \rangle$ if

- $\mathcal{D} \subseteq \gtrsim \cup \succ, \quad \mathcal{R} \subseteq \gtrsim$
- m is the maximal degree of polynomials $[f^\sharp]$

Example: $\langle \mathcal{D}, \mathcal{D}, \varnothing, \mathcal{R} \rangle \overset{\mathcal{P}ol_0}{\rightsquigarrow} \langle \mathcal{D}', \mathcal{D}', \varnothing, \mathcal{R} \rangle \overset{\mathcal{P}ol_0}{\rightsquigarrow} \langle \mathcal{D}', \mathcal{D}', \varnothing, \varnothing \rangle$

$\overset{\mathcal{P}ol_1}{\rightsquigarrow} \langle \mathcal{D}', \{(2)\}, \{(1)\}, \varnothing \rangle$

Polynomial Order

$$[\mathrm{Com}_1](x) = x$$
$$[\mathsf{q}^\sharp](x, y, z) = x$$
$$[\mathsf{s}](x) = x + 1$$

(1) $\mathsf{q}^\sharp(\mathsf{s}(x), \mathsf{s}(y), z) \succ \mathrm{Com}_1(\mathsf{q}^\sharp(x, y, z))$

(2) $\mathsf{q}^\sharp(x, 0, \mathsf{s}(z)) \gtrsim \mathrm{Com}_1(\mathsf{q}^\sharp(x, \mathsf{s}(z), \mathsf{s}(z)))$

Knowledge Propagation Processor

Lemma: $\iota_{\langle \mathcal{D}, \{w \to t\}, \mathcal{R} \rangle} \leq \iota_{\langle \mathcal{D}, Pre(w \to t), \mathcal{R} \rangle}$

- $Pre(w \to t)$: all predecessors of $w \to t$ in dependency graph
- $\langle \mathcal{D}, \mathcal{S}, \mathcal{K}, \mathcal{R} \rangle$: do not take $\iota_{\langle \mathcal{D}, \mathcal{S}, \mathcal{R} \rangle}$ into account if $\iota_{\langle \mathcal{D}, \mathcal{S}, \mathcal{R} \rangle} \leq \iota_{\langle \mathcal{D}, \mathcal{K}, \mathcal{R} \rangle}$

KP Processor: $\langle \mathcal{D}, \mathcal{S}, \mathcal{K}, \mathcal{R} \rangle \overset{\mathcal{P}ol_0}{\rightsquigarrow} \langle \mathcal{D},\ \mathcal{S} \setminus \{w \to t\},\ \mathcal{K} \cup \{w \to t\},\ \mathcal{R} \rangle$
if $w \to t \in \mathcal{S}$ and $Pre(w \to t) \subseteq \mathcal{K}$

Example: $\langle \mathcal{D}, \mathcal{D}, \varnothing, \mathcal{R} \rangle \overset{\mathcal{P}ol_0}{\rightsquigarrow} \langle \mathcal{D}', \mathcal{D}', \varnothing, \mathcal{R} \rangle \overset{\mathcal{P}ol_0}{\rightsquigarrow} \langle \mathcal{D}', \mathcal{D}', \varnothing, \varnothing \rangle$

$\overset{\mathcal{P}ol_1}{\rightsquigarrow} \langle \mathcal{D}', \{(2)\}, \{(1)\}, \varnothing \rangle \overset{\mathcal{P}ol_0}{\rightsquigarrow} \langle \mathcal{D}', \varnothing, \{(1), (2)\}, \varnothing \rangle$

$(1)\ \mathsf{q}^\sharp(\mathsf{s}(x), \mathsf{s}(y), z) \to \text{Com}_1(\mathsf{q}^\sharp(x, y, z))$

$Pre(\ (2)\ ) = \{(1)\}$

$(2)\ \mathsf{q}^\sharp(x, 0, \mathsf{s}(z)) \to \text{Com}_1(\mathsf{q}^\sharp(x, \mathsf{s}(z), \mathsf{s}(z)))$

Knowledge Propagation Processor

Proof Chain: $P_0 \overset{c_1}{\rightsquigarrow} \ldots \overset{c_k}{\rightsquigarrow} P_k$ $\qquad \iota_{\mathcal{R}} \leq \iota_{P_0} \leq \max(c_1, \ldots, c_k)$

$\mathcal{R}: \mathsf{q}(0, \mathsf{s}(y), \mathsf{s}(z)) \to 0, \;\; \mathsf{q}(\mathsf{s}(x), \mathsf{s}(y), z) \to \mathsf{q}(x, y, z), \;\; \mathsf{q}(x, 0, \mathsf{s}(z)) \to \mathsf{s}(\mathsf{q}(x, \mathsf{s}(z), \mathsf{s}(z)))$

Example: $\langle \mathcal{D}, \mathcal{D}, \varnothing, \mathcal{R} \rangle \overset{\mathcal{P}ol_0}{\rightsquigarrow} \langle \mathcal{D}', \mathcal{D}', \varnothing, \mathcal{R} \rangle \overset{\mathcal{P}ol_0}{\rightsquigarrow} \langle \mathcal{D}', \mathcal{D}', \varnothing, \varnothing \rangle$

$\overset{\mathcal{P}ol_1}{\rightsquigarrow} \langle \mathcal{D}', \{(2)\}, \{(1)\}, \varnothing \rangle \overset{\mathcal{P}ol_0}{\rightsquigarrow} \langle \mathcal{D}', \varnothing, \{(1), (2)\}, \varnothing \rangle$

$$\iota_{\mathcal{R}} \leq \max(\mathcal{P}ol_0, \mathcal{P}ol_0, \mathcal{P}ol_1, \mathcal{P}ol_0) = \mathcal{P}ol_1$$

Narrowing Processor

Narrowing Processor: $\langle\mathcal{D},\mathcal{S},\mathcal{K},\mathcal{R}\rangle \overset{\mathcal{P}ol_0}{\leadsto} \langle\mathcal{D}',\mathcal{S}',\mathcal{K}',\mathcal{R}\rangle$ where in $\mathcal{D}'$, $\mathcal{S}'$, some $w \to t$ is replaced by all its narrowings

$\langle\mathcal{D},\mathcal{D},\varnothing,\mathcal{R}\rangle$

$$
\begin{array}{llrcl rcl rcl}
\mathcal{R} & : & \mathsf{m}(x,y) & \to & \mathsf{if}(\mathsf{gt}(x,y),x,y) & \mathsf{gt}(0,k) & \to & \mathsf{false} & \mathsf{p}(0) & \to & 0 \\
 & & \mathsf{if}(\mathsf{false},x,y) & \to & 0 & \mathsf{gt}(\mathsf{s}(n),0) & \to & \mathsf{true} & \mathsf{p}(\mathsf{s}(n)) & \to & n \\
 & & \mathsf{if}(\mathsf{true},x,y) & \to & \mathsf{s}(\mathsf{m}(\mathsf{p}(x),y)) & \mathsf{gt}(\mathsf{s}(n),\mathsf{s}(k)) & \to & \mathsf{gt}(n,k) & & &
\end{array}
$$

Narrowing Processor

Narrowing Processor: $\langle \mathcal{D}, \mathcal{S}, \mathcal{K}, \mathcal{R} \rangle \overset{\mathcal{P}ol_0}{\rightsquigarrow} \langle \mathcal{D}', \mathcal{S}', \mathcal{K}', \mathcal{R} \rangle$ where
in $\mathcal{D}'$, $\mathcal{S}'$, some $w \to t$ is replaced by all its narrowings

Narrowings of $\mathsf{m}^\sharp(x, y) \to \textsc{Com}_2(\mathsf{if}^\sharp(\mathsf{gt}(x, y), x, y), \mathsf{gt}^\sharp(x, y))$

- $\mathsf{m}^\sharp(0, k) \to \textsc{Com}_2(\mathsf{if}^\sharp(\mathsf{false}, 0, k), \mathsf{gt}^\sharp(0, k))$
- $\mathsf{m}^\sharp(\mathsf{s}(n), 0) \to \textsc{Com}_2(\mathsf{if}^\sharp(\mathsf{true}, \mathsf{s}(n), 0), \mathsf{gt}^\sharp(\mathsf{s}(n), 0))$
- $\mathsf{m}^\sharp(\mathsf{s}(n), \mathsf{s}(k)) \to \textsc{Com}_2(\mathsf{if}^\sharp(\mathsf{gt}(n, k), \mathsf{s}(n), \mathsf{s}(k)), \mathsf{gt}^\sharp(\mathsf{s}(n), \mathsf{s}(k)))$

$\langle \mathcal{D}, \mathcal{D}, \varnothing, \mathcal{R} \rangle \overset{\mathcal{P}ol_0}{\rightsquigarrow}{}^* \langle \mathcal{D}_1, \mathcal{D}_1, \varnothing, \mathcal{R}_1 \rangle$

$\mathcal{R}_1$:

$$\begin{aligned} \mathsf{gt}(0, k) &\to \mathsf{false} & \mathsf{p}(0) &\to 0 \\ \mathsf{gt}(\mathsf{s}(n), 0) &\to \mathsf{true} & \mathsf{p}(\mathsf{s}(n)) &\to n \\ \mathsf{gt}(\mathsf{s}(n), \mathsf{s}(k)) &\to \mathsf{gt}(n, k) \end{aligned}$$

$\mathcal{D}_1$: $\mathsf{m}^\sharp(x, y) \to \textsc{Com}_2(\mathsf{if}^\sharp(\mathsf{gt}(x, y), x, y), \mathsf{gt}^\sharp(x, y))$

$\mathsf{if}^\sharp(\mathsf{true}, x, y) \to \textsc{Com}_2(\mathsf{m}^\sharp(\mathsf{p}(x), y), \mathsf{p}^\sharp(x))$ $\qquad \mathsf{gt}^\sharp(\mathsf{s}(n), \mathsf{s}(k)) \to \textsc{Com}_1(\mathsf{gt}^\sharp(n, k))$

Narrowing Processor

Narrowing Processor: $\langle \mathcal{D}, \mathcal{S}, \mathcal{K}, \mathcal{R} \rangle \overset{\mathcal{P}ol_0}{\leadsto} \langle \mathcal{D}', \mathcal{S}', \mathcal{K}', \mathcal{R} \rangle$ where
in $\mathcal{D}'$, $\mathcal{S}'$, some $w \to t$ is replaced by all its narrowings

Narrowings of $\mathsf{m}^\sharp(x, y) \to \textsc{Com}_2(\mathsf{if}^\sharp(\mathsf{gt}(x, y), x, y), \mathsf{gt}^\sharp(x, y))$

- $\mathsf{m}^\sharp(\mathsf{s}(n), 0) \to \textsc{Com}_2(\mathsf{if}^\sharp(\mathsf{true}, \mathsf{s}(n), 0), \mathsf{gt}^\sharp(\mathsf{s}(n), 0))$
- $\mathsf{m}^\sharp(\mathsf{s}(n), \mathsf{s}(k)) \to \textsc{Com}_2(\mathsf{if}^\sharp(\mathsf{gt}(n, k), \mathsf{s}(n), \mathsf{s}(k)), \mathsf{gt}^\sharp(\mathsf{s}(n), \mathsf{s}(k)))$

$\langle \mathcal{D}, \mathcal{D}, \varnothing, \mathcal{R} \rangle \overset{\mathcal{P}ol_0}{\leadsto}{}^* \langle \mathcal{D}_1, \mathcal{D}_1, \varnothing, \mathcal{R}_1 \rangle \overset{\mathcal{P}ol_0}{\leadsto}{}^* \langle \mathcal{D}_2, \mathcal{D}_2, \varnothing, \mathcal{R}_1 \rangle$

$\mathcal{R}_1$:

$$\begin{aligned} \mathsf{gt}(0, k) &\to \mathsf{false} & \mathsf{p}(0) &\to 0 \\ \mathsf{gt}(\mathsf{s}(n), 0) &\to \mathsf{true} & \mathsf{p}(\mathsf{s}(n)) &\to n \\ \mathsf{gt}(\mathsf{s}(n), \mathsf{s}(k)) &\to \mathsf{gt}(n, k) \end{aligned}$$

$\mathcal{D}_2$: $\mathsf{m}^\sharp(\mathsf{s}(n), 0) \to \textsc{Com}_2(\mathsf{if}^\sharp(\mathsf{true}, \mathsf{s}(n), 0), \mathsf{gt}^\sharp(\mathsf{s}(n), 0))$
$\mathsf{m}^\sharp(\mathsf{s}(n), \mathsf{s}(k)) \to \textsc{Com}_2(\mathsf{if}^\sharp(\mathsf{gt}(n, k), \mathsf{s}(n), \mathsf{s}(k)), \mathsf{gt}^\sharp(\mathsf{s}(n), \mathsf{s}(k)))$
$\mathsf{if}^\sharp(\mathsf{true}, x, y) \to \textsc{Com}_2(\mathsf{m}^\sharp(\mathsf{p}(x), y), \mathsf{p}^\sharp(x))$ $\quad$ $\mathsf{gt}^\sharp(\mathsf{s}(n), \mathsf{s}(k)) \to \textsc{Com}_1(\mathsf{gt}^\sharp(n, k))$

Narrowing Processor

Narrowing Processor: $\langle \mathcal{D}, \mathcal{S}, \mathcal{K}, \mathcal{R} \rangle \overset{\mathcal{P}ol_0}{\rightsquigarrow} \langle \mathcal{D}', \mathcal{S}', \mathcal{K}', \mathcal{R} \rangle$ where in $\mathcal{D}'$, $\mathcal{S}'$, some $w \to t$ is replaced by all its narrowings

$$\langle \mathcal{D}, \mathcal{D}, \varnothing, \mathcal{R} \rangle \overset{\mathcal{P}ol_0}{\rightsquigarrow}^* \langle \mathcal{D}_1, \mathcal{D}_1, \varnothing, \mathcal{R}_1 \rangle \overset{\mathcal{P}ol_0}{\rightsquigarrow}^* \langle \mathcal{D}_2, \mathcal{D}_2, \varnothing, \mathcal{R}_1 \rangle \overset{\mathcal{P}ol_0}{\rightsquigarrow}^* \langle \mathcal{D}_3, \mathcal{D}_3, \varnothing, \mathcal{R}_2 \rangle$$

$\mathcal{R}_2$:

$$\begin{aligned} \mathsf{gt}(0, k) &\to \mathsf{false} \\ \mathsf{gt}(\mathsf{s}(n), 0) &\to \mathsf{true} \\ \mathsf{gt}(\mathsf{s}(n), \mathsf{s}(k)) &\to \mathsf{gt}(n, k) \end{aligned}$$

$\mathcal{D}_3$:

$$\begin{aligned} \mathsf{m}^\sharp(\mathsf{s}(n), 0) &\to \mathrm{COM}_2(\mathsf{if}^\sharp(\mathsf{true}, \mathsf{s}(n), 0), \mathsf{gt}^\sharp(\mathsf{s}(n), 0)) \\ \mathsf{m}^\sharp(\mathsf{s}(n), \mathsf{s}(k)) &\to \mathrm{COM}_2(\mathsf{if}^\sharp(\mathsf{gt}(n, k), \mathsf{s}(n), \mathsf{s}(k)), \mathsf{gt}^\sharp(\mathsf{s}(n), \mathsf{s}(k))) \\ \mathsf{if}^\sharp(\mathsf{true}, \mathsf{s}(n), y) &\to \mathrm{COM}_2(\mathsf{m}^\sharp(n, y), \mathsf{p}^\sharp(\mathsf{s}(n))) \end{aligned}$$

$$\mathsf{gt}^\sharp(\mathsf{s}(n), \mathsf{s}(k)) \to \mathrm{COM}_1(\mathsf{gt}^\sharp(n, k))$$

Narrowing Processor

Reduction Pair Processor: $\langle \mathcal{D}, \mathcal{S}, \mathcal{K}, \mathcal{R} \rangle \overset{\mathcal{P}ol_m}{\rightsquigarrow} \langle \mathcal{D}, \mathcal{S} \setminus \mathcal{D}_{\succ}, \mathcal{K} \cup \mathcal{D}_{\succ}, \mathcal{R} \rangle$ where m is the maximal degree of polynomials $[f^\sharp]$

Polynomial Order

- $[0] = [\mathsf{true}] = [\mathsf{false}] = [\mathsf{p}^\sharp](x) = 0, \quad [\mathsf{s}](x) = x + 2$
- $[\mathsf{gt}](x, y) = [\mathsf{gt}^\sharp](x, y) = x$
- $[\mathsf{m}^\sharp](x, y) = (x + 1)^2, \quad [\mathsf{if}^\sharp](x, y, z) = y^2$

$$\langle \mathcal{D}, \mathcal{D}, \varnothing, \mathcal{R} \rangle \overset{\mathcal{P}ol_0}{\rightsquigarrow}{}^* \langle \mathcal{D}_1, \mathcal{D}_1, \varnothing, \mathcal{R}_1 \rangle \overset{\mathcal{P}ol_0}{\rightsquigarrow}{}^* \langle \mathcal{D}_2, \mathcal{D}_2, \varnothing, \mathcal{R}_1 \rangle \overset{\mathcal{P}ol_0}{\rightsquigarrow}{}^* \langle \mathcal{D}_3, \mathcal{D}_3, \varnothing, \mathcal{R}_2 \rangle$$

$$\overset{\mathcal{P}ol_2}{\rightsquigarrow} \langle \mathcal{D}_3, \varnothing, \mathcal{D}_3, \mathcal{R}_2 \rangle$$

$\mathcal{R}_2$:

$$\begin{aligned} \mathsf{gt}(0, k) &\succsim \mathsf{false} \\ \mathsf{gt}(\mathsf{s}(n), 0) &\succsim \mathsf{true} \\ \mathsf{gt}(\mathsf{s}(n), \mathsf{s}(k)) &\succsim \mathsf{gt}(n, k) \end{aligned}$$

$\mathcal{D}_3$:

$$\begin{aligned} \mathsf{m}^\sharp(\mathsf{s}(n), 0) &\succ \mathrm{COM}_2(\mathsf{if}^\sharp(\mathsf{true}, \mathsf{s}(n), 0), \mathsf{gt}^\sharp(\mathsf{s}(n), 0)) \\ \mathsf{m}^\sharp(\mathsf{s}(n), \mathsf{s}(k)) &\succ \mathrm{COM}_2(\mathsf{if}^\sharp(\mathsf{gt}(n, k), \mathsf{s}(n), \mathsf{s}(k)), \mathsf{gt}^\sharp(\mathsf{s}(n), \mathsf{s}(k))) \\ \mathsf{if}^\sharp(\mathsf{true}, \mathsf{s}(n), y) &\succ \mathrm{COM}_2(\mathsf{m}^\sharp(\mathsf{s}(n), y), \mathsf{p}^\sharp(\mathsf{s}(n))) \quad \mathsf{gt}^\sharp(\mathsf{s}(n), \mathsf{s}(k)) \succ \mathrm{COM}_1(\mathsf{gt}^\sharp(n, k)) \end{aligned}$$

Narrowing Processor

$$\langle \mathcal{D}, \mathcal{D}, \varnothing, \mathcal{R} \rangle \quad \overset{\mathcal{P}ol_0}{\rightsquigarrow}^* \langle \mathcal{D}_1, \mathcal{D}_1, \varnothing, \mathcal{R}_1 \rangle \overset{\mathcal{P}ol_0}{\rightsquigarrow}^* \langle \mathcal{D}_2, \mathcal{D}_2, \varnothing, \mathcal{R}_1 \rangle \overset{\mathcal{P}ol_0}{\rightsquigarrow}^* \langle \mathcal{D}_3, \mathcal{D}_3, \varnothing, \mathcal{R}_2 \rangle$$
$$\overset{\mathcal{P}ol_2}{\rightsquigarrow} \langle \mathcal{D}_3, \varnothing, \mathcal{D}_3, \mathcal{R}_2 \rangle$$

$$\iota_{\mathcal{R}} \leq \max(\mathcal{P}ol_0, \ldots, \mathcal{P}ol_0, \mathcal{P}ol_2) = \mathcal{P}ol_2$$

DT Framework for Innermost Complexity Analysis

- *Direct* adaption of DP framework for termination analysis
- *Modular* combination of different techniques
- Experiments on 1323 TRSs from *Termination Problem Data Base*

AProVE:	618 examples with polynomial runtime
CaT:	447 examples with polynomial runtime
TCT:	385 examples with polynomial runtime

		CaT					
		$\mathcal{P}ol_0$	$\mathcal{P}ol_1$	$\mathcal{P}ol_2$	$\mathcal{P}ol_3$	no result	$\sum$
AProVE	$\mathcal{P}ol_0$	-	182	-	-	27	209
	$\mathcal{P}ol_1$	-	187	7	-	76	270
	$\mathcal{P}ol_2$	-	32	2	-	83	117
	$\mathcal{P}ol_3$	-	6	-	-	16	22
	no result	-	27	3	1	674	705
	$\sum$	0	434	12	1	876	1323

DT Framework for Innermost Complexity Analysis

- *Direct* adaption of DP framework for termination analysis
- *Modular* combination of different techniques
- Experiments on 1323 TRSs from *Termination Problem Data Base*

AProVE:	618 examples with polynomial runtime
CaT:	447 examples with polynomial runtime
TCT:	385 examples with polynomial runtime

		TCT					
		$\mathcal{P}ol_0$	$\mathcal{P}ol_1$	$\mathcal{P}ol_2$	$\mathcal{P}ol_3$	no result	$\sum$
AProVE	$\mathcal{P}ol_0$	10	157	-	-	42	209
	$\mathcal{P}ol_1$	-	152	1	-	117	270
	$\mathcal{P}ol_2$	-	35	-	-	82	117
	$\mathcal{P}ol_3$	-	5	-	-	17	22
	no result	-	22	3	-	680	705
	$\sum$	10	371	4	0	938	1323

Overview

I. Termination of Term Rewriting

1. Termination of Term Rewrite Systems
2. Non-Termination of Term Rewrite Systems
3. Complexity of Term Rewrite Systems
4. Termination of Integer Term Rewrite Systems (RTA '09)

II. Termination of Programs

1. Termination of Functional Programs (Haskell)
2. Termination of Logic Programs (Prolog)
3. Termination of Imperative Programs (Java)

Termination of Programs

- direct approaches (e.g., Terminator for C-programs)
  - powerful for pre-defined data structures like integers
  - weak for algorithms on user-defined data structures

- transformational approaches via term rewriting (e.g., AProVE for Haskell, Prolog, Java)
  - powerful for algorithms on user-defined data structures (automatic generation of orders to compare arbitrary terms)
  - naive handling of pre-defined data structures (represent data objects by terms)

Representing integers

$$
\begin{aligned}
0 &\equiv \mathsf{pos}(0) \equiv \mathsf{neg}(0)\\
1 &\equiv \mathsf{pos}(\mathsf{s}(0))\\
-1 &\equiv \mathsf{neg}(\mathsf{s}(0))\\
1000 &\equiv \mathsf{pos}(\mathsf{s}(\mathsf{s}(\ldots \mathsf{s}(0)\ldots)))
\end{aligned}
$$

Rules for pre-defined operations

$$
\begin{aligned}
\mathsf{pos}(x) + \mathsf{neg}(y) &\to \mathsf{minus}(x, y)\\
\mathsf{neg}(x) + \mathsf{pos}(y) &\to \mathsf{minus}(y, x)\\
\mathsf{pos}(x) + \mathsf{pos}(y) &\to \mathsf{pos}(\mathsf{plus}(x, y))\\
\mathsf{neg}(x) + \mathsf{neg}(y) &\to \mathsf{neg}(\mathsf{plus}(x, y))\\
\mathsf{minus}(x, 0) &\to \mathsf{pos}(x)\\
\mathsf{minus}(0, y) &\to \mathsf{neg}(y)\\
\mathsf{minus}(\mathsf{s}(x), \mathsf{s}(y)) &\to \mathsf{minus}(x, y)
\end{aligned}
$$

Termination of Programs

- direct approaches (e.g., Terminator for C-programs)
  - powerful for pre-defined data structures like integers
  - weak for algorithms on user-defined data structures

- transformational approaches via term rewriting
  (e.g., AProVE for Haskell, Prolog, Java)
  - powerful for algorithms on user-defined data structures
    (automatic generation of orders to compare arbitrary terms)
  - naive handling of pre-defined data structures
    (represent data objects by terms)

Goal:

- integrate pre-defined data structures like $\mathbb{Z}$ into term rewriting
- develop method to prove termination of integer TRSs
  $\Rightarrow$ adapt DP framework to ITRSs
- for algorithms on integers: as powerful as direct techniques
- for user-defined data structures: as powerful as DP framework

Integer Term Rewriting

- $\mathcal{F}_{int}$: pre-defined symbols
  - $\mathbb{Z} = \{0, 1, -1, 2, -2, \dots\}$
  - $\mathbb{B} = \{\mathsf{true}, \mathsf{false}\}$
  - $+, -, *, /, \%$
  - $>, \geq, <, \leq, ==, !=$
  - $\neg, \wedge, \vee, \Rightarrow, \Leftrightarrow$
- $\mathcal{PD}$: pre-defined rules

  $2 * 21 \to 42$ $\quad 42 \geq 23 \to \mathsf{true}$

  $\mathsf{true} \wedge \mathsf{false} \to \mathsf{false}$ $\quad \dots$

  $\Rightarrow$ pre-defined operations only evaluated if all arguments are from $\mathbb{Z}$ or $\mathbb{B}$

- ITRS $\mathcal{R}$: finite TRS
  - no pre-defined symbols except $\mathbb{Z}$ and $\mathbb{B}$ in lhs
  - lhs $\notin \mathbb{Z} \cup \mathbb{B}$
  - rewrite relation $\hookrightarrow_{\mathcal{R}}$ defined as $\xrightarrow{\mathsf{i}}_{\mathcal{R} \cup \mathcal{PD}}$

Example ITRS computing $\sum_{i=y}^{x} i$

$$
\begin{array}{rcl}
\mathsf{sum}(x, y) & \to & \mathsf{sif}(x \geq y, x, y) \\
\mathsf{sif}(\mathsf{true}, x, y) & \to & y + \mathsf{sum}(x, y + 1) \\
\mathsf{sif}(\mathsf{false}, x, y) & \to & 0
\end{array}
$$

Integer Term Rewriting

$$\begin{array}{lllll}
\underline{\mathsf{sum}(1,1)} & \hookrightarrow_{\mathcal{R}} & \mathsf{sif}(\underline{1 \geq 1},1,1) & \hookrightarrow_{\mathcal{R}} & \underline{\mathsf{sif}(\mathsf{true},1,1)} \\
 & \hookrightarrow_{\mathcal{R}} & 1 + \mathsf{sum}(1,\underline{1+1}) & \hookrightarrow_{\mathcal{R}} & 1 + \underline{\mathsf{sum}(1,2)} \\
 & \hookrightarrow_{\mathcal{R}} & 1 + \mathsf{sif}(\underline{1 \geq 2},1,2) & \hookrightarrow_{\mathcal{R}} & 1 + \underline{\mathsf{sif}(\mathsf{false},1,2)} \\
 & \hookrightarrow_{\mathcal{R}} & \underline{1+0} & \hookrightarrow_{\mathcal{R}} & 1
\end{array}$$

- ITRS $\mathcal{R}$: finite TRS
  - no pre-defined symbols except $\mathbb{Z}$ and $\mathbb{B}$ in lhs
  - lhs $\notin \mathbb{Z} \cup \mathbb{B}$
  - rewrite relation $\hookrightarrow_{\mathcal{R}}$ defined as $\xrightarrow{\mathrm{i}}_{\mathcal{R} \cup \mathcal{PD}}$

Example ITRS computing $\sum_{i=y}^{x} i$

$$\begin{array}{rcl}
\mathsf{sum}(x,y) & \to & \mathsf{sif}(x \geq y, x, y) \\
\mathsf{sif}(\mathsf{true},x,y) & \to & y + \mathsf{sum}(x, y+1) \\
\mathsf{sif}(\mathsf{false},x,y) & \to & 0
\end{array}$$

Integer Term Rewriting

- Goal: prove innermost termination of $\mathcal{R} \cup \mathcal{PD}$ automatically

  Problem: $\mathcal{PD}$ is infinite

  Solution: handle $\mathcal{PD}$ implicitly by integrating it into the processors of the DP framework

- ITRS $\mathcal{R}$: finite TRS
  - no pre-defined symbols except $\mathbb{Z}$ and $\mathbb{B}$ in lhs
  - lhs $\notin \mathbb{Z} \cup \mathbb{B}$
  - rewrite relation $\hookrightarrow_{\mathcal{R}}$ defined as $\stackrel{\mathsf{i}}{\to}_{\mathcal{R} \cup \mathcal{PD}}$

Example ITRS computing $\sum_{i=y}^{x} i$

$$\begin{array}{rcl} \mathsf{sum}(x, y) & \to & \mathsf{sif}(x \geq y, x, y) \\ \mathsf{sif}(\mathsf{true}, x, y) & \to & y + \mathsf{sum}(x, y + 1) \\ \mathsf{sif}(\mathsf{false}, x, y) & \to & 0 \end{array}$$

Integer Dependency Pair Framework

- Defined symbols of TRS $\mathcal{R}$: roots of left-hand sides of $\mathcal{R}$

- Dependency pairs of TRS $\mathcal{R}$:
  if $f(s_1, \ldots, s_n) \to \ldots g(t_1, \ldots, t_m) \ldots \in \mathcal{R}$ and g is defined,
  then $F(s_1, \ldots, s_n) \to G(t_1, \ldots, t_m) \in DP(\mathcal{R})$

Example TRS

$$\begin{aligned} \mathsf{sum}(x, y) &\to \mathsf{sif}(x \geq y, x, y) \\ \mathsf{sif}(\mathsf{true}, x, y) &\to y + \mathsf{sum}(x, y + 1) \\ \mathsf{sif}(\mathsf{false}, x, y) &\to 0 \end{aligned}$$

DPs

$$\begin{aligned} \mathsf{SUM}(x, y) &\to \mathsf{SIF}(x \geq y, x, y) \\ \mathsf{SIF}(\mathsf{true}, x, y) &\to \mathsf{SUM}(x, y + 1) \end{aligned}$$

- $\mathcal{P}$-chain for DPs $\mathcal{P}$ and TRS $\mathcal{R}$: $s_1 \to t_1$, $s_2 \to t_2, \ldots$ where
  - $s_i \to t_i \in \mathcal{P}$
  - $t_i\sigma \xrightarrow{\mathrm{i}}{}^{*}_{\mathcal{R}} s_{i+1}\sigma$
  - $s_i\sigma$ in normal form w.r.t. $\xrightarrow{\mathrm{i}}_{\mathcal{R}}$

- Theorem TRS $\mathcal{R}$ terminating iff there is no infinite $DP(\mathcal{R})$-chain

Integer Dependency Pair Framework

- Defined symbols of ITRS $\mathcal{R}$: roots of left-hand sides of $\mathcal{R} \cup \mathcal{PD}$ including $+, -, >, \geq, \neg, \wedge, \vee, \ldots$
- Dependency pairs of ITRS $\mathcal{R}$:
  if $f(s_1, \ldots, s_n) \to \ldots g(t_1, \ldots, t_m) \ldots \in \mathcal{R}$ and $g \notin \mathcal{F}_{int}$ is defined,
  then $F(s_1, \ldots, s_n) \to G(t_1, \ldots, t_m) \in DP(\mathcal{R})$

Example ITRS

$$\begin{aligned} \mathsf{sum}(x,y) &\to \mathsf{sif}(x \geq y, x, y) \\ \mathsf{sif}(\mathsf{true}, x, y) &\to y + \mathsf{sum}(x, y+1) \\ \mathsf{sif}(\mathsf{false}, x, y) &\to 0 \end{aligned}$$

DPs

$$\begin{aligned} \mathsf{SUM}(x,y) &\to \mathsf{SIF}(x \geq y, x, y) \\ \mathsf{SIF}(\mathsf{true}, x, y) &\to \mathsf{SUM}(x, y+1) \end{aligned}$$

- $\mathcal{P}$-chain for DPs $\mathcal{P}$ and ITRS $\mathcal{R}$: $s_1 \to t_1$, $s_2 \to t_2, \ldots$ where
  - $s_i \to t_i \in \mathcal{P}$
  - $t_i\sigma \hookrightarrow^*_{\mathcal{R}} s_{i+1}\sigma$
  - $s_i\sigma$ in normal form w.r.t. $\hookrightarrow_{\mathcal{R}}$
- Theorem ITRS $\mathcal{R}$ terminating iff there is no infinite $DP(\mathcal{R})$-chain

Integer Dependency Pair Framework

Chain

$\mathrm{SUM}(x, y) \to \mathrm{SIF}(x \geq y, x, y), \quad \mathrm{SIF}(\mathrm{true}, x, y) \to \mathrm{SUM}(x, y+1)$ is chain for $\sigma(x) = \sigma(y) = 1$

$$\mathrm{SIF}(1 \geq 1, 1, 1) \hookrightarrow^*_{\mathcal{R}} \mathrm{SIF}(\mathrm{true}, 1, 1)$$

Example ITRS

$$\mathrm{sum}(x, y) \to \mathrm{sif}(x \geq y, x, y)$$
$$\mathrm{sif}(\mathrm{true}, x, y) \to y + \mathrm{sum}(x, y+1)$$
$$\mathrm{sif}(\mathrm{false}, x, y) \to 0$$

DPs

$$\mathrm{SUM}(x, y) \to \mathrm{SIF}(x \geq y, x, y)$$
$$\mathrm{SIF}(\mathrm{true}, x, y) \to \mathrm{SUM}(x, y+1)$$

- $\mathcal{P}$-chain for DPs $\mathcal{P}$ and ITRS $\mathcal{R}$: $s_1 \to t_1$, $s_2 \to t_2, \ldots$ where
  - $s_i \to t_i \in \mathcal{P}$
  - $t_i\sigma \hookrightarrow^*_{\mathcal{R}} s_{i+1}\sigma$
  - $s_i\sigma$ in normal form w.r.t. $\hookrightarrow_{\mathcal{R}}$
- Theorem ITRS $\mathcal{R}$ terminating iff there is no infinite $DP(\mathcal{R})$-chain

Integer Dependency Pair Framework

- DP processors $Proc(\mathcal{P}) = \{\mathcal{P}_1, \ldots, \mathcal{P}_n\}$
  - transform problem into simpler sub-problems
  - soundness: if there are no infinite $\mathcal{P}_1$-, ..., $\mathcal{P}_n$-chains,
    then there is no infinite $\mathcal{P}$-chain
  - start with initial problem $DP(\mathcal{R})$,
    apply processors repeatedly until all problems are solved

- numerous DP processors developed for TRSs
  - many processors only rely on DPs and on defined symbols
    $\Rightarrow$ adaption to ITRSs straightforward
  - reduction pair processor relies on DPs and on *rules*
    $\Rightarrow$ adaption to ITRSs problematic, since $\mathcal{PD}$ has infinitely many rules

Goal: adapt reduction pair processor to ITRSs

Reduction Pair Processor

Pre-defined rules $\mathcal{PD}$ and ITRS $\mathcal{R}$

$$\begin{aligned} 0+0 &\to 0 \\ -1+3 &\to 2 \\ &\dots \\ \mathsf{sum}(x,y) &\to \mathsf{sif}(x \geq y, x, y) \\ \mathsf{sif}(\mathsf{true},x,y) &\to y + \mathsf{sum}(x, y+1) \\ \mathsf{sif}(\mathsf{false},x,y) &\to 0 \end{aligned}$$

DPs $\mathcal{P}$

$$\begin{aligned} \mathsf{SUM}(x,y) &\to \mathsf{SIF}(x \geq y, x, y) \\ \mathsf{SIF}(\mathsf{true},x,y) &\to \mathsf{SUM}(x, y+1) \end{aligned}$$

- Generate constraints such that infinite chain leads to infinite decrease

$$\begin{array}{ccccccccccccc} s_1 & \to & t_1, & & s_2 & \to & t_2, & & s_3 & \to & t_3, & & \dots \\ s_1 & \underset{(\sim)}{\succ} & t_1 & \succsim & s_2 & \underset{(\sim)}{\succ} & t_2 & \succsim & s_3 & \underset{(\sim)}{\succ} & t_3 & \succsim & \dots \end{array}$$

- Reduction pair processor: $Proc(\mathcal{P}) = \{\mathcal{P} \setminus \succ\}$ if
  - $\ell \succsim r$ for all usable rules $\ell \to r$ in $\mathcal{R} \cup \mathcal{PD}$
  - $s \succ t$ or $s \succsim t$ for all $s \to t$ in $\mathcal{P}$

- Usable rules: rules that can reduce terms of $\mathcal{P}$'s right-hand sides when instantiating their variables with normal forms

Reduction Pair Processor

Usable rules in $\mathcal{R} \cup \mathcal{PD}$

$$\begin{array}{rcl} 0+0 & \succsim & 0 \\ -1+3 & \succsim & 2 \\ & \dots & \end{array}$$

DPs $\mathcal{P}$

$$\begin{array}{rcl} \mathsf{SUM}(x,y) & \overset{(\succ)}{\succsim} & \mathsf{SIF}(x \geq y, x, y) \\ \mathsf{SIF}(\mathsf{true},x,y) & \overset{(\succ)}{\succsim} & \mathsf{SUM}(x, y+1) \\ \mathsf{SUM}(x,y) & \overset{(\succ)}{\succsim} & \mathsf{c} \\ \mathsf{SIF}(\mathsf{true},x,y) & \overset{(\succ)}{\succsim} & \mathsf{c} \end{array}$$

- Problem: infinitely many constraints $\ell \succsim r$ since $\mathcal{PD}$ is infinite
  Solution: consider $\mathcal{PD}$ implicitly $\Rightarrow$ fix interpretation for $\mathcal{F}_{int}$

- Reduction pair processor: $Proc(\mathcal{P}) = \{\mathcal{P} \setminus \succ, \mathcal{P} \setminus \mathcal{P}_{bound}\}$ if
  - $\ell \succsim r$ for all usable rules $\ell \to r$ in $\mathcal{R} \cup \mathcal{PD}$
  - $s \succ t$ or $s \succsim t$ for all $s \to t$ in $\mathcal{P}$

  where $\mathcal{P}_{bound} = \{s \to t \in \mathcal{P} \mid s \succsim \mathsf{c}\}$

- search for integer max-polynomial interpretation *Pol* satisfying constraints

Reduction Pair Processor

Usable rules in $\mathcal{R}$

DPs $\mathcal{P}$

$$
\begin{array}{rcl}
\mathsf{SUM}(x, y) & \overset{\succ}{(\succsim)} & \mathsf{SIF}(x \geq y, x, y) \\
\mathsf{SIF}(\mathsf{true}, x, y) & \overset{\succ}{(\succsim)} & \mathsf{SUM}(x, y + 1) \\
\mathsf{SUM}(x, y) & \overset{\succ}{(\succsim)} & \mathsf{c} \\
\mathsf{SIF}(\mathsf{true}, x, y) & \overset{\succ}{(\succsim)} & \mathsf{c}
\end{array}
$$

- choose $\mathsf{SUM}_{Pol}(x, y) = x - y$ $\quad \mathsf{SIF}_{Pol}(x, y, z) = y - z$

  But: $\mathsf{SUM}(x, y) \succsim \mathsf{c}$, $\mathsf{SIF}(\mathsf{true}, x, y) \succsim \mathsf{c}$ not satisfied for any c_{Pol}

- Reduction pair processor: $Proc(\mathcal{P}) = \{\mathcal{P} \setminus \succ, \mathcal{P} \setminus \mathcal{P}_{bound}\}$ if
  - $\ell \succsim r$ for all usable rules $\ell \to r$ in $\mathcal{R}$
  - $s \succ t$ or $s \succsim t$ for all $s \to t$ in $\mathcal{P}$

  where $\mathcal{P}_{bound} = \{s \to t \in \mathcal{P} \mid s \succsim \mathsf{c}\}$

- I-interpretation: $0_{Pol} = 0, \quad 1_{Pol} = 1, \quad -1_{Pol} = -1, \ \ldots,$
  $x +_{Pol} y = x + y, \quad x -_{Pol} y = x - y, \quad x *_{Pol} y = x * y, \ \ldots$

Reduction Pair Processor with Conditional Constraints

- weaken constraints: do not require them for *all* x and y,
  but only for those instantiations σ used in chains

- require $\mathsf{SUM}(x, y) \succsim \mathsf{c}$ only for σ where
  $\mathsf{SIF}(x \geq y, x, y)\sigma$ reduces to $\mathsf{SUM}(x', y')\sigma$ or $\mathsf{SIF}(\mathsf{true}, x', y')\sigma$

Constraints

$$\mathsf{SIF}(x \geq y, x, y) = \mathsf{SUM}(x', y') \;\Rightarrow\; \mathsf{SUM}(x, y) \succsim \mathsf{c}$$
$$\mathsf{SIF}(x \geq y, x, y) = \mathsf{SIF}(\mathsf{true}, x', y') \;\Rightarrow\; \mathsf{SUM}(x, y) \succsim \mathsf{c}$$

DPs

$$\mathsf{SUM}(x, y) \to \mathsf{SIF}(x \geq y, x, y)$$
$$\mathsf{SIF}(\mathsf{true}, x, y) \to \mathsf{SUM}(x, y + 1)$$

Reduction Pair Processor with Conditional Constraints

Rules to simplify conditional constraints

1. **Constructor and Different Function Symbol**

$$\frac{f(s_1, \ldots, s_n) = g(t_1, \ldots, t_m) \wedge \varphi \;\Rightarrow\; \psi}{TRUE} \qquad \text{if } f \text{ is a constructor and } f \neq g$$

Constraints

$$\mathsf{SIF}(x \geq y, x, y) = \mathsf{SIF}(\mathsf{true}, x', y') \;\Rightarrow\; \mathsf{SUM}(x, y) \;\succsim\; \mathsf{c}$$

Reduction Pair Processor with Conditional Constraints

Rules to simplify conditional constraints

1. **Constructor and Different Function Symbol** $$\frac{f(s_1, ..., s_n) = g(t_1, ..., t_m) \wedge \varphi \;\Rightarrow\; \psi}{\mathit{TRUE}}$$	if f is a constructor and $f \neq g$
2. **Same Constructors on Both Sides** $$\frac{f(s_1, ..., s_n) = f(t_1, ..., t_n) \wedge \varphi \;\Rightarrow\; \psi}{s_1 = t_1 \wedge \ldots \wedge s_n = t_n \wedge \varphi \;\Rightarrow\; \psi}$$	if f is a constructor

Constraints

$$x \geq y = \mathsf{true} \;\wedge\; x = x' \;\wedge\; y = y' \;\Rightarrow\; \mathsf{SUM}(x, y) \;\succsim\; \mathsf{c}$$

Reduction Pair Processor with Conditional Constraints

Rules to simplify conditional constraints

1. Constructor and Different Function Symbol

$$\frac{f(s_1, ..., s_n) = g(t_1, ..., t_m) \wedge \varphi \;\Rightarrow\; \psi}{TRUE}$$

if f is a constructor and $f \neq g$

2. Same Constructors on Both Sides

$$\frac{f(s_1, ..., s_n) = f(t_1, ..., t_n) \wedge \varphi \;\Rightarrow\; \psi}{s_1 = t_1 \wedge \ldots \wedge s_n = t_n \wedge \varphi \;\Rightarrow\; \psi}$$

if f is a constructor

3. Lift Pre-Defined Symbols

$$\frac{s \geq t = \text{true} \wedge \varphi \;\Rightarrow\; \psi}{(s \succsim t \quad \wedge \varphi \;\Rightarrow\; \psi) \wedge \ell \succsim r \text{ for all usable rules } \ell \to r}$$

Constraints

$x \succsim y \quad \Rightarrow \text{SUM}(x, y) \succsim \text{c}$

Reduction Pair Processor with Conditional Constraints

Constraints

$$
\begin{array}{rcrcl}
 & & \mathsf{SUM}(x,y) & \underset{(\sim)}{\succ} & \mathsf{SIF}(x \geq y, x, y) \\
 & & \mathsf{SIF}(\mathsf{true}, x, y) & \underset{(\sim)}{\succ} & \mathsf{SUM}(x, y+1) \\
\mathsf{SIF}(x \geq y, x, y) = \mathsf{SUM}(x', y') & \Rightarrow & \mathsf{SUM}(x,y) & \succsim & \mathsf{c} \\
\mathsf{SIF}(x \geq y, x, y) = \mathsf{SIF}(\mathsf{true}, x', y') & \Rightarrow & \mathsf{SUM}(x,y) & \succsim & \mathsf{c}
\end{array}
$$

- Reduction pair processor: $Proc(\mathcal{P}) = \{\mathcal{P} \setminus \succ, \mathcal{P} \setminus \mathcal{P}_{bound}\}$ if
  - $\ell \succsim r$ for all usable rules $\ell \to r$ in $\mathcal{R}$
  - $s \succ t$ or $s \succsim t$ for all $s \to t$ in $\mathcal{P}$

  where $\mathcal{P}_{bound} = \{s \to t \in \mathcal{P} \mid s \succsim \mathsf{c}\}$

Reduction Pair Processor with Conditional Constraints

Constraints

$$\begin{array}{rrcl} & \mathsf{SUM}(x,y) & \underset{(\sim)}{\succsim} & \mathsf{SIF}(x \geq y, x, y) \\ & \mathsf{SIF}(\mathsf{true}, x, y) & \underset{(\sim)}{\succsim} & \mathsf{SUM}(x, y+1) \\ & & & \\ x \succsim y \;\Rightarrow & \mathsf{SUM}(x,y) & \succsim & \mathsf{c} \end{array}$$

- Reduction pair processor: $Proc(\mathcal{P}) = \{\mathcal{P} \setminus \succ, \mathcal{P} \setminus \mathcal{P}_{bound}\}$ if
  - $\ell \succsim r$ for all usable rules $\ell \to r$ in $\mathcal{R}$
  - $s \succ t$ or $s \succsim t$ for all $s \to t$ in $\mathcal{P}$

  where $\mathcal{P}_{bound} = \{s \to t \in \mathcal{P} \mid s \succsim \mathsf{c}\}$

- I-interpretation: $\mathsf{SUM}_{Pol}(x,y) = x - y \quad \mathsf{SIF}_{Pol}(x,y,z) = y - z \quad \mathsf{c}_{Pol} = 0$

Reduction Pair Processor with Conditional Constraints

Constraints

$$\begin{array}{rrcl} & \mathsf{SUM}(x,y) & \underset{(\sim)}{\succ} & \mathsf{SIF}(x \geq y, x, y) \\ & \mathsf{SIF}(\mathsf{true}, x, y) & \succ & \mathsf{SUM}(x, y+1) \\ x \succsim y \;\Rightarrow & \mathsf{SUM}(x,y) & \succsim & \mathsf{c} \end{array}$$

- Reduction pair processor: $Proc(\mathcal{P}) = \{\mathcal{P} \setminus \succ, \mathcal{P} \setminus \mathcal{P}_{bound}\}$ if
  - $\ell \succsim r$ for all usable rules $\ell \to r$ in $\mathcal{R}$
  - $s \succ t$ or $s \succsim t$ for all $s \to t$ in $\mathcal{P}$

  where $\mathcal{P}_{bound} = \{s \to t \in \mathcal{P} \mid s \succsim \mathsf{c}\}$

- I-interpretation: $\mathsf{SUM}_{Pol}(x,y) = x - y \quad \mathsf{SIF}_{Pol}(x,y,z) = y - z \quad \mathsf{c}_{Pol} = 0$

- *Proc* transforms initial problem into two separate problems

$$\begin{array}{lrcl} \mathcal{P} \setminus \succ: & \mathsf{SUM}(x,y) & \to & \mathsf{SIF}(x \geq y, x, y) \\ \mathcal{P} \setminus \mathcal{P}_{bound}: & \mathsf{SIF}(\mathsf{true}, x, y) & \to & \mathsf{SUM}(x, y+1) \end{array}$$

$\Rightarrow$ both can easily be solved separately

$\Rightarrow$ termination easy if one can generate I-interpretation automatically

Generating I-Interpretations

- *abstract* I-interpretation: $\mathsf{SUM}_{Pol}(x, y) = a_0 + a_1\, x + a_2\, y \qquad \mathsf{c}_{Pol} = c_0$

Conditional constraint (without =)

$$x \gtrsim y \quad \Rightarrow \quad \mathsf{SUM}(x, y) \gtrsim \mathsf{c}$$

Generating I-Interpretations

- *abstract* I-interpretation: $\mathsf{SUM}_{Pol}(x, y) = a_0 + a_1\, x + a_2\, y \qquad \mathsf{c}_{Pol} = c_0$

Inequality constraint

$$\forall x \in \mathbb{Z},\ y \in \mathbb{Z} \quad (\ x \ \geq \ y \quad \Rightarrow \quad a_0 + a_1 \quad x \qquad + a_2\, y \ \geq \ c_0\ )$$

Generating I-Interpretations

Rules to simplify inequality constraints

Goal: remove $\forall$, $\mathbb{Z}$, conditions $\Rightarrow$ Diophantine constraints $\Rightarrow$ SAT solving

1. **Eliminate Conditions**

$$\frac{\forall x \in \mathbb{Z}, \ldots \quad (x \geq p \wedge \varphi \Rightarrow \psi)}{\forall z \in \mathbb{N}, \ldots \quad (\varphi[x/p+z] \Rightarrow \psi[x/p+z])}$$

if x does not occur in the polynomial p

Inequality constraint

$\forall y \in \mathbb{Z},\ z \in \mathbb{N}$ $\qquad a_0 + a_1(\ \ y + z) + a_2\, y \ \geq\ c_0$

Generating I-Interpretations

Rules to simplify inequality constraints

Goal: remove $\forall$, $\mathbb{Z}$, conditions $\Rightarrow$ Diophantine constraints $\Rightarrow$ SAT solving

1. Eliminate Conditions

$$\frac{\forall x \in \mathbb{Z}, \ldots \quad (x \geq p \wedge \varphi \Rightarrow \psi)}{\forall z \in \mathbb{N}, \ldots \quad (\varphi[x/p+z] \Rightarrow \psi[x/p+z])}$$

if x does not occur in the polynomial p

2. Split

$$\frac{\forall y \in \mathbb{Z} \quad \varphi}{\forall y \in \mathbb{N} \quad \varphi \qquad \wedge \qquad \forall y \in \mathbb{N} \quad \varphi[y/-y]}$$

Inequality constraint

$$\forall y \in \mathbb{N},\ z \in \mathbb{N} \qquad a_0 + a_1\,(\ \ y + z) + a_2\,y \ \geq\ c_0$$

$$\forall y \in \mathbb{N},\ z \in \mathbb{N} \qquad a_0 + a_1\,(-y + z) - a_2\,y \ \geq\ c_0$$

Generating I-Interpretations

Rules to simplify inequality constraints

Goal: remove $\forall$, $\mathbb{Z}$, conditions $\Rightarrow$ Diophantine constraints $\Rightarrow$ SAT solving

1. **Eliminate Conditions**

$$\frac{\forall x \in \mathbb{Z}, \ldots \quad (x \geq p \wedge \varphi \Rightarrow \psi)}{\forall z \in \mathbb{N}, \ldots \quad (\varphi[x/p+z] \Rightarrow \psi[x/p+z])}$$

if x does not occur in the polynomial p

2. **Split**

$$\frac{\forall y \in \mathbb{Z} \quad \varphi}{\forall y \in \mathbb{N} \quad \varphi \qquad \wedge \qquad \forall y \in \mathbb{N} \quad \varphi[y/-y]}$$

Inequality constraint

$$\forall y \in \mathbb{N},\ z \in \mathbb{N} \qquad (a_1 + a_2)\, y + a_1\, z + (a_0 - c_0) \ \geq\ 0$$

Generating I-Interpretations

Rules to simplify inequality constraints

Goal: remove $\forall$, $\mathbb{Z}$, conditions $\Rightarrow$ Diophantine constraints $\Rightarrow$ SAT solving

1. Eliminate Conditions

$$\frac{\forall x \in \mathbb{Z}, \ldots \quad (x \geq p \wedge \varphi \Rightarrow \psi)}{\forall z \in \mathbb{N}, \ldots \quad (\varphi[x/p+z] \Rightarrow \psi[x/p+z])}$$

if x does not occur in the polynomial p

2. Split

$$\frac{\forall y \in \mathbb{Z} \quad \varphi}{\forall y \in \mathbb{N} \quad \varphi \qquad \wedge \qquad \forall y \in \mathbb{N} \quad \varphi[y/-y]}$$

3. Eliminate Universally Quantified Variables

$$\frac{\forall x_i \in \mathbb{N} \quad p_1\, x_1^{e_{11}} \ldots x_m^{e_{m1}} + \ldots + p_k\, x_1^{e_{1k}} \ldots x_m^{e_{mk}} \geq 0}{p_1 \geq 0 \ \wedge \ldots \wedge \ p_k \geq 0}$$

if the p_j do not contain variables

Inequality constraint

$$a_1 + a_2 \geq 0 \quad \wedge \quad a_1 \geq 0 \quad \wedge \quad a_0 - c_0 \geq 0$$

Solution: $a_0 = 0 \quad a_1 = 1 \quad a_2 = -1 \quad c_0 = 0$

Generating I-Interpretations

abstract I-interpretation: $\mathsf{SUM}_{Pol}(x, y) = a_0 + a_1\, x + a_2\, y$ $\quad \mathsf{c}_{Pol} = c_0$

actual I-interpretation: $\mathsf{SUM}_{Pol}(x, y) = x - y$ $\quad \mathsf{c}_{Pol} = 0$

Inequality constraint

$$a_1 + a_2 \geq 0 \quad \wedge \quad a_1 \geq 0 \quad \wedge \quad a_0 - c_0 \geq 0$$

Solution: $a_0 = 0 \quad a_1 = 1 \quad a_2 = -1 \quad c_0 = 0$

Proving Termination of Integer Term Rewriting

- ITRSs: TRSs with built-in integers, adapted DP framework to ITRSs
- also suitable for ITRSs with large numbers
  $\mathsf{f}(\mathsf{true},\ x) \ \to\ \mathsf{f}(\mathsf{ack}(10,10) \geq x,\ x+1)$
- implemented in AProVE and evaluated on collection of 117 ITRSs
  - examples from TPDB and rewriting papers adapted to integers
  - examples from termination of imperative programming

	YES	MAYBE	TIMEOUT
AProVE Integer	104 (avg. 4.9 s)	0	13
AProVE old	24 (avg. 7.2 s)	6 (avg. 0.9 s)	87
$\mathsf{T_TT_2}$	6 (avg. 3.6 s)	110 (avg. 4.9 s)	1

$\Rightarrow$ enormous benefit of built-in integers