# Automated Reasoning

## Week 10. Termination

Cynthia Kop

Fall 2024

# Recall: Term Rewriting Systems

$$
\begin{array}{rcl}
\mathtt{add}(x, \mathtt{s}(y)) & \Rightarrow & \mathtt{s}(\mathtt{add}(x, y)) \\
\mathtt{add}(x, \mathtt{p}(y)) & \Rightarrow & \mathtt{p}(\mathtt{add}(x, y)) \\
\mathtt{add}(x, \mathtt{0}) & \Rightarrow & x \\
\mathtt{s}(\mathtt{p}(x)) & \Rightarrow & x \\
\mathtt{p}(\mathtt{s}(x)) & \Rightarrow & x
\end{array}
$$

# Recall: Term Rewriting Systems

$$
\begin{array}{rcl}
\mathtt{add}(x, \mathtt{s}(y)) & \Rightarrow & \mathtt{s}(\mathtt{add}(x, y)) \\
\mathtt{add}(x, \mathtt{p}(y)) & \Rightarrow & \mathtt{p}(\mathtt{add}(x, y)) \\
\mathtt{add}(x, \mathtt{0}) & \Rightarrow & x \\
\mathtt{s}(\mathtt{p}(x)) & \Rightarrow & x \\
\mathtt{p}(\mathtt{s}(x)) & \Rightarrow & x
\end{array}
$$

$$
\begin{array}{rcl}
\mathtt{rev}(\mathtt{nil}) & \Rightarrow & \mathtt{nil} \\
\mathtt{rev}(a \mathbin{\mathtt{:}} x) & \Rightarrow & \mathtt{conc}(\mathtt{rev}(x), a \mathbin{\mathtt{:}} \mathtt{nil}) \\
\mathtt{conc}(\mathtt{nil}, x) & \Rightarrow & x \\
\mathtt{conc}(a \mathbin{\mathtt{:}} x, y) & \Rightarrow & a \mathbin{\mathtt{:}} \mathtt{conc}(x, y)
\end{array}
$$

# Lecture plan

Last week: LPO

# Lecture plan

Last week: LPO

- Simple
- Powerful
- Commonly used in, e.g., completion and superposition.

# Lecture plan

Last week: LPO

- Simple
- Powerful
- Commonly used in, e.g., completion and superposition.
- Not particularly intuitive
- Not the most commonly used method

# Lecture plan

Last week: LPO

- Simple
- Powerful
- Commonly used in, e.g., completion and superposition.
- Not particularly intuitive
- Not the most commonly used method

This lecture: **monotonic algebras** and **dependency pairs**.

# Basic intuition of **monotonic algebras**

# Basic intuition of **monotonic algebras**

*Find a weight function $W$ from terms to natural numbers in such a way that $W(u) > W(v)$ for all terms $u, v$ satisfying $u \Rightarrow_{\mathcal{R}} v$.*

# Example

Rules:

$$
\begin{array}{rcl}
\mathtt{add}(\mathtt{0}, y) & \rightarrow & y \\
\mathtt{add}(\mathtt{s}(x), y) & \rightarrow & \mathtt{s}(\mathtt{add}(x, y))
\end{array}
$$

# Example

Rules:

$$\begin{array}{rcl} \texttt{add}(\texttt{0}, y) & \to & y \\ \texttt{add}(\texttt{s}(x), y) & \to & \texttt{s}(\texttt{add}(x, y)) \end{array}$$

Define:

- $W(\texttt{0}) = 1$
- $W(\texttt{s}(t)) = W(t) + 1$
- $W(\texttt{add}(t, u)) = 2W(t) + W(u)$

# Example

## Rules:

$$
\begin{aligned}
\mathtt{add}(\mathtt{0}, y) &\rightarrow y \\
\mathtt{add}(\mathtt{s}(x), y) &\rightarrow \mathtt{s}(\mathtt{add}(x, y))
\end{aligned}
$$

## Define:

- $W(\mathtt{0}) = 1$
- $W(\mathtt{s}(t)) = W(t) + 1$
- $W(\mathtt{add}(t, u)) = 2W(t) + W(u)$

## Then:

$$
\begin{aligned}
W(\mathtt{add}(\mathtt{s}(\mathtt{0}), \mathtt{0})) &= 2 * W(\mathtt{s}(\mathtt{0})) + W(\mathtt{0}) = 2 * 2 + 1 = 5 \\
W(\mathtt{s}(\mathtt{add}(\mathtt{0}, \mathtt{0}))) &= W(\mathtt{add}(\mathtt{0}, \mathtt{0})) + 1 = (2 * 1 + 1) + 1 = 4 \\
W(\mathtt{s}(\mathtt{0})) &= W(\mathtt{0}) + 1 = 1 + 1 = 2
\end{aligned}
$$

# Better idea: reduction ordering

Problem: infinitely many terms!

# Better idea: reduction ordering

Problem: infinitely many terms!

Solution: use a reduction ordering!

# Better idea: reduction ordering

Problem: infinitely many terms!

Solution: use a reduction ordering!

- $\succ$ is **well-founded**
- $\succ$ is **stable** (so preserved under substitution)
- $\succ$ is **monotonic** (so preserved under contexts)

# Better idea: reduction ordering

Problem: infinitely many terms!

Solution: use a reduction ordering!

- $\succ$ is **well-founded**
- $\succ$ is **stable** (so preserved under substitution)
- $\succ$ is **monotonic** (so preserved under contexts)

Then it suffices to prove $\ell \succ r$ for all the rules.

# Monotonic algebras

Goal:

# Monotonic algebras

## Goal:

- Choose finitely many **function symbol** interpretations.

# Monotonic algebras

Goal:

- Choose finitely many **function symbol** interpretations.
- Prove $W(\ell) \succ W(r)$ for finitely many **rules**.

# Monotonic algebras

Goal:

- Choose finitely many **function symbol** interpretations.
- Prove $W(\ell) \succ W(r)$ for finitely many **rules**.

Side bonus: no restriction to $\mathbb{N}$.

# Definition

Let $\mathcal{A}$ be a set and $>$ a well-founded relation on $\mathcal{A}$.

# Definition

Let $\mathcal{A}$ be a set and $>$ a well-founded relation on $\mathcal{A}$.

For every function symbol $\mathtt{f}$ of arity $n$, we choose a **monotonic** function $[\mathtt{f}] : \mathcal{A}^n \to \mathcal{A}$.

# Definition

Let $\mathcal{A}$ be a set and $>$ a well-founded relation on $\mathcal{A}$.

For every function symbol $\mathtt{f}$ of arity $n$, we choose a **monotonic** function $[\mathtt{f}] : \mathcal{A}^n \to \mathcal{A}$.

Here **monotonic** means:

*if for all* $a_i, b_i \in \mathcal{A}$ *for* $i = 1, \ldots, n$ *with* $a_i > b_i$ *for some* $i$ *and* $a_j \geq b_j$ *for all* $j \neq i$ *then*

$$[\mathtt{f}](a_1, \ldots, a_n) \; > \; [\mathtt{f}](b_1, \ldots, b_n)$$

# Definition

Let $\mathcal{A}$ be a set and $>$ a well-founded relation on $\mathcal{A}$.

For every function symbol $\mathtt{f}$ of arity $n$, we choose a **monotonic** function $[\mathtt{f}] : \mathcal{A}^n \to \mathcal{A}$.

Here **monotonic** means:

> *if for all $a_i, b_i \in \mathcal{A}$ for $i = 1, \ldots, n$ with $a_i > b_i$ for some $i$ and $a_j \geq b_j$ for all $j \neq i$ then*

$$[\mathtt{f}](a_1, \ldots, a_n) \;>\; [\mathtt{f}](b_1, \ldots, b_n)$$

**Examples:**

| *monotonic* | *not monotonic* |
|---|---|
| $\lambda x.\ x$ | $\lambda x.\ 2$ |
| $\lambda x.\ x + 1$ | $\lambda x, y.\ x + 1$ |
| $\lambda x.\ 2 * x$ | $\lambda x, y.\ x * y$ |
| $\lambda x, y.\ x + y$ | $\lambda x, y.\ \max(x, y)$ |
| $\lambda x, y.\ 2 * x + y + 1$ | |

# Definition (continued)

Define:

- $W(x) = x$ for a variable
- $W(\mathtt{f}(s_1, \ldots, s_n)) = [\mathtt{f}](W(s_1), \ldots, W(s_n))$

# Definition (continued)

Define:

- $W(x) = x$ for a variable
- $W(\mathtt{f}(s_1, \ldots, s_n)) = [\mathtt{f}](W(s_1), \ldots, W(s_n))$

**Theorem**

The relation $\succ$ defined by:

$$s \succ t \text{ if and only if } \forall \vec{x}[W(s) \succ W(t)],$$
$$\text{where } \{\vec{x}\} \text{ is the set of variables occurring in } s, t$$

is a reduction ordering.

# Definition (continued)

Define:

- $W(x) = x$ for a variable
- $W(\mathtt{f}(s_1, \ldots, s_n)) = [\mathtt{f}](W(s_1), \ldots, W(s_n))$

**Theorem**

The relation $\succ$ defined by:

$$s \succ t \text{ if and only if } \forall \vec{x}[W(s) \succ W(t)],$$
$$\text{where } \{\vec{x}\} \text{ is the set of variables occurring in } s, t$$

is a reduction ordering.

**Proof idea.** Stability follows from the $\forall \vec{x}$, monotonicity from monotonicity of all $[\mathtt{f}]$ interpretation functions, and well-foundedness from well-foundedness of $>$ in $\mathcal{A}$. $\square$

# Example

$$
\begin{array}{rcl}
\mathtt{add}(\mathtt{0}, y) & \to & y \\
\mathtt{add}(\mathtt{s}(x), y) & \to & \mathtt{s}(\mathtt{add}(x, y))
\end{array}
$$

# Example

$$\begin{array}{rcl} \mathtt{add}(\mathtt{0}, y) & \to & y \\ \mathtt{add}(\mathtt{s}(x), y) & \to & \mathtt{s}(\mathtt{add}(x, y)) \end{array}$$

We let:

- $[\mathtt{0}] = 1$
- $[\mathtt{s}] = \lambda x.x + 1$
- $[\mathtt{add}] = \lambda x, y.2 * x + y$

# Example

$$\begin{array}{rcl} \mathtt{add}(\mathtt{0}, y) & \to & y \\ \mathtt{add}(\mathtt{s}(x), y) & \to & \mathtt{s}(\mathtt{add}(x, y)) \end{array}$$

We let:

- $[\mathtt{0}] = 1$
- $[\mathtt{s}] = \lambda x.x + 1$
- $[\mathtt{add}] = \lambda x, y.2 * x + y$

Now indeed for all $x, y$ we have:

$$W(\mathtt{add}(\mathtt{0}, y)) = 2 * 1 + y = 2 + y > y = W(y)$$

# Example

$$\begin{array}{rcl} \mathtt{add}(\mathtt{0}, y) & \to & y \\ \mathtt{add}(\mathtt{s}(x), y) & \to & \mathtt{s}(\mathtt{add}(x, y)) \end{array}$$

We let:

- $[\mathtt{0}] = 1$
- $[\mathtt{s}] = \lambda x.x + 1$
- $[\mathtt{add}] = \lambda x, y.2 * x + y$

Now indeed for all $x, y$ we have:

$$W(\mathtt{add}(\mathtt{0}, y)) = 2 * 1 + y = 2 + y > y = W(y)$$

and:

$$\begin{array}{rcl} W(\mathtt{add}(\mathtt{s}(x), y) & = & 2 * (x + 1) + y = 2 * x + y + 2 \\ & > & 2 * x + y + 1 = W(\mathtt{s}(\mathtt{add}(x, y))) \end{array}$$

# Another example

For the TRS $\mathcal{R}$ consisting of the single rule

# Another example

For the TRS $\mathcal{R}$ consisting of the single rule

$$\mathtt{f}(\mathtt{g}(x)) \to \mathtt{g}(\mathtt{g}(\mathtt{f}(x)))$$

# Another example

For the TRS $\mathcal{R}$ consisting of the single rule

$$\mathtt{f}(\mathtt{g}(x)) \to \mathtt{g}(\mathtt{g}(\mathtt{f}(x)))$$

we choose monotonic functions

$$[\mathtt{f}](x) = 3x$$

$$[\mathtt{g}](x) = x + 1$$

# Another example

For the TRS $\mathcal{R}$ consisting of the single rule

$$\mathtt{f}(\mathtt{g}(x)) \to \mathtt{g}(\mathtt{g}(\mathtt{f}(x)))$$

we choose monotonic functions

$$[\mathtt{f}](x) = 3x$$

$$[\mathtt{g}](x) = x + 1$$

Now indeed for all $x \in \mathbb{N}$ we have:

# Another example

For the TRS $\mathcal{R}$ consisting of the single rule

$$\mathtt{f}(\mathtt{g}(x)) \to \mathtt{g}(\mathtt{g}(\mathtt{f}(x)))$$

we choose monotonic functions

$$[\mathtt{f}](x) = 3x$$

$$[\mathtt{g}](x) = x + 1$$

Now indeed for all $x \in \mathbb{N}$ we have:

$$W(\mathtt{f}(\mathtt{g}(x))) = 3(x+1) >$$
$$3x + 1 + 1 = W(\mathtt{g}(\mathtt{g}(\mathtt{f}(x))))$$

# Another example

For the TRS $\mathcal{R}$ consisting of the single rule

$$\mathtt{f}(\mathtt{g}(x)) \to \mathtt{g}(\mathtt{g}(\mathtt{f}(x)))$$

we choose monotonic functions

$$[\mathtt{f}](x) = 3x$$

$$[\mathtt{g}](x) = x + 1$$

Now indeed for all $x \in \mathbb{N}$ we have:

$$W(\mathtt{f}(\mathtt{g}(x))) = 3(x+1) >$$
$$3x + 1 + 1 = W(\mathtt{g}(\mathtt{g}(\mathtt{f}(x))))$$

Hence proving termination of $\mathcal{R}$.

# Yet another example?

For the TRS consisting of the single rule

$$\mathtt{f}(x) \to \mathtt{g}(\mathtt{f}(x))$$

# Yet another example?

For the TRS consisting of the single rule

$$\mathtt{f}(x) \to \mathtt{g}(\mathtt{f}(x))$$

we choose the functions

$$[\mathtt{f}](x) = x + 1$$

$$[\mathtt{g}](x) = 0$$

# Yet another example?

For the TRS consisting of the single rule

$$\mathtt{f}(x) \to \mathtt{g}(\mathtt{f}(x))$$

we choose the functions

$$[\mathtt{f}](x) = x + 1$$

$$[\mathtt{g}](x) = 0$$

Then we indeed have:

$$W(\mathtt{f}(x)) = x + 1 > 0 = W(\mathtt{g}(\mathtt{f}(x)))$$

# Yet another example?

For the TRS consisting of the single rule

$$\mathtt{f}(x) \to \mathtt{g}(\mathtt{f}(x))$$

we choose the functions

$$[\mathtt{f}](x) = x + 1$$

$$[\mathtt{g}](x) = 0$$

Then we indeed have:

$$W(\mathtt{f}(x)) = x + 1 > 0 = W(\mathtt{g}(\mathtt{f}(x)))$$

Yet, this system is non-terminating!

# Yet another example?

For the TRS consisting of the single rule

$$\mathtt{f}(x) \to \mathtt{g}(\mathtt{f}(x))$$

we choose the functions

$$[\mathtt{f}](x) = x + 1$$

$$[\mathtt{g}](x) = 0$$

Then we indeed have:

$$W(\mathtt{f}(x)) = x + 1 > 0 = W(\mathtt{g}(\mathtt{f}(x)))$$

Yet, this system is non-terminating! Where is the error?

# Yet another example?

For the TRS consisting of the single rule

$$\mathtt{f}(x) \to \mathtt{g}(\mathtt{f}(x))$$

we choose the functions

$$[\mathtt{f}](x) = x + 1$$

$$[\mathtt{g}](x) = 0$$

Then we indeed have:

$$W(\mathtt{f}(x)) = x + 1 > 0 = W(\mathtt{g}(\mathtt{f}(x)))$$

Yet, this system is non-terminating! Where is the error?

$[\mathtt{g}]$ is not monotonic.

# Yet another example?

For the TRS consisting of the single rule

$$\mathtt{f}(x) \to \mathtt{g}(\mathtt{f}(x))$$

we choose the functions

$$[\mathtt{f}](x) = x + 1$$

$$[\mathtt{g}](x) = 0$$

Then we indeed have:

$$W(\mathtt{f}(x)) = x + 1 > 0 = W(\mathtt{g}(\mathtt{f}(x)))$$

Yet, this system is non-terminating! Where is the error?

$[\mathtt{g}]$ is not monotonic.

So monotonicity really is essential.

# Automating monotonic algebras

# Automating monotonic algebras

Observation: “is there an interpretation that orients all the rules”

# Automating monotonic algebras

Observation: "is there an interpretation that orients all the rules" sounds like a SAT/SMT problem!

# Automating monotonic algebras

Observation: “is there an interpretation that orients all the rules” sounds like a SAT/SMT problem!

Challenge: specify “an interpretation” in (basic) SMT!

# Parametric interpretations

Idea: assign to function symbol f of arity $n$ a *parametric interpretation function* of a specific shape; for instance

$$\begin{array}{rcl}[0] & = & \underline{n} \\ [s] & = & \lambda x.\underline{s_0} + \underline{s_1} * x \\ [add] & = & \lambda x, y.\underline{a_0} + \underline{a_1} * x + \underline{a_2} * y\end{array}$$

# Parametric interpretations

Idea: assign to function symbol f of arity $n$ a *parametric interpretation function* of a specific shape; for instance

$$
\begin{aligned}
[\mathtt{0}] &= \underline{n} \\
[\mathtt{s}] &= \lambda x.\underline{s_0} + \underline{s_1} * x \\
[\mathtt{add}] &= \lambda x, y.\underline{a_0} + \underline{a_1} * x + \underline{a_2} * y
\end{aligned}
$$

Compute the resulting requirements for the rules:

$$
\begin{aligned}
W(\mathtt{add}(\mathtt{0}, y)) &= \underline{a_0} + \underline{a_1} * \underline{n} + \underline{a_2} * y \\
W(y) &= y \\
W(\mathtt{add}(\mathtt{s}(x), y)) &= \underline{a_0} + \underline{a_1} * (\underline{s_0} + \underline{s_1} * x) + \underline{a_2} * y \\
&= \underline{a_0} + \underline{a_1} * \underline{s_0} + \underline{a_1} * \underline{s_1} * x + \underline{a_2} * y \\
W(\mathtt{s}(\mathtt{add}(x, y))) &= \underline{s_0} + \underline{s_1} * (\underline{a_0} + \underline{a_1} * x + \underline{a_2} * y) \\
&= \underline{s_0} + \underline{s_1} * \underline{a_0} + \underline{s_1} * \underline{a_1} * x + \underline{s_1} * \underline{a_2} * y
\end{aligned}
$$

# Inequalities with (universally quantified) variables

We now have to solve a problem of the shape:

*find parameters such that:*

- *all* $[f]$ *are monotonic functions, and*
- *for all rules* $\ell \to r$*, all* $\vec{x}$*:* $W(\ell) \succ W(r)$*.*

# Inequalities with (universally quantified) variables

We now have to solve a problem of the shape:

*find parameters such that:*

- *all* $[f]$ *are monotonic functions, and*
- *for all rules* $\ell \to r$, *all* $\vec{x}$: $W(\ell) \succ W(r)$.

Requiring monotonicity in our example is not hard:

$$\begin{array}{rcl} [\mathtt{0}] & = & \underline{n} \\ [\mathtt{s}] & = & \lambda x.\underline{s_0} + \underline{s_1} * x \\ [\mathtt{add}] & = & \lambda x, y.\underline{a_0} + \underline{a_1} * x + \underline{a_2} * y \end{array}$$

# Inequalities with (universally quantified) variables

We now have to solve a problem of the shape:

*find parameters such that:*

- *all* $[f]$ *are monotonic functions, and*
- *for all rules* $\ell \to r$*, all* $\vec{x}$*:* $W(\ell) \succ W(r)$.

Requiring monotonicity in our example is not hard:

$$\begin{array}{rcl} [\mathtt{0}] & = & \underline{n} \\ [\mathtt{s}] & = & \lambda x.\underline{s_0} + \underline{s_1} * x \\ [\mathtt{add}] & = & \lambda x, y.\underline{a_0} + \underline{a_1} * x + \underline{a_2} * y \end{array}$$

We require that: $\underline{s_1} \geq 1$, $\underline{a_1} \geq 1$, $\underline{a_2} \geq 1$.

# Inequalities with (universally quantified) variables

We now have to solve a problem of the shape:

*find parameters such that:*

- *all* $[\mathtt{f}]$ *are monotonic functions, and*
- *for all rules* $\ell \to r$*, all* $\vec{x}$*:* $W(\ell) \succ W(r)$*.*

Requiring monotonicity in our example is not hard:

$$\begin{array}{rcl} [\mathtt{0}] & = & \underline{n} \\ [\mathtt{s}] & = & \lambda x.\underline{s_0} + \underline{s_1} * x \\ [\mathtt{add}] & = & \lambda x, y.\underline{a_0} + \underline{a_1} * x + \underline{a_2} * y \end{array}$$

We require that: $\underline{s_1} \geq 1$, $\underline{a_1} \geq 1$, $\underline{a_2} \geq 1$.

For the second requirement, we use **absolute positiveness**.

# Absolute positiveness

Goal: a *sufficient condition* to compare two polynomials

# Absolute positiveness

Goal: a *sufficient condition* to compare two polynomials

$$a_0 + a_1 * x_1 + \cdots + a_m * x_m > b_0 + b_1 * x_1 + \cdots + b_m * x_m$$

certainly holds if:

- $a_0 > b_0$
- each $a_i \geq b_i$

# Example (continued)

We must show that:

$$
\begin{aligned}
W(\mathtt{add}(\mathtt{0}, y)) &= \underline{a_0} + \underline{a_1} * \underline{n} + \underline{a_2} * y \\
&> y \\
&= W(y) \\
W(\mathtt{add}(\mathtt{s}(x), y)) &= \underline{a_0} + \underline{a_1} * \underline{s_0} + \underline{a_1} * \underline{s_1} * x + \underline{a_2} * y \\
&> \underline{s_0} + \underline{s_1} * \underline{a_0} + \underline{s_1} * \underline{a_1} * x + \underline{s_1} * \underline{a_2} * y \\
&= (\mathtt{s}(\mathtt{add}(x, y)))
\end{aligned}
$$

# Example (continued)

We must show that:

$$\begin{array}{rcl}
W(\mathtt{add}(\mathtt{0}, y)) & = & \underline{a_0} + \underline{a_1} * \underline{n} + \underline{a_2} * y \\
& > & y \\
& = & W(y) \\
W(\mathtt{add}(\mathtt{s}(x), y)) & = & \underline{a_0} + \underline{a_1} * \underline{s_0} + \underline{a_1} * \underline{s_1} * x + \underline{a_2} * y \\
& > & \underline{s_0} + \underline{s_1} * \underline{a_0} + \underline{s_1} * \underline{a_1} * x + \underline{s_1} * \underline{a_2} * y \\
& = & (\mathtt{s}(\mathtt{add}(x, y)))
\end{array}$$

That is:

$$(\underline{a_0} + \underline{a_1} * \underline{n}) + \underline{a_2} * y > y$$

and

$$(\underline{a_0}+\underline{a_1}*\underline{s_0})+(\underline{a_1}*\underline{s_1})*x+\underline{a_2}*y > (\underline{s_0}+\underline{s_1}*\underline{a_0})+(\underline{s_1}*\underline{a_1})*x+(\underline{s_1}*\underline{a_2})*y$$

# Example (continued)

That is:

$$(\underline{a_0} + \underline{a_1} * \underline{n}) + \underline{a_2} * y > y$$

and

$$(\underline{a_0}+\underline{a_1}*\underline{s_0})+(\underline{a_1}*\underline{s_1})*x+\underline{a_2}*y > (\underline{s_0}+\underline{s_1}*\underline{a_0})+(\underline{s_1}*\underline{a_1})*x+(\underline{s_1}*\underline{a_2})*y$$

# Example (continued)

That is:

$$(\underline{a_0} + \underline{a_1} * \underline{n}) + \underline{a_2} * y > y$$

and

$$(\underline{a_0}+\underline{a_1}*\underline{s_0})+(\underline{a_1}*\underline{s_1})*x+\underline{a_2}*y > (\underline{s_0}+\underline{s_1}*\underline{a_0})+(\underline{s_1}*\underline{a_1})*x+(\underline{s_1}*\underline{a_2})*y$$

Using absolute positiveness, it suffices if:

$$\begin{array}{rcl} \underline{a_0} + \underline{a_1} * \underline{n} & > & 0 \\ \\ \underline{a_2} & \geq & 1 \end{array} \qquad \begin{array}{rcl} \underline{a_0} + \underline{a_1} * \underline{s_0} & > & \underline{s_0} + \underline{s_1} * \underline{a_0} \\ \underline{a_1} * \underline{s_1} & \geq & \underline{s_1} * \underline{a_1} \\ \underline{a_2} & \geq & \underline{s_1} * \underline{a_2} \end{array}$$

# Completing the example

$$\begin{array}{rcl} \underline{a_0} + \underline{a_1} * \underline{n} & > & 0 \\ & & \\ \underline{a_2} & \geq & 1 \end{array} \qquad \begin{array}{rcl} \underline{a_0} + \underline{a_1} * \underline{s_0} & > & \underline{s_0} + \underline{s_1} * \underline{a_0} \\ \underline{a_1} * \underline{s_1} & \geq & \underline{s_1} * \underline{a_1} \\ \underline{a_2} & \geq & \underline{s_1} * \underline{a_2} \end{array} \qquad \begin{array}{rcl} \underline{s_1} & \geq & 1 \\ \underline{a_1} & \geq & 1 \\ \underline{a_2} & \geq & 1 \end{array}$$

# Completing the example

$$a_0 + a_1 * n > 0 \qquad a_2 \geq 1$$

$$\begin{array}{rcl} a_0 + a_1 * s_0 & > & s_0 + s_1 * a_0 \\ a_1 * s_1 & \geq & s_1 * a_1 \\ a_2 & \geq & s_1 * a_2 \end{array} \qquad \begin{array}{rcl} s_1 & \geq & 1 \\ a_1 & \geq & 1 \\ a_2 & \geq & 1 \end{array}$$

An SMT solver will for instance yield

$$n = 1,\ s_0 = 1,\ s_1 = 1,\ a_0 = 0,\ a_1 = 2,\ a_2 = 1$$

giving the same interpretations we had before:

$$\begin{array}{rcl} [\mathtt{0}] & = & n \\ [\mathtt{s}] & = & \lambda x. s_0 + s_1 * x \\ [\mathtt{add}] & = & \lambda x, y. a_0 + a_1 * x + a_2 * y \end{array} \Longrightarrow \begin{array}{rcl} [\mathtt{0}] & = & 1 \\ [\mathtt{s}] & = & \lambda x. 1 + x \\ [\mathtt{add}] & = & \lambda x, y. 2 * x + y \end{array}$$

# Completing the example

$$\underline{a_0} + \underline{a_1} * \underline{n} > 0 \qquad \underline{a_2} \geq 1$$

$$\begin{aligned} \underline{a_0} + \underline{a_1} * \underline{s_0} &> \underline{s_0} + \underline{s_1} * \underline{a_0} \\ \underline{a_1} * \underline{s_1} &\geq \underline{s_1} * \underline{a_1} \\ \underline{a_2} &\geq \underline{s_1} * \underline{a_2} \end{aligned}$$

$$\begin{aligned} \underline{s_1} &\geq 1 \\ \underline{a_1} &\geq 1 \\ \underline{a_2} &\geq 1 \end{aligned}$$

An SMT solver will for instance yield

$$\underline{n} = 1,\ \underline{s_0} = 1,\ \underline{s_1} = 1,\ \underline{a_0} = 0,\ \underline{a_1} = 2,\ \underline{a_2} = 1$$

giving the same interpretations we had before:

$$\begin{aligned} [\mathtt{0}] &= \underline{n} \\ [\mathtt{s}] &= \lambda x.\underline{s_0} + \underline{s_1} * x \\ [\mathtt{add}] &= \lambda x, y.\underline{a_0} + \underline{a_1} * x + \underline{a_2} * y \end{aligned} \quad \Longrightarrow \quad \begin{aligned} [\mathtt{0}] &= 1 \\ [\mathtt{s}] &= \lambda x.1 + x \\ [\mathtt{add}] &= \lambda x, y.2 * x + y \end{aligned}$$

Many other interpretations are also possible.

# Completing the example

$$\begin{array}{rcl} \underline{a_0} + \underline{a_1} * \underline{n} & > & 0 \\ \\ \underline{a_2} & \geq & 1 \end{array} \qquad \begin{array}{rcl} \underline{a_0} + \underline{a_1} * \underline{s_0} & > & \underline{s_0} + \underline{s_1} * \underline{a_0} \\ \underline{a_1} * \underline{s_1} & \geq & \underline{s_1} * \underline{a_1} \\ \underline{a_2} & \geq & \underline{s_1} * \underline{a_2} \end{array} \qquad \begin{array}{rcl} \underline{s_1} & \geq & 1 \\ \underline{a_1} & \geq & 1 \\ \underline{a_2} & \geq & 1 \end{array}$$

An SMT solver will for instance yield

$$\underline{n} = 1,\ \underline{s_0} = 1,\ \underline{s_1} = 1,\ \underline{a_0} = 0,\ \underline{a_1} = 2,\ \underline{a_2} = 1$$

giving the same interpretations we had before:

$$\begin{array}{rcl} [\mathtt{0}] & = & \underline{n} \\ [\mathtt{s}] & = & \lambda x.\underline{s_0} + \underline{s_1} * x \\ [\mathtt{add}] & = & \lambda x, y.\underline{a_0} + \underline{a_1} * x + \underline{a_2} * y \end{array} \Longrightarrow \begin{array}{rcl} [\mathtt{0}] & = & 1 \\ [\mathtt{s}] & = & \lambda x.1 + x \\ [\mathtt{add}] & = & \lambda x, y.2 * x + y \end{array}$$

Many other interpretations are also possible.

In practice: $\{0, 1, 2, 3\}$ usually suffice.

# Limitations

Absolute positiveness also works with combinations of variables, e.g.,

$$\underline{a} + \underline{b}y + \underline{c}xy + \underline{d}x^2y > \underline{e}x + \underline{d}y + \underline{f}xy$$

if $\underline{a} > 0 \wedge 0 \geq \underline{e} \wedge \underline{b} \geq \underline{d} \wedge \underline{c} \geq \underline{f} \wedge \underline{d} \geq 0$.

# Limitations

Absolute positiveness also works with combinations of variables, e.g.,

$$\underline{a} + \underline{b}y + \underline{c}xy + \underline{d}x^2y > \underline{e}x + \underline{d}y + \underline{f}xy$$

if $\underline{a} > 0 \wedge 0 \geq \underline{e} \wedge \underline{b} \geq \underline{d} \wedge \underline{c} \geq \underline{f} \wedge \underline{d} \geq 0$.

Warning: this is not a complete method!

# Limitations

Absolute positiveness also works with combinations of variables, e.g.,

$$\underline{a} + \underline{b}y + \underline{c}xy + \underline{d}x^2y > \underline{e}x + \underline{d}y + \underline{f}xy$$

if $\underline{a} > 0 \wedge 0 \geq \underline{e} \wedge \underline{b} \geq \underline{d} \wedge \underline{c} \geq \underline{f} \wedge \underline{d} \geq 0$.

Warning: this is not a complete method!

- absolute positiveness does not capture all inequalities; e.g., $x^2 \geq x$

# Limitations

Absolute positiveness also works with combinations of variables, e.g.,

$$\underline{a} + \underline{b}y + \underline{c}xy + \underline{d}x^2y > \underline{e}x + \underline{d}y + \underline{f}xy$$

if $\underline{a} > 0 \wedge 0 \geq \underline{e} \wedge \underline{b} \geq \underline{d} \wedge \underline{c} \geq \underline{f} \wedge \underline{d} \geq 0$.

Warning: this is not a complete method!

- absolute positiveness does not capture all inequalities; e.g., $x^2 \geq x$
- we might not guess the right interpretation shape

# Limitations example

$$\mathtt{mul}(\mathtt{s}(x), y) \to \mathtt{add}(y, \mathtt{mul}(x, y))$$

# Limitations example

$$\texttt{mul}(\texttt{s}(x), y) \to \texttt{add}(y, \texttt{mul}(x, y))$$

Interpretation shape:

$$[\texttt{f}] = \lambda x_1, \ldots, x_n.\underline{a_0} + \Sigma_{i=1}^{n} \underline{a_i} * x_i$$

Result: **FAIL**

# Limitations example

$$\mathtt{mul}(\mathtt{s}(x), y) \to \mathtt{add}(y, \mathtt{mul}(x, y))$$

Interpretation shape:

$$[\mathtt{f}] = \lambda x_1, \ldots, x_n . \underline{a_0} + \Sigma_{i=1}^{n} \underline{a_i} * x_i$$

Result: **FAIL**

Needed: a **quadratic** interpretation function:

$$[\mathtt{f}] = \lambda x_1, \ldots, x_n . \underline{a} + \Sigma_{i=1}^{n} \underline{b_i} * x_i + \Sigma_{i=1}^{n} \Sigma_{j=i}^{n} \underline{c_{ij}} * x_i * x_j$$

# Limitations example

$$\mathtt{mul}(\mathtt{s}(x), y) \to \mathtt{add}(y, \mathtt{mul}(x, y))$$

Interpretation shape:

$$[\mathtt{f}] = \lambda x_1, \ldots, x_n.\underline{a_0} + \Sigma_{i=1}^{n} \underline{a_i} * x_i$$

Result: **FAIL**

Needed: a **quadratic** interpretation function:

$$[\mathtt{f}] = \lambda x_1, \ldots, x_n.\underline{a} + \Sigma_{i=1}^{n} \underline{b_i} * x_i + \Sigma_{i=1}^{n} \Sigma_{j=i}^{n} \underline{c_{ij}} * x_i * x_j$$

Downside: more sophisticated shape = more complex SMT problem

# Mapping to **pairs** of numbers

Note: $\mathcal{A}$ is not **required** to be $\mathbb{N}$. Any well-founded set will do.

# Mapping to **pairs** of numbers

Note: $\mathcal{A}$ is not **required** to be $\mathbb{N}$. Any well-founded set will do.

Example: $\mathcal{A} = \mathbb{N}^2$, and $(x_1, y_1) > (x_2, y_2)$ if $x_1 > x_2$ and $y_1 \geq y_2$.

# Mapping to **pairs** of numbers

Note: $\mathcal{A}$ is not **required** to be $\mathbb{N}$. Any well-founded set will do.

Example: $\mathcal{A} = \mathbb{N}^2$, and $(x_1, y_1) > (x_2, y_2)$ if $x_1 > x_2$ and $y_1 \geq y_2$.

Typical uses:

- **matrix interpretations**

# Mapping to **pairs** of numbers

Note: $\mathcal{A}$ is not **required** to be $\mathbb{N}$. Any well-founded set will do.

Example: $\mathcal{A} = \mathbb{N}^2$, and $(x_1, y_1) > (x_2, y_2)$ if $x_1 > x_2$ and $y_1 \geq y_2$.

Typical uses:

- **matrix interpretations**
- **interpretations using max**

# Mapping to **pairs** of numbers

Note: $\mathcal{A}$ is not **required** to be $\mathbb{N}$. Any well-founded set will do.

Example: $\mathcal{A} = \mathbb{N}^2$, and $(x_1, y_1) > (x_2, y_2)$ if $x_1 > x_2$ and $y_1 \geq y_2$.

Typical uses:

- **matrix interpretations**
- **interpretations using max**

Example:

$$\mathtt{f}(\mathtt{s}(x)) \to \mathtt{f}(\mathtt{p}(\mathtt{s}(x))) \qquad \mathtt{p}(\mathtt{0}) \to \mathtt{0} \qquad \mathtt{p}(\mathtt{s}(x)) \to x$$

# Mapping to **pairs** of numbers

Note: $\mathcal{A}$ is not **required** to be $\mathbb{N}$. Any well-founded set will do.

Example: $\mathcal{A} = \mathbb{N}^2$, and $(x_1, y_1) > (x_2, y_2)$ if $x_1 > x_2$ and $y_1 \geq y_2$.

Typical uses:

- **matrix interpretations**
- **interpretations using max**

Example:

$$\mathtt{f}(\mathtt{s}(x)) \to \mathtt{f}(\mathtt{p}(\mathtt{s}(x))) \qquad \mathtt{p}(\mathtt{0}) \to \mathtt{0} \qquad \mathtt{p}(\mathtt{s}(x)) \to x$$

$$\begin{array}{rcl}
[\mathtt{0}] & = & \langle 0, 0 \rangle \\
[\mathtt{s}] & = & \lambda \langle x_1, x_2 \rangle . \langle x_1, x_2 + 1 \rangle \\
[\mathtt{p}] & = & \lambda \langle x_1, x_2 \rangle . \langle x_1, \max(x_2 - 1, 0) \rangle \\
[\mathtt{f}] & = & \lambda \langle x_1, x_2 \rangle . \langle x_1 + x_2, 0 \rangle
\end{array}$$

# Cost-size interpretations

Idea: first component = cost, second component = size

# Cost-size interpretations

Idea: first component = cost, second component = size

For example, in

$$\begin{array}{rcl} \mathtt{add}(\mathtt{0}, y) & \to & y \\ \mathtt{add}(\mathtt{s}(x), y) & \to & \mathtt{s}(\mathtt{add}(x, y)) \end{array}$$

we may choose:

$$\begin{array}{rcl} [\mathtt{0}] & = & \langle 0, 0 \rangle \\ [\mathtt{s}] & = & \lambda \langle x_1, x_2 \rangle . \langle x_1, x_2 + 1 \rangle \\ [\mathtt{add}] & = & \lambda \langle x_1, x_2 \rangle, \langle y_1, y_2 \rangle . \langle x_1 + x_2 + y_1, x_2 + y_2 \rangle \end{array}$$

# Cost-size interpretations

Idea: first component = cost, second component = size

For example, in

$$\begin{array}{rcl} \mathtt{add}(\mathtt{0}, y) & \to & y \\ \mathtt{add}(\mathtt{s}(x), y) & \to & \mathtt{s}(\mathtt{add}(x, y)) \end{array}$$

we may choose:

$$\begin{array}{rcl} [\mathtt{0}] & = & \langle 0, 0 \rangle \\ [\mathtt{s}] & = & \lambda \langle x_1, x_2 \rangle . \langle x_1, x_2 + 1 \rangle \\ [\mathtt{add}] & = & \lambda \langle x_1, x_2 \rangle, \langle y_1, y_2 \rangle . \langle x_1 + x_2 + y_1, x_2 + y_2 \rangle \end{array}$$

This is also valuable in **complexity analysis** of programs.

# Alternative interpretation domains

Other domains used for monotonic algebras include:

# Alternative interpretation domains

Other domains used for monotonic algebras include:

- $\mathbb{N}^k$ for any number $k$

# Alternative interpretation domains

Other domains used for monotonic algebras include:

- $\mathbb{N}^k$ for any number $k$
- rational or real numbers (where "$a > b$" if $a \geq b + \epsilon$ for some fixed number $\epsilon$)

# Alternative interpretation domains

Other domains used for monotonic algebras include:

- $\mathbb{N}^k$ for any number $k$
- rational or real numbers (where "$a > b$" if $a \geq b + \epsilon$ for some fixed number $\epsilon$)
- integer numbers above some bound, e.g., $\{k, k+1, \dots\}$, where $k$ may be positive or negative

# Alternative interpretation domains

Other domains used for monotonic algebras include:

- $\mathbb{N}^k$ for any number $k$
- rational or real numbers (where "$a > b$" if $a \geq b + \epsilon$ for some fixed number $\epsilon$)
- integer numbers above some bound, e.g., $\{k, k+1, \dots\}$, where $k$ may be positive or negative
- sets of terms terminating under some different well-founded ordering $\succ$

# Subterm property

**Definition**

*subterm property*

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

# Subterm property

**Definition**

*subterm property*

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

All recursive path orderings have the subterm property, as do monotonic algebras to $\mathbb{N}$.

# Subterm property

**Definition**

*subterm property*

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

All recursive path orderings have the subterm property, as do monotonic algebras to $\mathbb{N}$.

The good:

- intuitive: a term is bigger than its subterms
- essential property for instance in the soundness proof of superposition

# Subterm property

**Definition**

*subterm property*

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

All recursive path orderings have the subterm property, as do monotonic algebras to $\mathbb{N}$.

The good:

- intuitive: a term is bigger than its subterms
- essential property for instance in the soundness proof of superposition

The bad:

- Not all TRSs can be ordered this way!

# Motivating example

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

# Motivating example

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

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

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

# Motivating example

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

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

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

# Motivating example

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

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

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

Hence, if the last rule is oriented with $\succ$, then by stability:

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

# Motivating example

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

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

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

Hence, if the last rule is oriented with $\succ$, then by stability:

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

Hence, this system **cannot** be ordered using any recursive path ordering, or an interpretation to $\mathbb{N}$.

# 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, incuding 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, incuding 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, incuding 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, incuding 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

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

# Identifying function calls

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

We isolate the calls from one defined symbol to another:

$$
\begin{array}{rcl}
\mathtt{minus}^\sharp(\mathtt{s}(x), \mathtt{s}(y)) & \Rightarrow & \mathtt{minus}^\sharp(x, y) \\
\mathtt{quot}^\sharp(\mathtt{s}(x), \mathtt{s}(y)) & \Rightarrow & \mathtt{quot}^\sharp(\mathtt{minus}(x, y), \mathtt{s}(y)) \\
\mathtt{quot}^\sharp(\mathtt{s}(x), \mathtt{s}(y)) & \Rightarrow & \mathtt{minus}^\sharp(x, y)
\end{array}
$$

# Dependency pair chains

We can prove the following result:

**Theorem**

For a given set of rules $R$, let $\mathtt{DP}$ be the corresponding set of dependency pairs.

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

# Dependency pair chains

We can prove the following result:

**Theorem**

For a given set of rules $R$, let $\mathtt{DP}$ be the corresponding set of dependency pairs.

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

Note: marking prevents $\Rightarrow_{\mathcal{R}}$ steps at the top!

# Dependency pair chains

We can prove the following result:

**Theorem**

For a given set of rules $R$, let $\mathtt{DP}$ be the corresponding set of dependency pairs.

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

Note: marking prevents $\Rightarrow_{\mathcal{R}}$ steps at the top!

So $\Rightarrow_{\mathcal{R}}$ is terminating iff there is no infinite sequence where:

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

# Example: an infinite DP chain

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

# Example: an infinite DP chain

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

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

# Example: an infinite DP chain

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

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

**Infinite chain:**

$$\underline{\mathtt{f}^\sharp(\mathtt{0}, \mathtt{0})} \Rightarrow_{\mathtt{DP}} \mathtt{f}^\sharp(\underline{\mathtt{g}(\mathtt{0})}, \mathtt{0}) \Rightarrow_{\mathcal{R}} \underline{\mathtt{f}^\sharp(\mathtt{0}, \mathtt{0})} \Rightarrow_{\mathtt{DP}} \mathtt{f}^\sharp(\underline{\mathtt{g}(\mathtt{0})}, \mathtt{0}) \Rightarrow_{\mathcal{R}} \ldots$$

# Building block: reduction pairs

How do you prove that there is no infinite 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 $\ell \Rightarrow r \in \mathtt{DP}$,

# 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 $\ell \Rightarrow r \in \mathtt{DP}$,
- $\ell \succeq r$ 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 $\ell \Rightarrow r \in \mathtt{DP}$,
- $\ell \succeq r$ 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 $\ell \Rightarrow r \in \mathtt{DP}$,
- $\ell \succeq r$ 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 $\ell \Rightarrow r \in \mathtt{DP}$,
- $\ell \succeq r$ 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 $\ell \Rightarrow r \in \mathtt{DP}$,
- $\ell \succeq r$ for all rules $\ell \rightarrow r \in R$,
- $\succ$ 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 $(\mathtt{DP}, R)$-chain!

# Building block: reduction pairs

**Theorem**

If:

- $\ell \succ r$ for all dependency pairs $\ell \Rightarrow r \in \mathtt{DP}$,
- $\ell \succeq r$ 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 $(\mathtt{DP}, R)$-chain!

**Proof idea.** By these requirements:

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

□

# Quot/minus: ordering requirements

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

# Weakly monotonic algebras

# Weakly monotonic algebras

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

# Weakly monotonic algebras

Like monotonic algebras, but $[\mathtt{f}]$ only needs to be **weakly monotonic.**

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

# Weakly monotonic algebras

Like monotonic algebras, but $[\mathtt{f}]$ only needs to be **weakly monotonic.**

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

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

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

# Completing quot/minus

Back to our example!

# Completing quot/minus

Back to our example!

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

# Completing quot/minus

Back to our example!

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

This is satisfied by choosing:

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

# 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:

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

# 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:

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

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:

$$
\begin{array}{llrcl}
\textbf{Rules:} & & \mathtt{minus}(x, \mathtt{0}) & \Rightarrow & x \\
& & \mathtt{minus}(\mathtt{s}(x), \mathtt{s}(y)) & \Rightarrow & \mathtt{minus}(x, y) \\
& & \mathtt{quot}(\mathtt{0}, \mathtt{s}(y)) & \Rightarrow & \mathtt{0} \\
& & \mathtt{quot}(\mathtt{s}(x), \mathtt{s}(y)) & \Rightarrow & \mathtt{s}(\mathtt{quot}(\mathtt{minus}(x, y), \mathtt{s}(y))) \\
\textbf{DPs:} & \mathrm{A} & \mathtt{minus}^\sharp(\mathtt{s}(x), \mathtt{s}(y)) & \Rightarrow & \mathtt{minus}^\sharp(x, y) \\
& \mathrm{B} & \mathtt{quot}^\sharp(\mathtt{s}(x), \mathtt{s}(y)) & \Rightarrow & \mathtt{quot}^\sharp(\mathtt{minus}(x, y), \mathtt{s}(y)) \\
& \mathrm{C} & \mathtt{quot}^\sharp(\mathtt{s}(x), \mathtt{s}(y)) & \Rightarrow & \mathtt{minus}^\sharp(x, y)
\end{array}
$$

# Step-by-step proofs

Challenge: dealing with **large** systems.

Idea: split large problems into smaller sub-problems.

Example:

**Rules:**

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

**DPs:**

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

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:**

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

**DPs:**

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

Question: what does an infinite chain look like?

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

# Mandatory material ends here

The material before this slide is expected knowledge, and you should be able to use monotonic algebras, and dependency pairs with weakly monotonic algebras, on the exam. You should also know the overall idea of the DP framework.

The following slides are optional material. However, you are *allowed* to use it on the exam; for example to answer a question “prove termination of this system”, which can often be done much faster using the dependency pair framework with the graph and subterm criterion (see subsequent slides).

# 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:

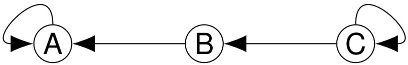
$$
\begin{array}{llcl}
\text{A.} & \mathtt{minus}^\sharp(\mathtt{s}(x), \mathtt{s}(y)) & \Rightarrow & \mathtt{minus}^\sharp(x, y) \\
\text{B.} & \mathtt{quot}^\sharp(\mathtt{s}(x), \mathtt{s}(y)) & \Rightarrow & \mathtt{minus}^\sharp(x, y) \\
\text{C.} & \mathtt{quot}^\sharp(\mathtt{s}(x), \mathtt{s}(y))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

# Using a graph

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

**Example:**

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

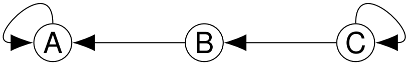


# Using a graph

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

Example:

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




Observation: each **strongly connected component** may be considered separately.

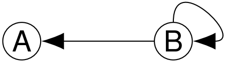
# Dependency graph processor – another example

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

# Dependency graph processor – another example

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

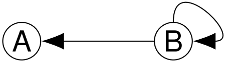
$$\mathtt{DP} = \left\{ \begin{array}{lrcl} \text{A.} & \mathtt{f}^\sharp(\mathtt{f}(x)) & \Rightarrow & \mathtt{f}^\sharp(\mathtt{g}(\mathtt{f}(x))) \\ \text{B.} & \mathtt{f}^\sharp(\mathtt{f}(x)) & \Rightarrow & \mathtt{f}^\sharp(x) \end{array} \right\}$$



# Dependency graph processor – another example

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

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

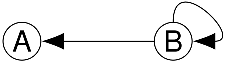



Hence, we can remove dependency pair A.

# Dependency graph processor – another example

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

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




Hence, we can remove dependency pair A.

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

# Dependency graph processor

Formally:

# Dependency graph processor

Formally:

**Definition**

Let $(\mathcal{D}, R)$ be a DP problem, and $G$ a graph whose nodes are the elements of $\mathcal{D}$, and which has an edge from $\rho$ to $\mu$ if it is possible for $\rho$ to be followed by $\mu$ in a $(\mathcal{D}, R)$-chain (there may be more edges than this).

Suppose $A_1, \ldots, A_n$ are the **strongly connected components** of $G$.

Then the dependency graph processor maps $(\mathcal{D}, R)$ to $\{(A_1, R), \ldots, (A_n, R)\}$.

# The subterm criterion

A. $\mathtt{exp}^\sharp(\mathtt{s}(x), y) \Rightarrow \mathtt{double}^\sharp(x, y, \mathtt{0})$

B. $\mathtt{double}^\sharp(x, \mathtt{0}, z) \Rightarrow \mathtt{exp}^\sharp(x, z)$

C. $\mathtt{double}^\sharp(x, \mathtt{0}, z) \Rightarrow \mathtt{double}^\sharp(x, y, \mathtt{s}(\mathtt{s}(z)))$

# The subterm criterion

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

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

# The subterm criterion

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

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

# The subterm criterion

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

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 $\mathtt{f}^\sharp$, let $\nu(\mathtt{f}^\sharp) \in \{1, \dots, arity(\mathtt{f})\}$.

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

Suppose:

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

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

# The subterm criterion processor

**Definition**

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

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

Suppose:

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

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

Implementation: a simple SMT implementation using integer variables $\nu(\mathtt{f}^\sharp)$, and boolean variables $\mathtt{strict}_\rho$.

# Graph + subterm criterion

Class exercise:

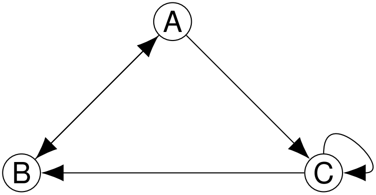
$$
\begin{array}{rcl}
\mathtt{exp}(\mathtt{0}, y) & \Rightarrow & y \\
\mathtt{exp}(\mathtt{s}(x), y) & \Rightarrow & \mathtt{double}(\mathtt{0}, x, y) \\
\mathtt{double}(r, x, \mathtt{0}) & \Rightarrow & \mathtt{exp}(x, r) \\
\mathtt{double}(r, x, \mathtt{s}(y)) & \Rightarrow & \mathtt{double}(\mathtt{s}(\mathtt{s}(r)),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

# Graph + subterm criterion

## Class exercise:

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

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

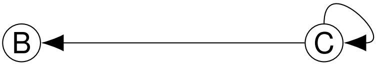


# Graph + subterm criterion

## Class exercise:

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

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

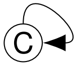


# Graph + subterm criterion

## Class exercise:

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

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



# Reduction pair processor

We can also reformulate reduction pairs as a processor:

**Definition**

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**

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 $\succ$.

# Example: using a reduction pair processor

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

# Example: using a reduction pair processor

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

We choose:

$$
\begin{array}{rcl}
[\mathtt{nil}] & = & 0 \\
[\mathtt{cons}] & = & \lambda x, y.\ y + 1
\end{array}
$$

$$
\begin{array}{rcll}
[\mathtt{append}] & = & \lambda x, y. & x + y \\
[\mathtt{rev}] & = & \lambda x. & x \\
[\mathtt{append}^\sharp] & = & \lambda x, y. & x + y \\
[\mathtt{rev}^\sharp] & = & \lambda x. & x
\end{array}
$$

# Example: 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:

$$\begin{array}{rcl} [\mathtt{nil}] & = & 0 \\ [\mathtt{cons}] & = & \lambda x,y.\ y+1 \end{array}$$

$$\begin{array}{rcll} [\mathtt{append}] & = & \lambda x,y. & x+y \\ [\mathtt{rev}] & = & \lambda x. & x \\ [\mathtt{append}^\sharp] & = & \lambda x,y. & x+y \\ [\mathtt{rev}^\sharp] & = & \lambda x. & x \end{array}$$

# Example: 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:

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

# Argument filterings

Idea: remove arguments before using a reduction ordering

# Argument filterings

Idea: remove arguments before using a reduction ordering

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

# Argument filterings

Idea: remove arguments before using a reduction ordering

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

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

# Argument filterings

Idea: remove arguments before using a reduction ordering

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

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

# Argument filterings

Idea: remove arguments before using a reduction ordering

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

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

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

# Argument filterings

Idea: remove arguments before using a reduction ordering

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

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

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

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.

$$\mathtt{quot}^\sharp(\mathtt{s}(x), \mathtt{s}(y)) \succ \mathtt{quot}^\sharp(\mathtt{minus}(x, y), \mathtt{s}(y))$$

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

# Usable rules

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

$$\mathtt{quot}^\sharp(\mathtt{s}(x),\mathtt{s}(y)) \succ \mathtt{quot}^\sharp(\underline{\mathtt{minus}(x,y)},\mathtt{s}(y))$$

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

# Usable rules

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

$$\mathtt{quot}^\sharp(\mathtt{s}(x), \mathtt{s}(y)) \succ \mathtt{quot}^\sharp(\mathtt{minus}(x, y), \mathtt{s}(y))$$

$$\begin{aligned}
\mathtt{minus}(x, \mathtt{0}) &\succeq x \\
\mathtt{minus}(\mathtt{s}(x), \mathtt{s}(y)) &\succeq \underline{\mathtt{minus}(x, y)} \\
\mathtt{quot}(\mathtt{0}, \mathtt{s}(y)) &\succeq \mathtt{0} \\
\mathtt{quot}(\mathtt{s}(x), \mathtt{s}(y)) &\succeq \mathtt{s}(\mathtt{quot}(\mathtt{minus}(x, y), \mathtt{s}(y)))
\end{aligned}$$

# Usable rules

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

$$\mathtt{quot}^\sharp(\mathtt{s}(x), \mathtt{s}(y)) \succ \mathtt{quot}^\sharp(\mathtt{minus}(x, y), \mathtt{s}(y))$$

$$\begin{aligned} \mathtt{minus}(x, \mathtt{0}) &\succeq x \\ \mathtt{minus}(\mathtt{s}(x), \mathtt{s}(y)) &\succeq \mathtt{minus}(x, y) \end{aligned}$$

# Usable rules

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

$$\mathtt{quot}^\sharp(\mathtt{s}(x),\mathtt{s}(y)) \quad \succ \quad \mathtt{quot}^\sharp(\mathtt{minus}(x,y),\mathtt{s}(y))$$

$$\begin{array}{rcl} \mathtt{minus}(x,\mathtt{0}) & \succeq & x \\ \mathtt{minus}(\mathtt{s}(x),\mathtt{s}(y)) & \succeq & \mathtt{minus}(x,y) \end{array}$$

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

# Usable rules

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

$$\mathtt{quot}^\sharp(\mathtt{s}(x), \mathtt{s}(y)) \succ \mathtt{quot}^\sharp(\mathtt{minus}(x, y), \mathtt{s}(y))$$

$$\begin{aligned} \mathtt{minus}(x, \mathtt{0}) &\succeq x \\ \mathtt{minus}(\mathtt{s}(x), \mathtt{s}(y)) &\succeq \mathtt{minus}(x, y) \end{aligned}$$

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

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.

$$\mathtt{quot}^\sharp(\mathtt{s}(x), \mathtt{s}(y)) \succ \mathtt{quot}^\sharp(\mathtt{minus}(x, y), \mathtt{s}(y))$$

$$\begin{aligned} \mathtt{minus}(x, \mathtt{0}) &\succeq x \\ \mathtt{minus}(\mathtt{s}(x), \mathtt{s}(y)) &\succeq \mathtt{minus}(x, y) \end{aligned}$$

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

Finding usable rules is a very simple reachability algorithm.

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

# Quiz

1. Prove termination of the following TRS using a monotonic algebra to $\mathbb{N}$:

$$\begin{array}{rcl} \mathtt{append}(\mathtt{nil}, z) & \to & z \\ \mathtt{append}(\mathtt{cons}(x, y), z) & \to & \mathtt{cons}(x, \mathtt{append}(y, z)) \end{array}$$

   - give (linear) parametric interpretations for all symbols
   - compute the requirements (monotonicity and rule orientation)
   - use absolute positiveness to find SMT requirements
   - solve them by hand and give the resulting inteprrretation functions, and check your result!

2. Determine the dependency pairs of:

$$\begin{array}{rcl} \mathtt{f}(\mathtt{h}(x), y) & \to & \mathtt{g}(x, \mathtt{f}(x, \mathtt{h}(y))) \\ \mathtt{g}(x, \mathtt{h}(y)) & \to & \mathtt{g}(\mathtt{h}(x), y) \end{array}$$

3. Split these dependency pairs up into one or more groups of DPs that can be analysed separately.