Therefore, $v = u\sigma$ for some non-variable subterm u of r. As $v \in T_\infty$, we see that $\mathrm{root}(u) = \mathrm{root}(v) \in D$. Moreover, u cannot be a proper subterm of l, since otherwise again there would be an infinite derivation starting from some t_i.

Putting everything together, we obtain

$$t = f(t_1, \dots, t_n) \xrightarrow{>\varepsilon}{}^*_R f(s_1, \dots, s_n) = l\sigma \xrightarrow{\varepsilon}_R r\sigma \trianglerighteq u\sigma$$

where $r \trianglerighteq u$, u is not a variable, $\mathrm{root}(u) \in D$, $l \not\triangleright u$.

Since $u\sigma \in T_\infty$, we can continue this process and obtain an infinite sequence.

If we define $S := \{\, l \to u \mid l \to r \in R,\ r \trianglerighteq u,\ u \notin X,\ \mathrm{root}(u) \in D,\ l \not\triangleright u \,\}$, we can combine the rewrite step at the root and the subterm step and obtain

$$t \xrightarrow{>\varepsilon}{}^*_R l\sigma \xrightarrow{\varepsilon}_S u\sigma.$$

To get rid of the superscripts ε and $>\varepsilon$, it turns out to be useful to introduce a new set of function symbols $f^\sharp$ that are only used for the root symbols of this derivation:

$$\Omega^\sharp := \{\, f^\sharp/n \mid f/n \in \Omega \,\}.$$

For a term $t = f(t_1, \dots, t_n)$ we define $t^\sharp := f^\sharp(t_1, \dots, t_n)$; for a set of terms T we define $T^\sharp := \{\, t^\sharp \mid t \in T \,\}$.

The set of *dependency pairs* of a TRS R is then defined by

$$\mathrm{DP}(R) := \{\, l^\sharp \to u^\sharp \mid l \to r \in R,\ r \trianglerighteq u,\ u \notin X,\ \mathrm{root}(u) \in D,\ l \not\triangleright u \,\}.$$

For $t \in T_\infty$, the sequence using the S-rule corresponds now to

$$t^\sharp \to^*_R l^\sharp\sigma \to_{\mathrm{DP}(R)} u^\sharp\sigma$$

where $t^\sharp \in T^\sharp_\infty$ and $u^\sharp\sigma \in T^\sharp_\infty$.

(Note that rules in R do not contain symbols from $\Omega^\sharp$, whereas all roots of terms in $\mathrm{DP}(R)$ come from $\Omega^\sharp$, so rules from R can only be applied below the root and rules from $\mathrm{DP}(R)$ can only be applied at the root.)

Since $u^\sharp\sigma$ is again in $T^\sharp_\infty$, we can continue the process in the same way. We obtain: R is non-terminating iff there is an infinite sequence

$$t_1 \to^*_R t_2 \to_{\mathrm{DP}(R)} t_3 \to^*_R t_4 \to_{\mathrm{DP}(R)} \cdots$$

with $t_i \in T^\sharp_\infty$ for all i.

Moreover, if there exists such an infinite sequence, then there exists an infinite sequence in which all DPs that are used are used infinitely often. (If some DP is used only finitely often, we can cut off the initial part of the sequence up to the last occurrence of that DP; the remainder is still an infinite sequence.)

Dependency Graphs

Such infinite sequences correspond to "cycles" in the "dependency graph":

Dependency graph $\mathrm{DG}(R)$ of a TRS R:

directed graph

nodes: dependency pairs $s \to t \in \mathrm{DP}(R)$

edges: from $s \to t$ to $u \to v$ if there are σ, τ such that $t\sigma \to_R^* u\tau$.

Intuitively, we draw an edge between two dependency pairs, if these two dependency pairs can be used after another in an infinite sequence (with some R-steps in between). While this relation is undecidable in general, there are reasonable overapproximations:

The functions cap and ren are defined by:

$$\begin{aligned} \mathrm{cap}(x) &= x \\ \mathrm{cap}(f(t_1,\dots,t_n)) &= \begin{cases} y & \text{if } f \in D \\ f(\mathrm{cap}(t_1),\dots,\mathrm{cap}(t_n)) & \text{if } f \in C \cup D^\sharp \end{cases} \end{aligned}$$

$$\begin{aligned} \mathrm{ren}(x) &= y, \quad y \text{ fresh} \\ \mathrm{ren}(f(t_1,\dots,t_n)) &= f(\mathrm{ren}(t_1),\dots,\mathrm{ren}(t_n)) \end{aligned}$$

The overapproximated dependency graph contains an edge from $s \to t$ to $u \to v$ if $\mathrm{ren}(\mathrm{cap}(t))$ and u are unifiable.

A *cycle* in the dependency graph is a non-empty subset $K \subseteq \mathrm{DP}(R)$ such that there is a non-empty path in K from every DP in K to every DP in K (the two DPs may be identical).

Let $K \subseteq \mathrm{DP}(R)$. An infinite rewrite sequence in $R \cup K$ of the form

$$t_1 \to_R^* t_2 \to_K t_3 \to_R^* t_4 \to_K \dots$$

with $t_i \in T_\infty^\sharp$ is called K-minimal, if all rules in K are used infinitely often.

R is non-terminating iff there is a cycle $K \subseteq \mathrm{DP}(R)$ and a K-minimal infinite rewrite sequence.

5.2 Subterm Criterion

Our task is to show that there are no K-minimal infinite rewrite sequences.

Suppose that every dependency pair symbol $f^\sharp$ in K has positive arity (i. e., no constants). A *simple projection* π is a mapping $\pi : \Omega^\sharp \to \mathbb{N}$ such that $\pi(f^\sharp) = i \in \{1,\dots,\mathrm{arity}(f^\sharp)\}$.

We define $\pi(f^\sharp(t_1,\dots,t_n)) = t_{\pi(f^\sharp)}$.

Theorem 5.1 (Hirokawa and Middeldorp) *Let K be a cycle in $\mathrm{DG}(R)$. If there is a simple projection π for K such that $\pi(l) \trianglerighteq \pi(r)$ for every $l \to r \in K$ and $\pi(l) \triangleright \pi(r)$ for some $l \to r \in K$, then there are no K-minimal sequences.*

Proof. Suppose that

$$t_1 \to_R^* u_1 \to_K t_2 \to_R^* u_2 \to_K \dots$$

is a K-minimal infinite rewrite sequence. Apply π to every t_i:

Case 1: $u_i \to_K t_{i+1}$. There is an $l \to r \in K$ such that $u_i = l\sigma$, $t_{i+1} = r\sigma$. Then $\pi(u_i) = \pi(l)\sigma$ and $\pi(t_{i+1}) = \pi(r)\sigma$. By assumption, $\pi(l) \trianglerighteq \pi(r)$. If $\pi(l) = \pi(r)$, then $\pi(u_i) = \pi(t_{i+1})$. If $\pi(l) \triangleright \pi(r)$, then $\pi(u_i) = \pi(l)\sigma \triangleright \pi(r)\sigma = \pi(t_{i+1})$. In particular, $\pi(u_i) \triangleright \pi(t_{i+1})$ for infinitely many i (since every DP is used infinitely often).

Case 2: $t_i \to_R^* u_i$. Then $\pi(t_i) \to_R^* \pi(u_i)$.

By applying π to every term in the K-minimal infinite rewrite sequence, we obtain an infinite $(\to_R \cup \triangleright)$-sequence containing infinitely many $\triangleright$-steps. Since $\triangleright$ is well-founded, there must also exist infinitely many $\to_R$-steps (otherwise the infinite sequence would have an infinite tail consisting only of $\triangleright$-steps, contradicting well-foundedness.)

Now note that $\triangleright \circ \to_R \subseteq \to_R \circ \triangleright$. Therefore we can commute $\triangleright$-steps and $\to_R$-steps and move all $\to_R$-steps to the front. We obtain an infinite $\to_R$-sequence that starts with $\pi(t_1)$. However $t_1 \triangleright \pi(t_1)$ and $t_1 \in T_\infty^\sharp$, so there cannot be an infinite $\to_R$-sequence starting from $\pi(t_1)$. □

Problem: The number of cycles in $\mathrm{DG}(R)$ can be exponential.

Better method: Analyze strongly connected components (SCCs).

SCC of a graph: maximal subgraph in which there is a non-empty path from every node to every node. (The two nodes can be identical.)[3]

Important property: Every cycle is contained in some SCC.

Idea: Search for a simple projection π such that $\pi(l) \trianglerighteq \pi(r)$ for all DPs $l \to r$ in the SCC. Delete all DPs in the SCC for which $\pi(l) \triangleright \pi(r)$ (by the previous theorem, there cannot be any K-minimal infinite rewrite sequences using these DPs). Then re-compute SCCs for the remaining graph and re-start.

No SCCs left ⇒ no cycles left ⇒ R is terminating.

Example: See Ex. 13 from Hirokawa and Middeldorp.

[3] There are several definitions of SCCs that differ in the treatment of edges from a node to itself.

5.3 Reduction Pairs and Argument Filterings

Goal: Show the non-existence of K-minimal infinite rewrite sequences

$$t_1 \rightarrow_R^* u_1 \rightarrow_K t_2 \rightarrow_R^* u_2 \rightarrow_K \dots$$

using well-founded orderings.

We observe that the requirements for the orderings used here are less restrictive than for reduction orderings:

K-rules are only used at the top, so we need stability under substitutions, but compatibility with contexts is unnecessary.

While $\rightarrow_K$-steps should be decreasing, for $\rightarrow_R$-steps it would be sufficient to show that they are not increasing.

This motivates the following definitions:

Rewrite quasi-ordering $\succsim$:

reflexive and transitive binary relation, stable under substitutions, compatible with contexts.

Reduction pair $(\succsim, \succ)$:

$\succsim$ is a rewrite quasi-ordering.

$\succ$ is a well-founded ordering that is stable under substitutions.

$\succsim$ and $\succ$ are compatible: $\succsim \circ \succ \subseteq \succ$ or $\succ \circ \succsim \subseteq \succ$.

(In practice, $\succ$ is almost always the strict part of the quasi-ordering $\succsim$.)

Clearly, for any reduction ordering $\succ$, $(\succeq, \succ)$ is a reduction pair. More general reduction pairs can be obtained using argument filterings:

Argument filtering π:

$\pi : \Omega \cup \Omega^\sharp \rightarrow \mathbb{N} \cup \text{list of } \mathbb{N}$

$$\pi(f) = \begin{cases} i \in \{1, \dots, \text{arity}(f)\}, \text{ or} \\ [i_1, \dots, i_k], \text{ where } 1 \leq i_1 < \dots < i_k \leq \text{arity}(f),\ 0 \leq k \leq \text{arity}(f) \end{cases}$$

Extension to terms:

$\pi(x) = x$

$\pi(f(t_1, \dots, t_n)) = \pi(t_i)$, if $\pi(f) = i$

$\pi(f(t_1, \dots, t_n)) = f'(\pi(t_{i_1}), \dots, \pi(t_{i_k}))$, if $\pi(f) = [i_1, \dots, i_k]$,
where f'/k is a new function symbol.

Let $\succ$ be a reduction ordering, let π be an argument filtering. Define $s \succ_\pi t$ iff $\pi(s) \succ \pi(t)$ and $s \succsim_\pi t$ iff $\pi(s) \succeq \pi(t)$.

Lemma 5.2 *$(\succsim_\pi, \succ_\pi)$ is a reduction pair.*

Proof. Follows from the following two properties:

$\pi(s\sigma) = \pi(s)\sigma_\pi$, where σ_π is the substitution that maps x to $\pi(\sigma(x))$.

$$\pi(s[u]_p) = \begin{cases} \pi(s), & \text{if } p \text{ does not correspond to any position in } \pi(s) \\ \pi(s)[\pi(u)]_q, & \text{if } p \text{ corresponds to } q \text{ in } \pi(s) \end{cases}$$ □

For interpretation-based orderings (such as polynomial orderings) the idea of "cutting out" certain subterms can be included directly in the definition of the ordering:

Reduction pairs by interpretation:

Let $\mathcal{A}$ be a Σ-algebra; let $\succ$ be a well-founded strict partial ordering on its universe.

Assume that all interpretations $f_\mathcal{A}$ of function symbols are *weakly monotone*, i. e., $a_i \succeq b_i$ implies $f(a_1, \dots, , a_n) \succeq f(b_1, \dots, b_n)$ for all $a_i, b_i \in U_\mathcal{A}$.

Define $s \succsim_\mathcal{A} t$ iff $\mathcal{A}(\beta)(s) \succeq \mathcal{A}(\beta)(t)$ for all assignments $\beta : X \to U_\mathcal{A}$; define $s \succ_\mathcal{A} t$ iff $\mathcal{A}(\beta)(s) \succ \mathcal{A}(\beta)(t)$ for all assignments $\beta : X \to U_\mathcal{A}$.

Then $(\succsim_\mathcal{A}, \succ_\mathcal{A})$ is a reduction pair.

For polynomial orderings, this definition permits interpretations of function symbols where some variable does not occur at all (e. g., $P_f(X_1, X_2) = 2X_1 + 1$ for a *binary* function symbol). It is no longer required that *every* variable must occur with some positive coefficient.

Theorem 5.3 (Arts and Giesl) *Let K be a cycle in the dependency graph of the TRS R. If there is a reduction pair $(\succsim, \succ)$ such that*

- *$l \succsim r$ for all $l \to r \in R$,*
- *$l \succsim r$ or $l \succ r$ for all $l \to r \in K$,*
- *$l \succ r$ for at least one $l \to r \in K$,*

then there is no K-minimal infinite sequence.

Proof. Assume that

$$t_1 \to_R^* u_1 \to_K t_2 \to_R^* u_2 \to_K \dots$$

is a K-minimal infinite rewrite sequence.

As $l \succsim r$ for all $l \to r \in R$, we obtain $t_i \succsim u_i$ by stability under substitutions, compatibility with contexts, reflexivity and transitivity.

As $l \succsim r$ or $l \succ r$ for all $l \to r \in K$, we obtain u_i $(\succsim \cup \succ)$ t_{i+1} by stability under substitutions.

So we get an infinite $(\succsim \cup \succ)$-sequence containing infinitely many $\succ$-steps (since every DP in K, in particular the one for which $l \succ r$ holds, is used infinitely often).

By compatibility of $\succsim$ and $\succ$, we can transform this into an infinite $\succ$-sequence, contradicting well-foundedness. □

The idea can be extended to SCCs in the same way as for the subterm criterion:

Search for a reduction pair $(\succsim, \succ)$ such that $l \succsim r$ for all $l \to r \in R$ and $l \succsim r$ or $l \succ r$ for all DPs $l \to r$ in the SCC. Delete all DPs in the SCC for which $l \succ r$. Then re-compute SCCs for the remaining graph and re-start.

Example: Consider the following TRS R from [Arts and Giesl]:

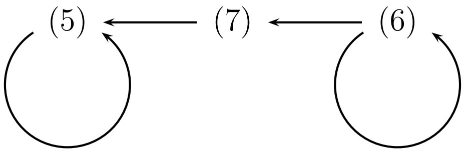
$$\begin{array}{ll}
\mathit{minus}(x, 0) \to x & (1)\\
\mathit{minus}(s(x), s(y)) \to \mathit{minus}(x, y) & (2)\\
\mathit{quot}(0, s(y)) \to 0 & (3)\\
\mathit{quot}(s(x), s(y)) \to s(\mathit{quot}(\mathit{minus}(x, y), s(y))) & (4)
\end{array}$$

(R is not contained in any simplification ordering, since the left-hand side of rule (4) is embedded in the right-hand side after instantiating y by $s(x)$.)

R has three dependency pairs:

$$\begin{array}{ll}
\mathit{minus}^\sharp(s(x), s(y)) \to \mathit{minus}^\sharp(x, y) & (5)\\
\mathit{quot}^\sharp(s(x), s(y)) \to \mathit{quot}^\sharp(\mathit{minus}(x, y), s(y)) & (6)\\
\mathit{quot}^\sharp(s(x), s(y)) \to \mathit{minus}^\sharp(x, y) & (7)
\end{array}$$

The dependency graph of R is



There are exactly two SCCs (and also two cycles). The cycle at (5) can be handled using the subterm criterion with $\pi(minus^\sharp) = 1$. For the cycle at (6) we can use an argument filtering π that maps *minus* to 1 and leaves all other function symbols unchanged (that is, $\pi(g) = [1, \dots, \text{arity}(g)]$ for every g different from *minus*.) After applying the argument filtering, we compare left and right-hand sides using an LPO with precedence $quot > s$ (the precedence of other symbols is irrelevant). We obtain $l \succ r$ for (6) and $l \succsim r$ for (1), (2), (3), (4), so the previous theorem can be applied.

DP Processors

The methods described so far are particular cases of *DP processors:*

A DP processor

$$\frac{(G, R)}{(G_1, R_1),\ \ \dots,\ \ (G_n, R_n)}$$

takes a graph G and a TRS R as input and produces a set of pairs consisting of a graph and a TRS.

It is sound and complete if there are K-minimal infinite sequences for G and R if and only if there are K-minimal infinite sequences for at least one of the pairs (G_i, R_i).

Examples:

$$\frac{(G, R)}{(SCC_1, R),\ \ \dots,\ \ (SCC_n, R)}$$

where $SCC_1, \dots, SCC_n$ are the strongly connected components of G.

$$\frac{(G, R)}{(G \setminus N, R)}$$

if there is an SCC of G and a simple projection π such that $\pi(l) \trianglerighteq \pi(r)$ for all DPs $l \to r$ in the SCC, and N is the set of DPs of the SCC for which $\pi(l) \rhd \pi(r)$.

(and analogously for reduction pairs)

Innermost Termination

The dependency method can also be used for proving termination of *innermost rewriting:* $s \xrightarrow{\text{i}}_R t$ if $s \to_R t$ at position p and no rule of R can be applied at a position strictly below p. (DP processors for innermost termination are more powerful than for ordinary termination, and for program analysis, innermost termination is usually sufficient.)

6 Implementing Saturation Procedures

Problem:

Refutational completeness is nice in theory, but ...

... it guarantees only that proofs will be found eventually, not that they will be found quickly.

Even though orderings and selection functions reduce the number of possible inferences, the search space problem is enormous.

First-order provers "look for a needle in a haystack": It may be necessary to make some millions of inferences to find a proof that is only a few dozens of steps long.

Coping with Large Sets of Formulas

Consequently:

- We must deal with large sets of formulas.
- We must use efficient techniques to find formulas that can be used as partners in an inference.
- We must simplify/eliminate as many formulas as possible.
- We must use efficient techniques to check whether a formula can be simplified/eliminated.

Note:

Often there are several competing implementation techniques.

Design decisions are not independent of each other.

Design decisions are not independent of the particular class of problems we want to solve. (FOL without equality/FOL with equality/unit equations, size of the signature, special algebraic properties like AC, etc.)