ISR 2026 — Basic Track

Jörg Endrullis
Cynthia Kop
Framke van Raamsdonk

partially based on slides by: Aart Middeldorp

hosting institution:
Radboud Universiteit Nijmegen
The Netherlands

- Part 1-2: Abstract Rewriting
- Part 3: Term Rewriting
- Part 5: Termination: Monotonic Algebras
- Part 6: Termination: Recursive Path Ordering
- Part 7: Confluence: Unification and Critical Pairs
- Part 9-10: Weak and Strong Normalization for Typed Lambda Calculus
- Part 11: Equational Reasoning and Completion
- Part 13: Termination: Dependency Pair Framework
- Part 15: Confluence: Orthogonality
- Part 16: Confluence: Decreasing Diagrams
- Part 17: Advanced Topics

Outline

- Overview
- Motivation
- Dependency pairs and chains: definition
- Using a reduction pair
- A modular framework
- Other processors

Recall: subterm property

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

Recall: subterm property

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

If $\mathsf{minus}(x, y) \succ y$ and $\mathsf{s}(a) \succ a$, then:

$$\mathsf{s}(\mathsf{quot}(\underline{\mathsf{minus}(x, y)}, \mathsf{s}(y))) \quad \succ \quad \mathsf{s}(\mathsf{quot}(\underline{y}, \mathsf{s}(y)))$$

Recall: subterm property

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

If $\mathsf{minus}(x, y) \succ y$ and $\mathsf{s}(a) \succ a$, then:

$$\begin{array}{rcl} \mathsf{s}(\mathsf{quot}(\mathsf{minus}(x, y), \mathsf{s}(y))) & \succ & \mathsf{s}(\mathsf{quot}(y, \mathsf{s}(y))) \\ & \succ & \mathsf{quot}(y, \mathsf{s}(y)) \end{array}$$

Recall: subterm property

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

If $\mathsf{minus}(x, y) \succ y$ and $\mathsf{s}(a) \succ a$, then:

$$\begin{array}{rcl} \mathsf{s}(\mathsf{quot}(\mathsf{minus}(x, y), \mathsf{s}(y))) & \succ & \mathsf{s}(\mathsf{quot}(y, \mathsf{s}(y))) \\ & \succ & \mathsf{quot}(y, \mathsf{s}(y)) \end{array}$$

Definition (Subterm Property)

A reduction ordering $\succ$ has the subterm property if $\mathsf{f}(\dots, s, \dots) \succeq s$ for all f.

Secondary motivation

Program analysis: often hundreds or thousands of rules.

Secondary motivation

Program analysis: often hundreds or thousands of rules.

Problem: finding an ordering for all at once is computationally difficult.

Secondary motivation

Program analysis: often hundreds or thousands of rules.

Problem: finding an ordering for all at once is computationally difficult.

Wish: split a termination problem into multiple smaller problems.

The dependency pair framework

Idea: a general framework for termination analysis:

The dependency pair framework

Idea: a general framework for termination analysis:

- in principle applicable to all TRSs; no subterm property limitation

The dependency pair framework

Idea: a general framework for termination analysis:

- in principle applicable to all TRSs; no subterm property limitation
- several building blocks, including reduction orderings

The dependency pair framework

Idea: a general framework for termination analysis:

- in principle applicable to all TRSs; no subterm property limitation
- several building blocks, including reduction orderings
- both for termination and non-termination

The dependency pair framework

Idea: a general framework for termination analysis:

- in principle applicable to all TRSs; no subterm property limitation
- several building blocks, including reduction orderings
- both for termination and non-termination

The core idea of dependency pairs is to look at function calls.

The dependency pair framework

Idea: a general framework for termination analysis:

- in principle applicable to all TRSs; no subterm property limitation
- several building blocks, including reduction orderings
- both for termination and non-termination

The core idea of dependency pairs is to look at function calls.

To start, split functions in constructors and defined symbols.

Identifying function calls

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

Identifying function calls

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

We isolate the calls from one defined symbol to another:

$$\begin{aligned}
\mathsf{minus}^\sharp(\mathsf{s}(x), \mathsf{s}(y)) &\Rightarrow \mathsf{minus}^\sharp(x, y) \\
\mathsf{quot}^\sharp(\mathsf{s}(x), \mathsf{s}(y)) &\Rightarrow \mathsf{quot}^\sharp(\mathsf{minus}(x, y), \mathsf{s}(y)) \\
\mathsf{quot}^\sharp(\mathsf{s}(x), \mathsf{s}(y)) &\Rightarrow \mathsf{minus}^\sharp(x, y)
\end{aligned}$$

Dependency pair chains

We can prove the following result:

Theorem (Dependency Pair Chains)

For a given set of rules R, let DP be the corresponding set of dependency pairs.

$\to_R$ is terminating if and only if there is no infinite (DP, R)-chain: a reduction $s_1 \Rightarrow_{DP} \to_R^* s_2 \Rightarrow_{DP} \to_R^* s_3 \ldots$

Dependency pair chains

We can prove the following result:

Theorem (Dependency Pair Chains)

For a given set of rules R, let DP be the corresponding set of dependency pairs.

$\to_R$ is terminating if and only if there is no infinite (DP, R)-chain: a reduction $s_1 \Rightarrow_{DP} \to_R^* s_2 \Rightarrow_{DP} \to_R^* s_3 \ldots$

Note: marking prevents $\to_R$ steps at the top!

Dependency pair chains

We can prove the following result:

Theorem (Dependency Pair Chains)

For a given set of rules R, let DP be the corresponding set of dependency pairs.

$\to_R$ is terminating if and only if there is no infinite (DP, R)-chain: a reduction $s_1 \Rightarrow_{DP} \to_R^* s_2 \Rightarrow_{DP} \to_R^* s_3 \ldots$

Note: marking prevents $\to_R$ steps at the top!

So $\to_R$ is terminating iff there is no infinite sequence where:

- the steps using $\Rightarrow_{\mathrm{DP}}$ occur at the root of the term;
- the steps using $\to_R$ do not occur at the root of the term;
- there are infinitely many steps using $\Rightarrow_{\mathrm{DP}}$.

Example: an infinite DP chain

$$R = \left\{ \begin{array}{rcl} \mathsf{f}(0, x) & \to & \mathsf{g}(\mathsf{f}(\mathsf{g}(x), x)) \\ \mathsf{g}(x) & \to & x \end{array} \right\}$$

Example: an infinite DP chain

$$R = \left\{ \begin{array}{rcl} \mathsf{f}(0, x) & \to & \mathsf{g}(\mathsf{f}(\mathsf{g}(x), x)) \\ \mathsf{g}(x) & \to & x \end{array} \right\}$$

$$\mathrm{DP} = \left\{ \begin{array}{rcl} \mathsf{f}^\sharp(0, x) & \Rightarrow & \mathsf{g}^\sharp(\mathsf{f}(\mathsf{g}(x), x)) \\ \mathsf{f}^\sharp(0, x) & \Rightarrow & \mathsf{f}^\sharp(\mathsf{g}(x), x) \\ \mathsf{f}^\sharp(0, x) & \Rightarrow & \mathsf{g}^\sharp(x) \end{array} \right\}$$

Example: an infinite DP chain

$$R = \left\{ \begin{array}{rcl} \mathsf{f}(0,x) & \to & \mathsf{g}(\mathsf{f}(\mathsf{g}(x),x)) \\ \mathsf{g}(x) & \to & x \end{array} \right\}$$

$$\mathrm{DP} = \left\{ \begin{array}{rcl} \mathsf{f}^\sharp(0,x) & \Rightarrow & \mathsf{g}^\sharp(\mathsf{f}(\mathsf{g}(x),x)) \\ \mathsf{f}^\sharp(0,x) & \Rightarrow & \mathsf{f}^\sharp(\mathsf{g}(x),x) \\ \mathsf{f}^\sharp(0,x) & \Rightarrow & \mathsf{g}^\sharp(x) \end{array} \right\}$$

Infinite chain: $\underline{\mathsf{f}^\sharp(0,0)} \Rightarrow_{\mathrm{DP}} \mathsf{f}^\sharp(\underline{\mathsf{g}(0)},0) \to_R \underline{\mathsf{f}^\sharp(0,0)} \Rightarrow_{\mathrm{DP}} \mathsf{f}^\sharp(\underline{\mathsf{g}(0)},0) \to_R \ldots$

Proving the dependency pair theorem

Theorem: $\to_R$ is terminating if and only if there is no infinite reduction $s_1 \Rightarrow_{\mathsf{DP}} \to_R^* s_2 \Rightarrow_{\mathsf{DP}} \to_R^* s_3 \ldots$

Proof ($\Longrightarrow$):

Proving the dependency pair theorem

Theorem: $\to_R$ is terminating if and only if there is no infinite reduction $s_1 \Rightarrow_{\mathsf{DP}} \to_R^* s_2 \Rightarrow_{\mathsf{DP}} \to_R^* s_3 \ldots$

Proof ($\Longrightarrow$):

- Suppose there is an infinite chain.

Proving the dependency pair theorem

Theorem: $\to_R$ is terminating if and only if there is no infinite reduction $s_1 \Rightarrow_{\mathrm{DP}} \to_R^* s_2 \Rightarrow_{\mathrm{DP}} \to_R^* s_3 \ldots$

Proof ($\Longrightarrow$):

- Suppose there is an infinite chain.
- Note that $s \Rightarrow_{\mathrm{DP}} t$ implies $s \to_R s' \trianglerighteq t$ for some s'.

Proving the dependency pair theorem

Theorem: $\to_R$ is terminating if and only if there is no infinite reduction $s_1 \Rightarrow_{\mathsf{DP}} \to_R^* s_2 \Rightarrow_{\mathsf{DP}} \to_R^* s_3 \dots$

Proof ($\Longrightarrow$):

- Suppose there is an infinite chain.
- Note that $s \Rightarrow_{\mathsf{DP}} t$ implies $s \to_R s' \trianglerighteq t$ for some s'.
- Hence, there is an infinite chain
  $s_1 \to_R C_1[t_1] \to_R^* C_1[s_2] \to_R C_1[C_2[t_2]] \to_R^* C_1[C_2[s_3]] \dots$

Proving the dependency pair theorem

Theorem: $\to_R$ is terminating if and only if there is no infinite reduction $s_1 \Rightarrow_{\mathsf{DP}} \to_R^* s_2 \Rightarrow_{\mathsf{DP}} \to_R^* s_3 \ldots$

Proof ($\Longleftarrow$):

Proving the dependency pair theorem

Theorem: $\to_R$ is terminating if and only if there is no infinite reduction $s_1 \Rightarrow_{\mathsf{DP}} \to_R^* s_2 \Rightarrow_{\mathsf{DP}} \to_R^* s_3 \ldots$

Proof ($\Longleftarrow$):

- Suppose $\to_R$ is non-terminating.

Proving the dependency pair theorem

Theorem: $\to_R$ is terminating if and only if there is no infinite reduction $s_1 \Rightarrow_{\mathsf{DP}} \to_R^* s_2 \Rightarrow_{\mathsf{DP}} \to_R^* s_3 \ldots$

Proof ($\Longleftarrow$):

- Suppose $\to_R$ is non-terminating.
- Then there is a minimal non-terminating term $\mathsf{f}(s_1, \ldots, s_n)$.

Proving the dependency pair theorem

Theorem: $\to_R$ is terminating if and only if there is no infinite reduction $s_1 \Rightarrow_{\mathsf{DP}} \to_R^* s_2 \Rightarrow_{\mathsf{DP}} \to_R^* s_3 \ldots$

Proof ($\Longleftarrow$):

- Suppose $\to_R$ is non-terminating.
- Then there is a minimal non-terminating term $\mathsf{f}(s_1, \ldots, s_n)$.
- Consider: what does an infinite reduction starting in $\mathsf{f}(s_1, \ldots, s_n)$ look like?

Proving the dependency pair theorem

Theorem: $\to_R$ is terminating if and only if there is no infinite reduction $s_1 \Rightarrow_{\mathsf{DP}} \to_R^* s_2 \Rightarrow_{\mathsf{DP}} \to_R^* s_3 \ldots$

Proof ($\Longleftarrow$):

- Suppose $\to_R$ is non-terminating.
- Then there is a minimal non-terminating term $\mathsf{f}(s_1, \ldots, s_n)$.
- Consider: what does an infinite reduction starting in $\mathsf{f}(s_1, \ldots, s_n)$ look like?
  $\Longrightarrow \mathsf{f}(s_1, \ldots, s_n) \to_R^* \mathsf{f}(s'_1, \ldots, s'_n) = \ell\sigma \to_R r\sigma \to_R^*$

Proving the dependency pair theorem

Theorem: $\to_R$ is terminating if and only if there is no infinite reduction $s_1 \Rightarrow_{\mathsf{DP}} \to_R^* s_2 \Rightarrow_{\mathsf{DP}} \to_R^* s_3 \ldots$

Proof ($\Longleftarrow$):

- Suppose $\to_R$ is non-terminating.
- Then there is a minimal non-terminating term $\mathsf{f}(s_1, \ldots, s_n)$.
- Consider: what does an infinite reduction starting in $\mathsf{f}(s_1, \ldots, s_n)$ look like?
  $\Longrightarrow \mathsf{f}(s_1, \ldots, s_n) \to_R^* \mathsf{f}(s'_1, \ldots, s'_n) = \ell\sigma \to_R r\sigma \to_R^*$
  $\Longrightarrow \mathsf{f}(s_1, \ldots, s_n) \to_R^* \mathsf{f}(s'_1, \ldots, s'_n) = \mathsf{f}(\ell_1, \ldots, \ell_n)\sigma \to_R r\sigma \to_R^*$

Proving the dependency pair theorem

Theorem: $\to_R$ is terminating if and only if there is no infinite reduction $s_1 \Rightarrow_{\mathsf{DP}} \to_R^* s_2 \Rightarrow_{\mathsf{DP}} \to_R^* s_3 \ldots$

Proof ($\Longleftarrow$):

- Suppose $\to_R$ is non-terminating.
- Then there is a minimal non-terminating term $\mathsf{f}(s_1, \ldots, s_n)$.
- Consider: what does an infinite reduction starting in $\mathsf{f}(s_1, \ldots, s_n)$ look like?
  $\Longrightarrow \mathsf{f}(s_1, \ldots, s_n) \to_R^* \mathsf{f}(s'_1, \ldots, s'_n) = \ell\sigma \to_R r\sigma \to_R^*$
  $\Longrightarrow \mathsf{f}(s_1, \ldots, s_n) \to_R^* \mathsf{f}(s'_1, \ldots, s'_n) = \mathsf{f}(\ell_1, \ldots, \ell_n)\sigma \to_R r\sigma \to_R^*$
- Let $\mathsf{g}(t_1, \ldots, t_m)$ be a minimal non-terminating subterm of $r\sigma$

Proving the dependency pair theorem

Theorem: $\to_R$ is terminating if and only if there is no infinite reduction $s_1 \Rightarrow_{\mathsf{DP}} \to_R^* s_2 \Rightarrow_{\mathsf{DP}} \to_R^* s_3 \ldots$

Proof ($\Longleftarrow$):

- Suppose $\to_R$ is non-terminating.
- Then there is a minimal non-terminating term $\mathsf{f}(s_1, \ldots, s_n)$.
- Consider: what does an infinite reduction starting in $\mathsf{f}(s_1, \ldots, s_n)$ look like?
  $\Longrightarrow \mathsf{f}(s_1, \ldots, s_n) \to_R^* \mathsf{f}(s'_1, \ldots, s'_n) = \ell\sigma \to_R r\sigma \to_R^*$
  $\Longrightarrow \mathsf{f}(s_1, \ldots, s_n) \to_R^* \mathsf{f}(s'_1, \ldots, s'_n) = \mathsf{f}(\ell_1, \ldots, \ell_n)\sigma \to_R r\sigma \to_R^*$
- Let $\mathsf{g}(t_1, \ldots, t_m)$ be a minimal non-terminating subterm of $r\sigma$
- Then necessarily: $\mathsf{g}(t_1, \ldots, t_m) = r'\sigma$ for some $r' \trianglelefteq r$

Building block: reduction pairs

How do you prove that there is no infinite chain?

then there is no infinite (DP, R)-chain!

Building block: reduction pairs

How do you prove that there is no infinite chain?

One possibility: using a variant of a reduction ordering: a reduction pair.

Theorem

If:

Building block: reduction pairs

How do you prove that there is no infinite chain?

One possibility: using a variant of a reduction ordering: a reduction pair.

Theorem

If:

- $\ell \succ r$ *for all dependency pairs* $\ell \Rightarrow r \in DP$,

Building block: reduction pairs

How do you prove that there is no infinite chain?

One possibility: using a variant of a reduction ordering: a reduction pair.

Theorem

If:

- $\ell \succ r$ *for all dependency pairs* $\ell \Rightarrow r \in DP$,
- $\ell \succeq r$ *for all rules* $\ell \to r \in R$,

Building block: reduction pairs

How do you prove that there is no infinite chain?

One possibility: using a variant of a reduction ordering: a reduction pair.

Theorem

If:

- $\ell \succ r$ *for all dependency pairs* $\ell \Rightarrow r \in DP$,
- $\ell \succeq r$ *for all rules* $\ell \to r \in R$,
- $\succ$ *is well-founded and stable (but not necessarily monotonic),*

Building block: reduction pairs

How do you prove that there is no infinite chain?

One possibility: using a variant of a reduction ordering: a reduction pair.

Theorem

If:

- $\ell \succ r$ *for all dependency pairs* $\ell \Rightarrow r \in DP$,
- $\ell \succeq r$ *for all rules* $\ell \to r \in R$,
- $\succ$ *is well-founded and stable (but not necessarily monotonic),*
- $\succeq$ *is stable and monotonic (but not necessarily well-founded),*

Building block: reduction pairs

How do you prove that there is no infinite chain?

One possibility: using a variant of a reduction ordering: a reduction pair.

Theorem

If:

- $\ell \succ r$ *for all dependency pairs* $\ell \Rightarrow r \in DP$,
- $\ell \succeq r$ *for all rules* $\ell \to r \in R$,
- $\succ$ *is well-founded and stable (but not necessarily monotonic),*
- $\succeq$ *is stable and monotonic (but not necessarily well-founded),*
- *and* $s \succ t \succeq u$ *implies* $s \succ u$,

Building block: reduction pairs

How do you prove that there is no infinite chain?

One possibility: using a variant of a reduction ordering: a reduction pair.

Theorem

If:

- $\ell \succ r$ *for all dependency pairs* $\ell \Rightarrow r \in DP$,
- $\ell \succeq r$ *for all rules* $\ell \to r \in R$,
- $\succ$ *is well-founded and stable (but not necessarily monotonic),*
- $\succeq$ *is stable and monotonic (but not necessarily well-founded),*
- *and* $s \succ t \succeq u$ *implies* $s \succ u$,

then there is no infinite (DP, R)*-chain!*

Building block: reduction pairs

Theorem

If:

- $\ell \succ r$ *for all dependency pairs* $\ell \Rightarrow r \in DP$,
- $\ell \succeq r$ *for all rules* $\ell \to r \in R$,
- $\succ$ *is well-founded and stable*,
- $\succeq$ *is stable and monotonic*,
- *and* $s \succ t \succeq u$ *implies* $s \succ u$,

then there is no infinite (DP, R)*-chain!*

Proof idea. By these requirements:

- If $s \Rightarrow_{\mathsf{DP}} t$ (by a step at the root), then $s \succ t$.
- If $s \to_R t$ then $s \succeq t$.
- Hence, if $s \Rightarrow_{\mathsf{DP}} \cdot \to_R^* t$, then $s \succ \cdot \succeq^* t$, and therefore $s \succ t$.

□

Quot/minus: ordering requirements

$$
\begin{array}{rcl}
\mathsf{minus}(x, 0) & \succeq & x \\
\mathsf{minus}(\mathsf{s}(x), \mathsf{s}(y)) & \succeq & \mathsf{minus}(x, y) \\
\mathsf{quot}(0, \mathsf{s}(y)) & \succeq & 0 \\
\mathsf{quot}(\mathsf{s}(x), \mathsf{s}(y)) & \succeq & \mathsf{s}(\mathsf{quot}(\mathsf{minus}(x, y), \mathsf{s}(y))) \\
\mathsf{minus}^\sharp(\mathsf{s}(x), \mathsf{s}(y)) & \succ & \mathsf{minus}^\sharp(x, y) \\
\mathsf{quot}^\sharp(\mathsf{s}(x), \mathsf{s}(y)) & \succ & \mathsf{quot}^\sharp(\mathsf{minus}(x, y), \mathsf{s}(y)) \\
\mathsf{quot}^\sharp(\mathsf{s}(x), \mathsf{s}(y)) & \succ & \mathsf{minus}^\sharp(x, y)
\end{array}
$$

Weakly monotonic algebras

Weakly monotonic algebras

Like monotonic algebras, but [f] only needs to be weakly monotonic.

Weakly monotonic algebras

Like monotonic algebras, but [f] only needs to be weakly monotonic.

$$\text{if } s \geq t, \text{ then } [\mathsf{f}](\ldots, s, \ldots) \geq [\mathsf{f}](\ldots, t, \ldots).$$

Weakly monotonic algebras

Like monotonic algebras, but [f] only needs to be weakly monotonic.

$$\text{if } s \geq t, \text{ then } [\mathsf{f}](\ldots, s, \ldots) \geq [\mathsf{f}](\ldots, t, \ldots).$$

Many functions that are not monotonic, are still weakly monotonic; for example:

- functions that ignore arguments: $\lambda x, y.\ x$
- min / max functions: $\lambda x, y.\ \min(x, y)$ or $\lambda x.\ \max(x - 1, 0)$

Completing quot/minus

Back to our example!

Completing quot/minus

Back to our example!

$$
\begin{array}{rcl}
\mathsf{minus}(x, 0) & \succeq & x \\
\mathsf{minus}(\mathsf{s}(x), \mathsf{s}(y)) & \succeq & \mathsf{minus}(x, y) \\
\mathsf{quot}(0, \mathsf{s}(y)) & \succeq & 0 \\
\mathsf{quot}(\mathsf{s}(x), \mathsf{s}(y)) & \succeq & \mathsf{s}(\mathsf{quot}(\mathsf{minus}(x, y), \mathsf{s}(y))) \\
\mathsf{minus}^\sharp(\mathsf{s}(x), \mathsf{s}(y)) & \succ & \mathsf{minus}^\sharp(x, y) \\
\mathsf{quot}^\sharp(\mathsf{s}(x), \mathsf{s}(y)) & \succ & \mathsf{quot}^\sharp(\mathsf{minus}(x, y), \mathsf{s}(y)) \\
\mathsf{quot}^\sharp(\mathsf{s}(x), \mathsf{s}(y)) & \succ & \mathsf{minus}^\sharp(x, y)
\end{array}
$$

Completing quot/minus

Back to our example!

$$\begin{array}{rcl}
\mathsf{minus}(x, 0) & \succeq & x \\
\mathsf{minus}(\mathsf{s}(x), \mathsf{s}(y)) & \succeq & \mathsf{minus}(x, y) \\
\mathsf{quot}(0, \mathsf{s}(y)) & \succeq & 0 \\
\mathsf{quot}(\mathsf{s}(x), \mathsf{s}(y)) & \succeq & \mathsf{s}(\mathsf{quot}(\mathsf{minus}(x, y), \mathsf{s}(y))) \\
\mathsf{minus}^\sharp(\mathsf{s}(x), \mathsf{s}(y)) & \succ & \mathsf{minus}^\sharp(x, y) \\
\mathsf{quot}^\sharp(\mathsf{s}(x), \mathsf{s}(y)) & \succ & \mathsf{quot}^\sharp(\mathsf{minus}(x, y), \mathsf{s}(y)) \\
\mathsf{quot}^\sharp(\mathsf{s}(x), \mathsf{s}(y)) & \succ & \mathsf{minus}^\sharp(x, y)
\end{array}$$

This is satisfied by choosing:

$$\begin{array}{rclrclrcl}
[0] & = & 0 & [\mathsf{minus}] & = & \lambda x, y.\ x & [\mathsf{minus}^\sharp] & = & \lambda x, y.\ x \\
[\mathsf{s}] & = & \lambda x.\ x + 1 & [\mathsf{quot}] & = & \lambda x, y.\ x & [\mathsf{quot}^\sharp] & = & \lambda x, y.\ x
\end{array}$$

Completing quot/minus

Back to our example!

$$\begin{array}{rcl} x & \geq & x \\ x+1 & \geq & x \\ 0 & \geq & 0 \\ x+1 & \geq & x+1 \\ x+1 & > & x \\ x+1 & > & x \\ x+1 & > & x \end{array}$$

This is satisfied by choosing:

$$\begin{array}{rclrclrcl} [0] & = & 0 & [\mathsf{minus}] & = & \lambda x, y.\ x & [\mathsf{minus}^\sharp] & = & \lambda x, y.\ x \\ [\mathsf{s}] & = & \lambda x.\ x+1 & [\mathsf{quot}] & = & \lambda x, y.\ x & [\mathsf{quot}^\sharp] & = & \lambda x, y.\ x \end{array}$$

Completing quot/minus

Back to our example!

$$\begin{array}{rcl} x & \geq & x \\ x+1 & \geq & x \\ 0 & \geq & 0 \\ x+1 & \geq & x+1 \\ x+1 & > & x \\ x+1 & > & x \\ x+1 & > & x \end{array}$$

This is satisfied by choosing:

$$\begin{array}{rclrclrcl} [0] & = & 0 & [\mathsf{minus}] & = & \lambda x, y.\ x & [\mathsf{minus}^\sharp] & = & \lambda x, y.\ x \\ [\mathsf{s}] & = & \lambda x.\ x+1 & [\mathsf{quot}] & = & \lambda x, y.\ x & [\mathsf{quot}^\sharp] & = & \lambda x, y.\ x \end{array}$$

Hence, we clearly gained something!

Step-by-step proofs

Challenge: dealing with large systems.

Step-by-step proofs

Challenge: dealing with large systems.

Idea: split large problems into smaller sub-problems.

Step-by-step proofs

Challenge: dealing with large systems.

Idea: split large problems into smaller sub-problems.

Example:

Rules:

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

DPs:

$$\begin{aligned}
\mathsf{A}\quad \mathsf{minus}^\sharp(\mathsf{s}(x),\mathsf{s}(y)) &\Rightarrow \mathsf{minus}^\sharp(x,y)\\
\mathsf{B}\quad \mathsf{quot}^\sharp(\mathsf{s}(x),\mathsf{s}(y)) &\Rightarrow \mathsf{minus}^\sharp(x,y)\\
\mathsf{C}\quad \mathsf{quot}^\sharp(\mathsf{s}(x),\mathsf{s}(y)) &\Rightarrow \mathsf{quot}^\sharp(\mathsf{minus}(x,y),\mathsf{s}(y))
\end{aligned}$$

Step-by-step proofs

Challenge: dealing with large systems.

Idea: split large problems into smaller sub-problems.

Example:

Rules:		$\mathsf{minus}(x, 0)$	$\to$	x
		$\mathsf{minus}(\mathsf{s}(x), \mathsf{s}(y))$	$\to$	$\mathsf{minus}(x, y)$
		$\mathsf{quot}(0, \mathsf{s}(y))$	$\to$	0
		$\mathsf{quot}(\mathsf{s}(x), \mathsf{s}(y))$	$\to$	$\mathsf{s}(\mathsf{quot}(\mathsf{minus}(x, y), \mathsf{s}(y)))$
DPs:	A	$\mathsf{minus}^\sharp(\mathsf{s}(x), \mathsf{s}(y))$	$\Rightarrow$	$\mathsf{minus}^\sharp(x, y)$
	B	$\mathsf{quot}^\sharp(\mathsf{s}(x), \mathsf{s}(y))$	$\Rightarrow$	$\mathsf{minus}^\sharp(x, y)$
	C	$\mathsf{quot}^\sharp(\mathsf{s}(x), \mathsf{s}(y))$	$\Rightarrow$	$\mathsf{quot}^\sharp(\mathsf{minus}(x, y), \mathsf{s}(y))$

Question: what does an infinite chain look like?

Step-by-step proofs

Challenge: dealing with large systems.

Idea: split large problems into smaller sub-problems.

Example:

Rules:		$\mathsf{minus}(x, 0)$	$\to$	x
		$\mathsf{minus}(\mathsf{s}(x), \mathsf{s}(y))$	$\to$	$\mathsf{minus}(x, y)$
		$\mathsf{quot}(0, \mathsf{s}(y))$	$\to$	0
		$\mathsf{quot}(\mathsf{s}(x), \mathsf{s}(y))$	$\to$	$\mathsf{s}(\mathsf{quot}(\mathsf{minus}(x, y), \mathsf{s}(y)))$
DPs:	A	$\mathsf{minus}^\sharp(\mathsf{s}(x), \mathsf{s}(y))$	$\Rightarrow$	$\mathsf{minus}^\sharp(x, y)$
	B	$\mathsf{quot}^\sharp(\mathsf{s}(x), \mathsf{s}(y))$	$\Rightarrow$	$\mathsf{minus}^\sharp(x, y)$
	C	$\mathsf{quot}^\sharp(\mathsf{s}(x), \mathsf{s}(y))$	$\Rightarrow$	$\mathsf{quot}^\sharp(\mathsf{minus}(x, y), \mathsf{s}(y))$

Question: what does an infinite chain look like?

We conclude: split DP into $\{A\}$ and $\{C\}$!

Using a graph

Using a graph

Idea: Graph with DPs as nodes, edges between nodes if one can follow another in a chain.

Using a graph

Idea: Graph with DPs as nodes, edges between nodes if one can follow another in a chain.

Example:

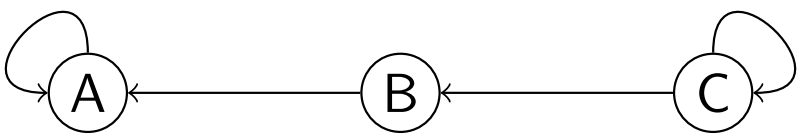
$$\begin{array}{lrcl} \text{A.} & \mathsf{minus}^\sharp(\mathsf{s}(x),\mathsf{s}(y)) & \Rightarrow & \mathsf{minus}^\sharp(x,y) \\ \text{B.} & \mathsf{quot}^\sharp(\mathsf{s}(x),\mathsf{s}(y)) & \Rightarrow & \mathsf{minus}^\sharp(x,y) \\ \text{C.} & \mathsf{quot}^\sharp(\mathsf{s}(x),\mathsf{s}(y)) & \Rightarrow & \mathsf{quot}^\sharp(\mathsf{minus}(x,y),\mathsf{s}(y)) \end{array}$$

Using a graph

Idea: Graph with DPs as nodes, edges between nodes if one can follow another in a chain.

Example:

$$
\begin{array}{lrcl}
\text{A.} & \mathsf{minus}^\sharp(\mathsf{s}(x),\mathsf{s}(y)) & \Rightarrow & \mathsf{minus}^\sharp(x,y) \\
\text{B.} & \mathsf{quot}^\sharp(\mathsf{s}(x),\mathsf{s}(y)) & \Rightarrow & \mathsf{minus}^\sharp(x,y) \\
\text{C.} & \mathsf{quot}^\sharp(\mathsf{s}(x),\mathsf{s}(y)) & \Rightarrow & \mathsf{quot}^\sharp(\mathsf{minus}(x,y),\mathsf{s}(y))
\end{array}
$$

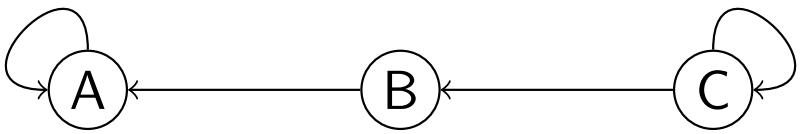


Using a graph

Idea: Graph with DPs as nodes, edges between nodes if one can follow another in a chain.

Example:

$$\begin{array}{lrcl} \text{A.} & \mathsf{minus}^\sharp(\mathsf{s}(x),\mathsf{s}(y)) & \Rightarrow & \mathsf{minus}^\sharp(x,y) \\ \text{B.} & \mathsf{quot}^\sharp(\mathsf{s}(x),\mathsf{s}(y)) & \Rightarrow & \mathsf{minus}^\sharp(x,y) \\ \text{C.} & \mathsf{quot}^\sharp(\mathsf{s}(x),\mathsf{s}(y)) & \Rightarrow & \mathsf{quot}^\sharp(\mathsf{minus}(x,y),\mathsf{s}(y)) \end{array}$$



Observation: each strongly connected component may be considered separately.

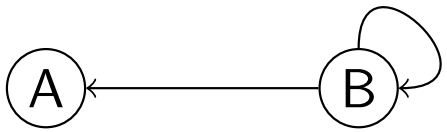
Dependency graph processor – another example

$$R = \{\mathsf{f}(\mathsf{f}(x)) \ \rightarrow \ \mathsf{f}(\mathsf{g}(\mathsf{f}(x)))\}$$

Dependency graph processor – another example

$$R = \{\mathsf{f}(\mathsf{f}(x)) \;\to\; \mathsf{f}(\mathsf{g}(\mathsf{f}(x)))\}$$

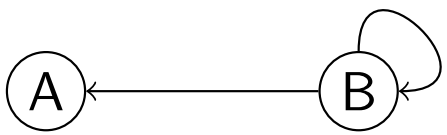
$$\mathrm{DP} = \left\{ \begin{array}{lllll} \text{A.} & \mathsf{f}^\sharp(\mathsf{f}(x)) & \Rightarrow & \mathsf{f}^\sharp(\mathsf{g}(\mathsf{f}(x))) \\ \text{B.} & \mathsf{f}^\sharp(\mathsf{f}(x)) & \Rightarrow & \mathsf{f}^\sharp(x) \end{array} \right\}$$



Dependency graph processor – another example

$$R = \{\mathsf{f}(\mathsf{f}(x)) \;\to\; \mathsf{f}(\mathsf{g}(\mathsf{f}(x)))\}$$

$$\mathrm{DP} = \left\{ \begin{array}{lllll} \text{A.} & \mathsf{f}^\sharp(\mathsf{f}(x)) & \Rightarrow & \mathsf{f}^\sharp(\mathsf{g}(\mathsf{f}(x))) \\ \text{B.} & \mathsf{f}^\sharp(\mathsf{f}(x)) & \Rightarrow & \mathsf{f}^\sharp(x) \end{array} \right\}$$

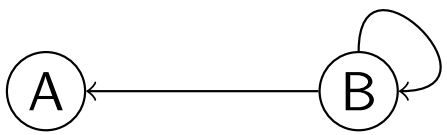


Hence, we can remove dependency pair A.

Dependency graph processor – another example

$$R = \{\mathsf{f}(\mathsf{f}(x)) \;\to\; \mathsf{f}(\mathsf{g}(\mathsf{f}(x)))\}$$

$$\mathrm{DP} = \left\{ \begin{array}{lll} \text{A.} & \mathsf{f}^\sharp(\mathsf{f}(x)) & \Rightarrow \quad \mathsf{f}^\sharp(\mathsf{g}(\mathsf{f}(x))) \\ \text{B.} & \mathsf{f}^\sharp(\mathsf{f}(x)) & \Rightarrow \quad \mathsf{f}^\sharp(x) \end{array} \right\}$$



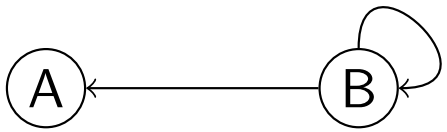
Hence, we can remove dependency pair A.

(And then quickly remove B, too, using weakly monotonic algebras.)

Dependency graph processor – another example

$$R = \{\mathsf{f}(\mathsf{f}(x)) \;\to\; \mathsf{f}(\mathsf{g}(\mathsf{f}(x)))\}$$

$$\mathrm{DP} = \left\{ \begin{array}{lll} \text{A.} & \mathsf{f}^\sharp(\mathsf{f}(x)) & \Rightarrow \quad \mathsf{f}^\sharp(\mathsf{g}(\mathsf{f}(x))) \\ \text{B.} & \mathsf{f}^\sharp(\mathsf{f}(x)) & \Rightarrow \quad \mathsf{f}^\sharp(x) \end{array} \right\}$$



Hence, we can remove dependency pair A.

(And then quickly remove B, too, using weakly monotonic algebras.)

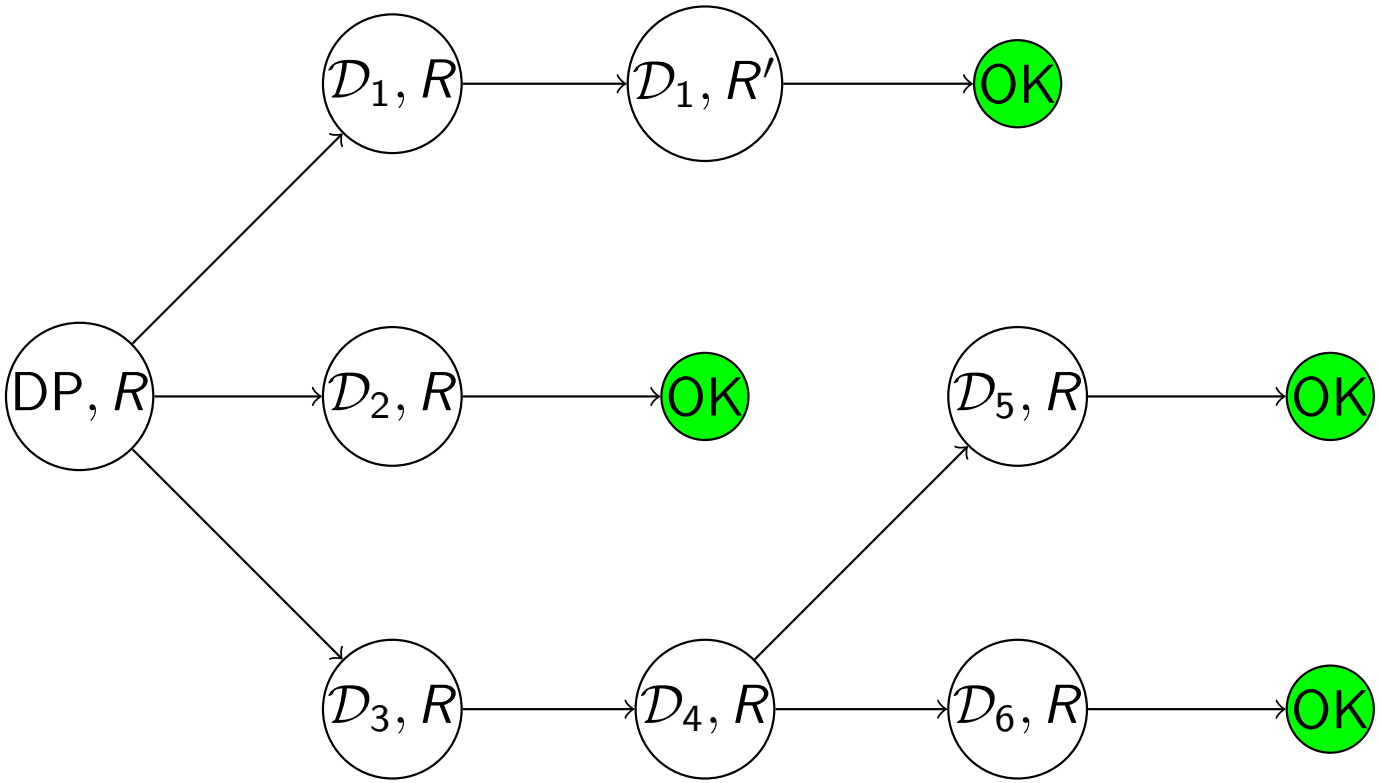
Note: not always easy to see if one DP can follow another, but we can use approximations.

The DP framework visualised

We may visualise an example termination proof by the DP framework as follows:

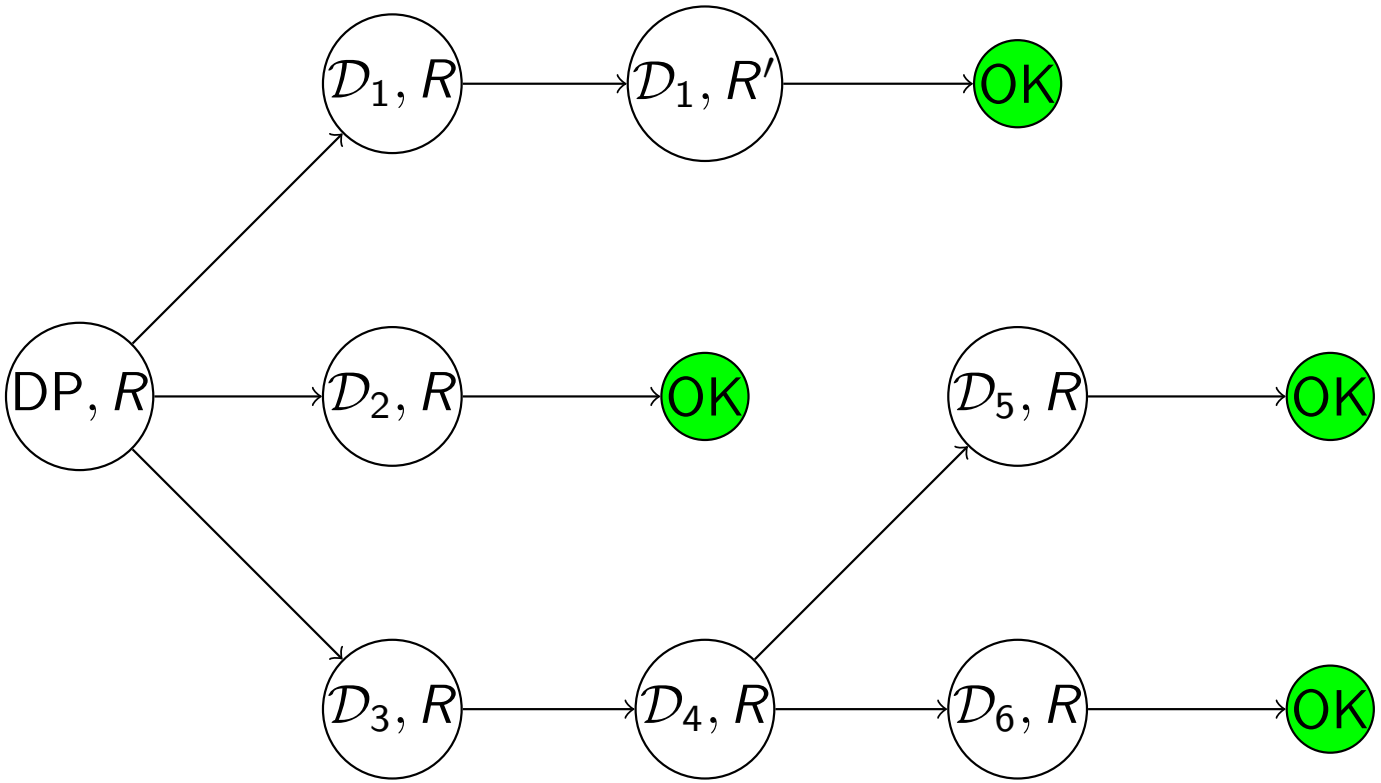
The DP framework visualised

We may visualise an example termination proof by the DP framework as follows:



The DP framework visualised

We may visualise an example termination proof by the DP framework as follows:



A pair $(\mathcal{D}, R)$ is called a DP problem, and a method to alter a DP problem is a DP processor.

The subterm criterion

$$
\begin{array}{lrcl}
\text{A.} & \exp^\sharp(\mathsf{s}(x), y) & \Rightarrow & \mathrm{double}^\sharp(x, y, 0) \\
\text{B.} & \mathrm{double}^\sharp(x, 0, z) & \Rightarrow & \exp^\sharp(x, z) \\
\text{C.} & \mathrm{double}^\sharp(x, 0, z) & \Rightarrow & \mathrm{double}^\sharp(x, y, \mathsf{s}(\mathsf{s}(z)))
\end{array}
$$

The subterm criterion

$$
\begin{array}{lrcl}
\text{A.} & \exp^\sharp(\mathsf{s}(x), y) & \Rightarrow & \mathrm{double}^\sharp(x, y, 0) \\
\text{B.} & \mathrm{double}^\sharp(x, 0, z) & \Rightarrow & \exp^\sharp(x, z) \\
\text{C.} & \mathrm{double}^\sharp(x, 0, z) & \Rightarrow & \mathrm{double}^\sharp(x, y, \mathsf{s}(\mathsf{s}(z)))
\end{array}
$$

Idea: consider the first argument of each side of the dependency pairs.

The subterm criterion

$$
\begin{array}{lrcl}
\text{A.} & \exp^\sharp(\underline{\mathsf{s}(x)}, y) & \Rightarrow & \mathrm{double}^\sharp(\underline{x}, y, 0) \\
\text{B.} & \mathrm{double}^\sharp(\underline{x}, 0, z) & \Rightarrow & \exp^\sharp(\underline{x}, z) \\
\text{C.} & \mathrm{double}^\sharp(\underline{x}, 0, z) & \Rightarrow & \mathrm{double}^\sharp(\underline{x}, y, \mathsf{s}(\mathsf{s}(z)))
\end{array}
$$

Idea: consider the first argument of each side of the dependency pairs.

The subterm criterion

$$
\begin{array}{lrcl}
\text{A.} & \exp^\sharp(\underline{\mathsf{s}(x)}, y) & \Rightarrow & \mathrm{double}^\sharp(\underline{x}, y, 0) \\
\text{B.} & \mathrm{double}^\sharp(\underline{x}, 0, z) & \Rightarrow & \exp^\sharp(\underline{x}, z) \\
\text{C.} & \mathrm{double}^\sharp(\underline{x}, 0, z) & \Rightarrow & \mathrm{double}^\sharp(\underline{x}, y, \mathsf{s}(\mathsf{s}(z)))
\end{array}
$$

Idea: consider the first argument of each side of the dependency pairs.

Then we can remove the dependency pairs where the chosen argument becomes smaller (in this case A).

The subterm criterion processor

Definition

For all marked symbols $\mathsf{f}^\sharp$, let $\nu(\mathsf{f}^\sharp) \in \{1, \ldots, \mathit{arity}(\mathsf{f})\}$.

Let $(\mathcal{D}, R)$ be a DP problem, and write $\mathcal{D} = \mathcal{D}_1 \cup \mathcal{D}_2$.

Suppose:

- $\ell_{\nu(\mathsf{f}^\sharp)} = r_{\nu(\mathsf{g}^\sharp)}$ for all $\mathsf{f}^\sharp(\ell_1, \ldots, \ell_n) \Rightarrow \mathsf{g}^\sharp(r_1, \ldots, r_m) \in \mathcal{D}_1$
- $r_{\nu(\mathsf{f}^\sharp)}$ is a subterm of $\ell_{\nu(\mathsf{g}^\sharp)}$ for all $\mathsf{f}^\sharp(\ell_1, \ldots, \ell_n) \Rightarrow \mathsf{g}^\sharp(r_1, \ldots, r_m) \in \mathcal{D}_2$

Then the subterm criterion processor maps $(\mathcal{D}, R)$ to $\{(\mathcal{D}_1, R)\}$.

Graph + subterm criterion

Class exercise:

$$
\begin{array}{rcl}
\mathsf{exp}(0, y) & \to & y \\
\mathsf{exp}(\mathsf{s}(x), y) & \to & \mathsf{double}(0, x, y) \\
\mathsf{double}(r, x, 0) & \to & \mathsf{exp}(x, r) \\
\mathsf{double}(r, x, \mathsf{s}(y)) & \to & \mathsf{double}(\mathsf{s}(\mathsf{s}(r)), x, y)
\end{array}
$$

Graph + subterm criterion

Class exercise:

$$
\begin{array}{rcl}
\mathsf{exp}(\mathsf{0}, y) & \to & y \\
\mathsf{exp}(\mathsf{s}(x), y) & \to & \mathsf{double}(\mathsf{0}, x, y) \\
\mathsf{double}(r, x, \mathsf{0}) & \to & \mathsf{exp}(x, r) \\
\mathsf{double}(r, x, \mathsf{s}(y)) & \to & \mathsf{double}(\mathsf{s}(\mathsf{s}(r)), x, y)
\end{array}
$$

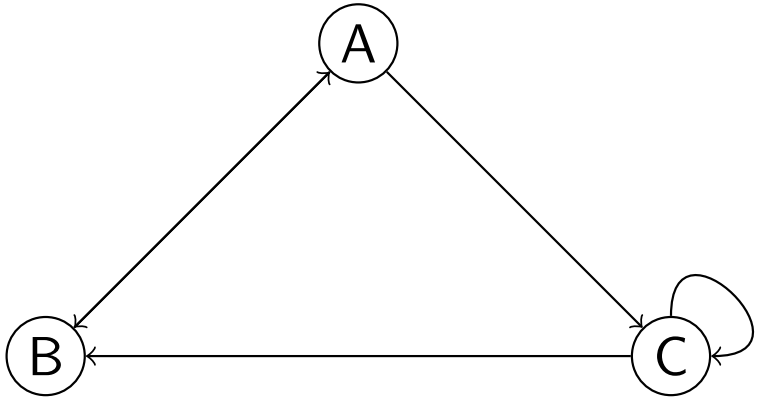
$$
\begin{array}{lrcl}
A. & \mathsf{exp}^\sharp(\mathsf{s}(x), y) & \Rightarrow & \mathsf{double}^\sharp(\mathsf{0}, x, y) \\
B. & \mathsf{double}^\sharp(r, x, \mathsf{0}) & \Rightarrow & \mathsf{exp}^\sharp(x, r) \\
C. & \mathsf{double}^\sharp(r, x, \mathsf{s}(y)) & \Rightarrow & \mathsf{double}^\sharp(\mathsf{s}(\mathsf{s}(r)), x, y)
\end{array}
$$

Graph + subterm criterion

Class exercise:

$$
\begin{array}{rcl}
\mathsf{exp}(0, y) & \to & y \\
\mathsf{exp}(\mathsf{s}(x), y) & \to & \mathsf{double}(0, x, y) \\
\mathsf{double}(r, x, 0) & \to & \mathsf{exp}(x, r) \\
\mathsf{double}(r, x, \mathsf{s}(y)) & \to & \mathsf{double}(\mathsf{s}(\mathsf{s}(r)), x, y)
\end{array}
$$

$$
\begin{array}{lrcl}
A. & \mathsf{exp}^\sharp(\mathsf{s}(x), y) & \Rightarrow & \mathsf{double}^\sharp(0, x, y) \\
B. & \mathsf{double}^\sharp(r, x, 0) & \Rightarrow & \mathsf{exp}^\sharp(x, r) \\
C. & \mathsf{double}^\sharp(r, x, \mathsf{s}(y)) & \Rightarrow & \mathsf{double}^\sharp(\mathsf{s}(\mathsf{s}(r)), x, y)
\end{array}
$$

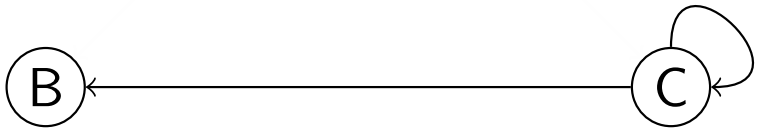


Graph + subterm criterion

Class exercise:

$$\begin{aligned}
\mathsf{exp}(0, y) &\to y \\
\mathsf{exp}(\mathsf{s}(x), y) &\to \mathsf{double}(0, x, y) \\
\mathsf{double}(r, x, 0) &\to \mathsf{exp}(x, r) \\
\mathsf{double}(r, x, \mathsf{s}(y)) &\to \mathsf{double}(\mathsf{s}(\mathsf{s}(r)), x, y)
\end{aligned}$$

$$\begin{aligned}
A.\quad \mathsf{exp}^\sharp(\mathsf{s}(x), y) &\Rightarrow \mathsf{double}^\sharp(0, x, y) \\
B.\quad \mathsf{double}^\sharp(r, x, 0) &\Rightarrow \mathsf{exp}^\sharp(x, r) \\
C.\quad \mathsf{double}^\sharp(r, x, \mathsf{s}(y)) &\Rightarrow \mathsf{double}^\sharp(\mathsf{s}(\mathsf{s}(r)), x, y)
\end{aligned}$$



Graph + subterm criterion

Class exercise:

$$
\begin{array}{rcl}
\mathsf{exp}(0, y) & \to & y \\
\mathsf{exp}(\mathsf{s}(x), y) & \to & \mathsf{double}(0, x, y) \\
\mathsf{double}(r, x, 0) & \to & \mathsf{exp}(x, r) \\
\mathsf{double}(r, x, \mathsf{s}(y)) & \to & \mathsf{double}(\mathsf{s}(\mathsf{s}(r)), x, y)
\end{array}
$$

$$
\begin{array}{lrcl}
A. & \mathsf{exp}^\sharp(\mathsf{s}(x), y) & \Rightarrow & \mathsf{double}^\sharp(0, x, y) \\
B. & \mathsf{double}^\sharp(r, x, 0) & \Rightarrow & \mathsf{exp}^\sharp(x, r) \\
C. & \mathsf{double}^\sharp(r, x, \mathsf{s}(y)) & \Rightarrow & \mathsf{double}^\sharp(\mathsf{s}(\mathsf{s}(r)), x, y)
\end{array}
$$

Graph + subterm criterion

Class exercise:

$$\begin{array}{rcl} \mathsf{exp}(0, y) & \to & y \\ \mathsf{exp}(\mathsf{s}(x), y) & \to & \mathsf{double}(0, x, y) \\ \mathsf{double}(r, x, 0) & \to & \mathsf{exp}(x, r) \\ \mathsf{double}(r, x, \mathsf{s}(y)) & \to & \mathsf{double}(\mathsf{s}(\mathsf{s}(r)), x, y) \end{array}$$

$$\begin{array}{lrcl} A. & \mathsf{exp}^\sharp(\mathsf{s}(x), y) & \Rightarrow & \mathsf{double}^\sharp(0, x, y) \\ B. & \mathsf{double}^\sharp(r, x, 0) & \Rightarrow & \mathsf{exp}^\sharp(x, r) \\ C. & \mathsf{double}^\sharp(r, x, \mathsf{s}(y)) & \Rightarrow & \mathsf{double}^\sharp(\mathsf{s}(\mathsf{s}(r)), x, y) \end{array}$$

Reduction pair processor

We can also reformulate reduction pairs as a processor:

Definition (Reduction Pair Processor)

Let $\succ$ be a well-founded, stable ordering and $\succeq$ a stable monotonic quasi-ordering on terms, such that $\succ\succeq \subseteq \succ$.

Let $(\mathcal{D}, R)$ be a DP problem, and write $\mathcal{D} = \mathcal{D}_1 \cup \mathcal{D}_2$.

Suppose:

- $\ell \succeq r$ for all $\ell \to r \in R$
- $\ell \succeq r$ for all $\ell \Rightarrow r \in \mathcal{D}_1$, and
- $\ell \succ r$ for all $\ell \Rightarrow r \in \mathcal{D}_2$.

Then the reduction pair processor maps $(\mathcal{D}, R)$ to $\{(\mathcal{D}_1, R)\}$.

Reduction pair processor

We can also reformulate reduction pairs as a processor:

Definition (Reduction Pair Processor)

Let $\succ$ be a well-founded, stable ordering and $\succeq$ a stable monotonic quasi-ordering on terms, such that $\succ\succeq \subseteq \succ$.

Let $(\mathcal{D}, R)$ be a DP problem, and write $\mathcal{D} = \mathcal{D}_1 \cup \mathcal{D}_2$.

Suppose:

- $\ell \succeq r$ for all $\ell \to r \in R$
- $\ell \succeq r$ for all $\ell \Rightarrow r \in \mathcal{D}_1$, and
- $\ell \succ r$ for all $\ell \Rightarrow r \in \mathcal{D}_2$.

Then the reduction pair processor maps $(\mathcal{D}, R)$ to $\{(\mathcal{D}_1, R)\}$.

That is, we can use a reduction pair, and remove all dependency pairs that were ordered with $\succ$.

Using a reduction pair processor

$$
\begin{array}{rcl}
\mathsf{append}(\mathsf{nil}, z) & \to & z \\
\mathsf{append}(\mathsf{cons}(x, y), z) & \to & \mathsf{cons}(x, \mathsf{append}(y, z)) \\
\mathsf{rev}(\mathsf{nil}) & \to & \mathsf{nil} \\
\mathsf{rev}(\mathsf{cons}(x, y)) & \to & \mathsf{append}(\mathsf{rev}(y), \mathsf{cons}(x, \mathsf{nil})) \\
\mathsf{append}^\sharp(\mathsf{cons}(x, y), z) & \Rightarrow & \mathsf{append}^\sharp(y, z) \\
\mathsf{rev}^\sharp(\mathsf{cons}(x, y)) & \Rightarrow & \mathsf{rev}^\sharp(y) \\
\mathsf{rev}^\sharp(\mathsf{cons}(x, y)) & \Rightarrow & \mathsf{append}^\sharp(\mathsf{rev}(y), \mathsf{cons}(x, \mathsf{nil}))
\end{array}
$$

Using a reduction pair processor

$$
\begin{array}{rcl}
\mathsf{append}(\mathsf{nil}, z) & \to & z \\
\mathsf{append}(\mathsf{cons}(x, y), z) & \to & \mathsf{cons}(x, \mathsf{append}(y, z)) \\
\mathsf{rev}(\mathsf{nil}) & \to & \mathsf{nil} \\
\mathsf{rev}(\mathsf{cons}(x, y)) & \to & \mathsf{append}(\mathsf{rev}(y), \mathsf{cons}(x, \mathsf{nil})) \\
\mathsf{append}^\sharp(\mathsf{cons}(x, y), z) & \Rightarrow & \mathsf{append}^\sharp(y, z) \\
\mathsf{rev}^\sharp(\mathsf{cons}(x, y)) & \Rightarrow & \mathsf{rev}^\sharp(y) \\
\mathsf{rev}^\sharp(\mathsf{cons}(x, y)) & \Rightarrow & \mathsf{append}^\sharp(\mathsf{rev}(y), \mathsf{cons}(x, \mathsf{nil}))
\end{array}
$$

We choose:

$$
\begin{array}{rcl}
[\mathsf{nil}] & = & 0 \\
[\mathsf{cons}] & = & \lambda x, y.\ y + 1
\end{array}
\qquad
\begin{array}{rcll}
[\mathsf{append}] & = & \lambda x, y. & x + y \\
[\mathsf{rev}] & = & \lambda x. & x \\
[\mathsf{append}^\sharp] & = & \lambda x, y. & x + y \\
[\mathsf{rev}^\sharp] & = & \lambda x. & x
\end{array}
$$

Using a reduction pair processor

$$\begin{array}{rcl} z & \geq & z \\ (y+1)+z & \geq & (y+z)+1 \\ 0 & \geq & 0 \\ y+1 & \geq & y+(0+1) \\ (y+1)+z & > & y+z \\ y+1 & > & y \\ y+1 & \geq & y+(0+1) \end{array}$$

We choose:

$$\begin{array}{rcl} [\mathsf{nil}] & = & 0 \\ [\mathsf{cons}] & = & \lambda x, y.\ y+1 \end{array} \qquad \begin{array}{rcl} [\mathsf{append}] & = & \lambda x, y.\ x+y \\ [\mathsf{rev}] & = & \lambda x.\ x \\ [\mathsf{append}^\sharp] & = & \lambda x, y.\ x+y \\ [\mathsf{rev}^\sharp] & = & \lambda x.\ x \end{array}$$

Using a reduction pair processor

$$\begin{array}{rcl} z & \geq & z \\ (y+1)+z & \geq & (y+z)+1 \\ 0 & \geq & 0 \\ y+1 & \geq & y+(0+1) \\ (y+1)+z & > & y+z \\ y+1 & > & y \\ y+1 & \geq & y+(0+1) \end{array}$$

Hence, we can remove all but the last dependency pair, and continue with:

$$(\{\mathsf{rev}^\sharp(\mathsf{cons}(x,y)) \Rightarrow \mathsf{append}^\sharp(\mathsf{rev}(y),\mathsf{cons}(x,\mathsf{nil}))\},R)$$

Argument filterings

Idea: remove arguments before using a reduction ordering

Argument filterings

Idea: remove arguments before using a reduction ordering

$$
\begin{array}{rcl}
\mathsf{minus}(x, 0) & \succeq & x \\
\mathsf{minus}(\mathsf{s}(x), \mathsf{s}(y)) & \succeq & \mathsf{minus}(x, y) \\
\mathsf{quot}(0, \mathsf{s}(y)) & \succeq & 0 \\
\mathsf{quot}(\mathsf{s}(x), \mathsf{s}(y)) & \succeq & \mathsf{s}(\mathsf{quot}(\mathsf{minus}(x, y), \mathsf{s}(y))) \\
\mathsf{minus}^\sharp(\mathsf{s}(x), \mathsf{s}(y)) & \succ & \mathsf{minus}^\sharp(x, y) \\
\mathsf{quot}^\sharp(\mathsf{s}(x), \mathsf{s}(y)) & \succ & \mathsf{quot}^\sharp(\mathsf{minus}(x, y), \mathsf{s}(y)) \\
\mathsf{quot}^\sharp(\mathsf{s}(x), \mathsf{s}(y)) & \succ & \mathsf{minus}^\sharp(x, y)
\end{array}
$$

Argument filterings

Idea: remove arguments before using a reduction ordering

$$
\begin{array}{rcl}
\mathsf{minus}(x,0) & \succeq & x \\
\mathsf{minus}(\mathsf{s}(x),\mathsf{s}(y)) & \succeq & \mathsf{minus}(x,y) \\
\mathsf{quot}(0,\mathsf{s}(y)) & \succeq & 0 \\
\mathsf{quot}(\mathsf{s}(x),\mathsf{s}(y)) & \succeq & \mathsf{s}(\mathsf{quot}(\mathsf{minus}(x,y),\mathsf{s}(y))) \\
\mathsf{minus}^\sharp(\mathsf{s}(x),\mathsf{s}(y)) & \succ & \mathsf{minus}^\sharp(x,y) \\
\mathsf{quot}^\sharp(\mathsf{s}(x),\mathsf{s}(y)) & \succ & \mathsf{quot}^\sharp(\mathsf{minus}(x,y),\mathsf{s}(y)) \\
\mathsf{quot}^\sharp(\mathsf{s}(x),\mathsf{s}(y)) & \succ & \mathsf{minus}^\sharp(x,y)
\end{array}
$$

Replace $\mathsf{minus}(x,y)$ by $\mathsf{minus}'(x)$.

Argument filterings

Idea: remove arguments before using a reduction ordering

$$
\begin{array}{rcl}
\mathsf{minus}'(x) & \trianglerighteq & x \\
\mathsf{minus}'(\mathsf{s}(x)) & \trianglerighteq & \mathsf{minus}'(x) \\
\mathsf{quot}(0, \mathsf{s}(y)) & \trianglerighteq & 0 \\
\mathsf{quot}(\mathsf{s}(x), \mathsf{s}(y)) & \trianglerighteq & \mathsf{s}(\mathsf{quot}(\mathsf{minus}'(x), \mathsf{s}(y))) \\
\mathsf{minus}^\sharp(\mathsf{s}(x), \mathsf{s}(y)) & \succ & \mathsf{minus}^\sharp(x, y) \\
\mathsf{quot}^\sharp(\mathsf{s}(x), \mathsf{s}(y)) & \succ & \mathsf{quot}^\sharp(\mathsf{minus}'(x), \mathsf{s}(y)) \\
\mathsf{quot}^\sharp(\mathsf{s}(x), \mathsf{s}(y)) & \succ & \mathsf{minus}^\sharp(x, y)
\end{array}
$$

Replace $\mathsf{minus}(x, y)$ by $\mathsf{minus}'(x)$.

Argument filterings

Idea: remove arguments before using a reduction ordering

$$
\begin{array}{rcl}
\mathsf{minus}'(x) & \succeq & x \\
\mathsf{minus}'(\mathsf{s}(x)) & \succeq & \mathsf{minus}'(x) \\
\mathsf{quot}(0, \mathsf{s}(y)) & \succeq & 0 \\
\mathsf{quot}(\mathsf{s}(x), \mathsf{s}(y)) & \succeq & \mathsf{s}(\mathsf{quot}(\mathsf{minus}'(x), \mathsf{s}(y))) \\
\mathsf{minus}^\sharp(\mathsf{s}(x), \mathsf{s}(y)) & \succ & \mathsf{minus}^\sharp(x, y) \\
\mathsf{quot}^\sharp(\mathsf{s}(x), \mathsf{s}(y)) & \succ & \mathsf{quot}^\sharp(\mathsf{minus}'(x), \mathsf{s}(y)) \\
\mathsf{quot}^\sharp(\mathsf{s}(x), \mathsf{s}(y)) & \succ & \mathsf{minus}^\sharp(x, y)
\end{array}
$$

Replace $\mathsf{minus}(x, y)$ by $\mathsf{minus}'(x)$.

This can be handled using LPO with $\mathsf{s} \rhd \mathsf{minus}'$.

Argument filterings

Idea: remove arguments before using a reduction ordering

$$
\begin{array}{rcl}
\mathsf{minus}'(x) & \succeq & x \\
\mathsf{minus}'(\mathsf{s}(x)) & \succeq & \mathsf{minus}'(x) \\
\mathsf{quot}(0, \mathsf{s}(y)) & \succeq & 0 \\
\mathsf{quot}(\mathsf{s}(x), \mathsf{s}(y)) & \succeq & \mathsf{s}(\mathsf{quot}(\mathsf{minus}'(x), \mathsf{s}(y))) \\
\mathsf{minus}^\sharp(\mathsf{s}(x), \mathsf{s}(y)) & \succ & \mathsf{minus}^\sharp(x, y) \\
\mathsf{quot}^\sharp(\mathsf{s}(x), \mathsf{s}(y)) & \succ & \mathsf{quot}^\sharp(\mathsf{minus}'(x), \mathsf{s}(y)) \\
\mathsf{quot}^\sharp(\mathsf{s}(x), \mathsf{s}(y)) & \succ & \mathsf{minus}^\sharp(x, y)
\end{array}
$$

Replace $\mathsf{minus}(x, y)$ by $\mathsf{minus}'(x)$.

This can be handled using LPO with $\mathsf{s} \rhd \mathsf{minus}'$.

Searching a filter can be included in the SAT encoding of LPO.

Usable rules

When considering only a small number of dependency pairs, we might not need all the rules anymore.

Usable rules

When considering only a small number of dependency pairs, we might not need all the rules anymore.

$$
\begin{array}{rcl}
\mathsf{quot}^\sharp(\mathsf{s}(x),\mathsf{s}(y)) & \succ & \mathsf{quot}^\sharp(\mathsf{minus}(x,y),\mathsf{s}(y)) \\
\\
\mathsf{minus}(x,0) & \succeq & x \\
\mathsf{minus}(\mathsf{s}(x),\mathsf{s}(y)) & \succeq & \mathsf{minus}(x,y) \\
\mathsf{quot}(0,\mathsf{s}(y)) & \succeq & 0 \\
\mathsf{quot}(\mathsf{s}(x),\mathsf{s}(y)) & \succeq & \mathsf{s}(\mathsf{quot}(\mathsf{minus}(x,y),\mathsf{s}(y)))
\end{array}
$$

Usable rules

When considering only a small number of dependency pairs, we might not need all the rules anymore.

$$
\begin{array}{rcl}
\mathsf{quot}^\sharp(\mathsf{s}(x),\mathsf{s}(y)) & \succ & \mathsf{quot}^\sharp(\underline{\mathsf{minus}(x,y)},\mathsf{s}(y)) \\
\\
\mathsf{minus}(x,0) & \succeq & x \\
\mathsf{minus}(\mathsf{s}(x),\mathsf{s}(y)) & \succeq & \mathsf{minus}(x,y) \\
\mathsf{quot}(0,\mathsf{s}(y)) & \succeq & 0 \\
\mathsf{quot}(\mathsf{s}(x),\mathsf{s}(y)) & \succeq & \mathsf{s}(\mathsf{quot}(\mathsf{minus}(x,y),\mathsf{s}(y)))
\end{array}
$$

Usable rules

When considering only a small number of dependency pairs, we might not need all the rules anymore.

$$
\begin{array}{rcl}
\mathsf{quot}^\sharp(\mathsf{s}(x),\mathsf{s}(y)) & \succ & \mathsf{quot}^\sharp(\mathsf{minus}(x,y),\mathsf{s}(y)) \\
\\
\mathsf{minus}(x,0) & \succeq & x \\
\mathsf{minus}(\mathsf{s}(x),\mathsf{s}(y)) & \succeq & \underline{\mathsf{minus}(x,y)} \\
\mathsf{quot}(0,\mathsf{s}(y)) & \succeq & 0 \\
\mathsf{quot}(\mathsf{s}(x),\mathsf{s}(y)) & \succeq & \mathsf{s}(\mathsf{quot}(\mathsf{minus}(x,y),\mathsf{s}(y)))
\end{array}
$$

Usable rules

When considering only a small number of dependency pairs, we might not need all the rules anymore.

$$\begin{array}{rcl} \mathsf{quot}^\sharp(\mathsf{s}(x),\mathsf{s}(y)) & \succ & \mathsf{quot}^\sharp(\mathsf{minus}(x,y),\mathsf{s}(y)) \\ \\ \mathsf{minus}(x,0) & \succeq & x \\ \mathsf{minus}(\mathsf{s}(x),\mathsf{s}(y)) & \succeq & \mathsf{minus}(x,y) \end{array}$$

Usable rules

When considering only a small number of dependency pairs, we might not need all the rules anymore.

$$\begin{array}{rcl} \mathsf{quot}^\sharp(\mathsf{s}(x),\mathsf{s}(y)) & \succ & \mathsf{quot}^\sharp(\mathsf{minus}(x,y),\mathsf{s}(y)) \\ \\ \mathsf{minus}(x,0) & \succeq & x \\ \mathsf{minus}(\mathsf{s}(x),\mathsf{s}(y)) & \succeq & \mathsf{minus}(x,y) \end{array}$$

We do get two extra requirements, $\mathsf{c}(x,y) \succeq x$ and $\mathsf{c}(x,y) \succeq y$.

Usable rules

When considering only a small number of dependency pairs, we might not need all the rules anymore.

$$\begin{array}{rcl} \mathsf{quot}^\sharp(\mathsf{s}(x),\mathsf{s}(y)) & \succ & \mathsf{quot}^\sharp(\mathsf{minus}(x,y),\mathsf{s}(y)) \\ \\ \mathsf{minus}(x,0) & \succeq & x \\ \mathsf{minus}(\mathsf{s}(x),\mathsf{s}(y)) & \succeq & \mathsf{minus}(x,y) \end{array}$$

We do get two extra requirements, $\mathsf{c}(x,y) \succeq x$ and $\mathsf{c}(x,y) \succeq y$.

Finding usable rules is a very simple reachability algorithm.

Usable rules

When considering only a small number of dependency pairs, we might not need all the rules anymore.

$$\begin{array}{rcl} \mathsf{quot}^\sharp(\mathsf{s}(x),\mathsf{s}(y)) & \succ & \mathsf{quot}^\sharp(\mathsf{minus}(x,y),\mathsf{s}(y)) \\ \\ \mathsf{minus}(x,0) & \succeq & x \\ \mathsf{minus}(\mathsf{s}(x),\mathsf{s}(y)) & \succeq & \mathsf{minus}(x,y) \end{array}$$

We do get two extra requirements, $\mathsf{c}(x,y) \succeq x$ and $\mathsf{c}(x,y) \succeq y$.

Finding usable rules is a very simple reachability algorithm.

More complex (but powerful!): combining usable rules with an argument filtering and a reduction pair.