Department of Computer Science
Technical Report

Proving Termination by Bounded Increase

Jürgen Giesl, René Thiemann, Stephan Swiderski, Peter Schneider-Kamp

ISSN 0935–3232 · Aachener Informatik Berichte · AIB-2007-03

RWTH Aachen · Department of Computer Science · May 2007 (revised version)

The publications of the Department of Computer Science of *RWTH Aachen University* are in general accessible through the World Wide Web.

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

Proving Termination by Bounded Increase*

Jürgen Giesl, René Thiemann, Stephan Swiderski, and Peter Schneider-Kamp

LuFG Informatik 2, RWTH Aachen, Germany,
{giesl,thiemann,swiderski,psk}@informatik.rwth-aachen.de

Abstract. Most methods and tools for termination analysis of term rewrite systems (TRSs) essentially try to find arguments of functions that *decrease* in recursive calls. However, they fail if the reason for termination is that an argument is *increased* in recursive calls repeatedly until it reaches a bound. In this paper, we solve that problem and present a method to prove innermost termination of TRSs with bounded increase automatically.

1 Introduction

In programming, one often writes algorithms that terminate because a value is increased until it reaches a bound. Hence, to apply termination techniques of TRSs in practice, they must be able to deal with those algorithms successfully. But unfortunately, all existing methods and tools for automated termination analysis of TRSs fail on such examples. Therefore, proving termination of TRSs with bounded increase was identified as one of the most urgent and challenging problems at the annual *International Competition of Termination Tools* 2006 [16].

Example 1. As an example consider the following TRS for subtraction. TRSs of this form often result from the transformation of conditional TRSs or from functional, logic, or imperative programs.

$$
\begin{array}{rlrrlr}
\mathsf{minus}(x,y) & \to \mathsf{cond}(\mathsf{gt}(x,y),x,y) & (1) & \mathsf{gt}(\mathsf{0},v) & \to \mathsf{false} & (4)\\
\mathsf{cond}(\mathsf{false},x,y) & \to \mathsf{0} & (2) & \mathsf{gt}(\mathsf{s}(u),\mathsf{0}) & \to \mathsf{true} & (5)\\
\mathsf{cond}(\mathsf{true},x,y) & \to \mathsf{s}(\mathsf{minus}(x,\mathsf{s}(y))) & (3) & \mathsf{gt}(\mathsf{s}(u),\mathsf{s}(v)) & \to \mathsf{gt}(u,v) & (6)
\end{array}
$$

To handle TRSs like Ex. 1, we propose to use polynomial interpretations [14]. But instead of classical polynomial interpretations on natural numbers, we use interpretations on *integers*. Such interpretations can measure the difference between the first and second argument of minus. Indeed, minus is terminating since this difference decreases in each recursive call. However, using integer polynomial interpretations is unsound in the existing termination techniques for TRSs.

This is also true for the *dependency pair (DP) method* [1], which is a powerful method for automated termination analysis of TRSs that is implemented in

* Supported by the Deutsche Forschungsgemeinschaft DFG under grant GI 274/5-1 and by the DFG Research Training Group 1298 (*AlgoSyn*).

virtually all current automated termination tools. This method relies on the use of *reduction pairs* $(\succsim, \succ)$ to compare terms. Here, $\succsim$ is a stable quasi-order and $\succ$ is a stable order, where $\succsim$ and $\succ$ are compatible (i.e., $\succ \circ \succsim \subseteq \succ$ or $\succsim \circ \succ \subseteq \succ$). Moreover, $\succsim$ and $\succ$ have to satisfy the following properties:

(a) $\succsim$ is monotonic (b) $\succ$ is well founded

After recapitulating the DP method in Sect. 2, in Sect. 3 we extend it to *general* reduction pairs (without requirements (a) and (b)). Then one can also use reduction pairs based on integer polynomial interpretations, which violate the requirements (a) and (b).

In Sect. 4 we extend the DP method further to exploit implicit *conditions*. This is needed to prove that an increase is bounded. For instance, the recursive call of minus in Ex. 1 only takes place under the *condition* $\mathsf{gt}(x, y) = \mathsf{true}$.[1] With our extensions, termination provers based on DPs can handle most algorithms with bounded increase that typically occur in practice. In Sect. 5, we discuss the implementation of our method in our termination tool AProVE [10]. To demonstrate the power of our approach, the appendix contains a collection of typical TRSs with bounded increase where all existing techniques and tools failed up to now, but where the implementation of our new technique succeeds.

2 Dependency Pairs

We assume familiarity with term rewriting [2] and briefly recapitulate the DP method. See [1, 8, 11–13] for further motivations and extensions.

Definition 2 (Dependency Pairs). *For a TRS $\mathcal{R}$, the* defined *symbols $\mathcal{D}$ are the root symbols of left-hand sides of rules. All other function symbols are called* constructors. *For every defined symbol $f \in \mathcal{D}$, we introduce a fresh* tuple symbol *$f^\sharp$ with the same arity. To ease readability, we often write F instead of $f^\sharp$, etc. If $t = f(t_1, \ldots, t_n)$ with $f \in \mathcal{D}$, we write $t^\sharp$ for $f^\sharp(t_1, \ldots, t_n)$. If $\ell \to r \in \mathcal{R}$ and t is a subterm of r with defined root symbol, then the rule $\ell^\sharp \to t^\sharp$ is a* dependency pair *of $\mathcal{R}$. The set of all dependency pairs of $\mathcal{R}$ is denoted $DP(\mathcal{R})$.*

Ex. 1 has the following DPs, where MINUS is the tuple symbol for minus, etc.

$$\mathsf{MINUS}(x, y) \to \mathsf{COND}(\mathsf{gt}(x, y), x, y) \quad (7) \qquad \mathsf{COND}(\mathsf{true}, x, y) \to \mathsf{MINUS}(x, \mathsf{s}(y)) \quad (9)$$
$$\mathsf{MINUS}(x, y) \to \mathsf{GT}(x, y) \quad (8) \qquad \mathsf{GT}(\mathsf{s}(u), \mathsf{s}(v)) \to \mathsf{GT}(u, v) \quad (10)$$

[1] Proving termination of TRSs like Ex. 1 is far more difficult than proving termination of programs in a language where one uses a *predefined* function gt. (For such languages, there already exist termination techniques that can handle certain forms of bounded increase [5, 15].) However, if a function like gt is not predefined but written by the "user", then the termination technique cannot presuppose any knowledge about gt's semantics. In contrast, the termination technique has to deduce any needed informations about gt from the user-defined gt-rules.

In this paper, we only focus on *innermost* termination, i.e., we only regard the innermost rewrite relation $\xrightarrow{\mathrm{i}}$. The reason is that proving innermost termination is considerably easier than proving full termination and there are large classes of TRSs where innermost termination is already sufficient for termination. In particular, this holds for non-overlapping TRSs like Ex. 1.

The main result of the DP method for innermost termination states that a TRS $\mathcal{R}$ is innermost terminating iff there is no infinite minimal innermost $(DP(\mathcal{R}), \mathcal{R})$-*chain*. For any TRSs $\mathcal{P}$ and $\mathcal{R}$, a minimal innermost $(\mathcal{P}, \mathcal{R})$-*chain* is a sequence of (variable renamed) pairs $s_1 \to t_1, s_2 \to t_2, \ldots$ from $\mathcal{P}$ such that there is a substitution σ (with possibly infinite domain) where $t_i\sigma \xrightarrow{\mathrm{i}}{}^{*}_{\mathcal{R}} s_{i+1}\sigma$, where all $s_i\sigma$ are in normal form, and where all $t_i\sigma$ are innermost terminating w.r.t. $\mathcal{R}$.

Termination techniques are now called *DP processors* and they operate on sets of dependency pairs (which are called *DP problems*).[2] Formally, a DP processor *Proc* takes a DP problem as input and returns a set of new DP problems which then have to be solved instead. A processor *Proc* is *sound* if for all DP problems $\mathcal{P}$ with infinite minimal innermost $(\mathcal{P}, \mathcal{R})$-chain there is also a $\mathcal{P}' \in \mathit{Proc}(\mathcal{P})$ with an infinite minimal innermost $(\mathcal{P}', \mathcal{R})$-chain. Soundness of a DP processor is required to prove innermost termination and in particular, to conclude that there is no infinite minimal innermost $(\mathcal{P}, \mathcal{R})$-chain if $\mathit{Proc}(\mathcal{P}) = \{\varnothing\}$.

So innermost termination proofs in the DP framework start with the initial DP problem $DP(\mathcal{R})$. Then the DP problem is simplified repeatedly by sound DP processors. If all resulting DP problems have been simplified to $\varnothing$, then innermost termination is proved. In Thm. 3, we recapitulate one of the most important processors of the framework, the so-called *reduction pair processor*.

For a DP problem $\mathcal{P}$, the reduction pair processor generates inequality constraints which should be satisfied by a reduction pair $(\succsim, \succ)$. The constraints require that all DPs in $\mathcal{P}$ are strictly or weakly decreasing and all *usable rules* $\mathcal{U}(\mathcal{P})$ are weakly decreasing. Then one can delete all strictly decreasing DPs.

The *usable rules* include all rules that can be used to reduce the terms in right-hand sides of $\mathcal{P}$ when their variables are instantiated with normal forms. More precisely, for a term containing a defined symbol f, all f-rules are usable. Moreover, if the f-rules are usable and g occurs in the right-hand side of an f-rule, then the g-rules are usable as well. In Thm. 3, note that both TRSs and relations can be seen as sets of pairs of terms. Thus, "$\mathcal{P} \setminus \succ$" denotes $\{s \to t \in \mathcal{P} \mid s \not\succ t\}$.

Theorem 3 (Reduction Pair Processor and Usable Rules). *Let $(\succsim, \succ)$ be a reduction pair. Then the following DP processor Proc is sound.*

$$\mathit{Proc}(\mathcal{P}) = \begin{cases} \{\, \mathcal{P} \setminus \succ \,\} & \text{if } \mathcal{P} \subseteq {\succ} \cup {\succsim} \text{ and } \mathcal{U}(\mathcal{P}) \subseteq {\succsim} \\ \{\, \mathcal{P} \,\} & \text{otherwise} \end{cases}$$

[2] To ease readability we use a simpler definition of *DP problems* than [8], since this simple definition suffices for the presentation of the new results of this paper.

For any function symbol f, let $Rls(f) = \{\ell \to r \in \mathcal{R} \mid \mathrm{root}(\ell) = f\}$. For any term t, the usable rules $\mathcal{U}(t)$ *are the smallest set such that*

- $\mathcal{U}(x) = \varnothing$ *for every variable x and*
- $\mathcal{U}(f(t_1, \ldots, t_n)) = Rls(f) \ \cup \ \bigcup_{\ell \to r \in Rls(f)} \mathcal{U}(r) \ \cup \ \bigcup_{i=1}^{n} \mathcal{U}(t_i)$

For a set of dependency pairs $\mathcal{P}$, its usable rules *are* $\mathcal{U}(\mathcal{P}) = \bigcup_{s \to t \in \mathcal{P}} \mathcal{U}(t)$.

For the TRS of Ex. 1, according to Thm. 3 we search for a reduction pair with $s \mathrel{_{(}\succsim_{)}} t$ for all dependency pairs $s \to t \in DP(\mathcal{R}) = \{(7), \ldots, (10)\}$ and with $\ell \succsim r$ for all usable rules $\ell \to r \in \mathcal{U}(DP(\mathcal{R})) = \{(4), (5), (6)\}$.

A popular method to search for suitable relations $\succsim$ and $\succ$ automatically is the use of *polynomial interpretations* [14]. A polynomial interpretation $\mathcal{P}ol$ maps every n-ary function symbol f to a polynomial $f_{\mathcal{P}ol}$ over n variables $x_1, \ldots, x_n$. Traditionally, one uses polynomials with coefficients from $\mathbb{N} = \{0, 1, 2, \ldots\}$. This mapping is then extended to terms by defining $[x]_{\mathcal{P}ol} = x$ for all variables x and by defining $[f(t_1, \ldots, t_n)]_{\mathcal{P}ol} = f_{\mathcal{P}ol}([t_1]_{\mathcal{P}ol}, \ldots, [t_n]_{\mathcal{P}ol})$. If $\mathcal{P}ol$ is clear from the context, we also write $[t]$ instead of $[t]_{\mathcal{P}ol}$. Now one defines $s \succ_{\mathcal{P}ol} t$ (resp. $s \succsim_{\mathcal{P}ol} t$) iff $[s] > [t]$ (resp. $[s] \geq [t]$) holds for all instantiations of the variables with natural numbers. It is easy to see that $(\succsim_{\mathcal{P}ol}, \succ_{\mathcal{P}ol})$ is a reduction pair.

As an example, consider the polynomial interpretation $\mathcal{P}ol_1$ with $\mathsf{GT}_{\mathcal{P}ol_1} = x_1$, $\mathsf{MINUS}_{\mathcal{P}ol_1} = x_1 + 1$, $\mathsf{COND}_{\mathcal{P}ol_1} = x_2 + 1$, $\mathsf{s}_{\mathcal{P}ol_1} = x_1 + 1$, and $f_{\mathcal{P}ol_1} = 0$ for all other function symbols f. Then the DPs (8) and (10) are strictly decreasing. The reason for $\mathsf{GT}(\mathsf{s}(x), \mathsf{s}(y)) \succ_{\mathcal{P}ol_1} \mathsf{GT}(x, y)$ is that $[\mathsf{GT}(\mathsf{s}(x), \mathsf{s}(y))] = x + 1$ is greater than $[\mathsf{GT}(x, y)] = x$ for all natural numbers x. Moreover, all other DPs and the usable rules are weakly decreasing w.r.t. $\succsim_{\mathcal{P}ol_1}$. Thus, the DPs (8) and (10) can be removed and the reduction pair processor transforms the initial DP problem $DP(\mathcal{R})$ into $\{(7), (9)\}$. We refer to [4, 7] for efficient algorithms to generate suitable polynomial interpretations automatically. However, it is impossible to transform the problem further into the empty DP problem $\varnothing$. More precisely, there is no reduction pair based on polynomial interpretations (or on any other classical order amenable to automation)[3] where one of the DPs (7) and (9) is strictly decreasing and the other one and the usable rules are weakly decreasing. Indeed, up to now all implementations of the DP method failed on Ex. 1.

3 General Reduction Pairs

Our aim is to handle *integer* polynomial interpretations. More precisely, we want to use polynomial interpretations where all function symbols except tuple

[3] There is no such quasi-simplification order and also no polynomial order. The reason for the latter is the following: We must have $\mathsf{MINUS}(\mathsf{s}(\mathsf{0}), \mathsf{0}) \succ \mathsf{MINUS}(\mathsf{s}(\mathsf{0}), \mathsf{s}(\mathsf{0}))$. If we had $\mathsf{0} \precsim \mathsf{s}(\mathsf{0})$, then by weak monotonicity we would obtain $\mathsf{MINUS}(\mathsf{s}(\mathsf{0}), \mathsf{0}) \precsim \mathsf{MINUS}(\mathsf{s}(\mathsf{0}), \mathsf{s}(\mathsf{0}))$ which is a contradiction. But since polynomial orders are total on ground terms, this implies $\mathsf{0} \succ \mathsf{s}(\mathsf{0})$. Hence, weak monotonicity implies $\mathsf{0} \succsim \mathsf{s}(\mathsf{0}) \succsim \mathsf{s}(\mathsf{s}(\mathsf{0})) \succsim \ldots$. None of the terms in this chain can be $\approx$-equal, since for $i > j$ we have $\mathsf{MINUS}(\mathsf{s}^i(\mathsf{0}), \mathsf{s}^j(\mathsf{0})) \succ \mathsf{MINUS}(\mathsf{s}^i(\mathsf{0}), \mathsf{s}^i(\mathsf{0}))$. But since on ground terms we have $\succsim \, = \, \succ \cup \approx$ for polynomial orders, this implies the contradiction $\mathsf{0} \succ \mathsf{s}(\mathsf{0}) \succ \mathsf{s}(\mathsf{s}(\mathsf{0})) \succ \ldots$.

symbols are still mapped to polynomials with natural coefficients, but where tuple symbols may be mapped to polynomials with arbitrary integer coefficients. For such integer polynomial interpretations, we still define $s \succ_{\mathcal{P}ol} t$ (resp. $s \succsim_{\mathcal{P}ol} t$) iff $[s] > [t]$ (resp. $[s] \geq [t]$) holds for all instantiations of the variables with *natural* (not with *integer*) numbers. If $\mathcal{F}$ is the original signature without tuple symbols, then the relations $\succ_{\mathcal{P}ol}$ and $\succsim_{\mathcal{P}ol}$ are $\mathcal{F}$-stable, i.e., $s \underset{(\sim)}{\succ}_{\mathcal{P}ol} t$ implies $s\sigma \underset{(\sim)}{\succ}_{\mathcal{P}ol} t\sigma$ for all substitutions σ with terms over $\mathcal{F}$. It is easy to show that $\mathcal{F}$-stability is sufficient for the reduction pairs used in the reduction pair processor.

To solve the remaining DP problem $\{(7), (9)\}$, we want to use the integer polynomial interpretation $\mathcal{P}ol_2$ where $\mathsf{MINUS}_{\mathcal{P}ol_2} = x_1 - x_2$, $\mathsf{COND}_{\mathcal{P}ol_2} = x_2 - x_3$, $\mathsf{s}_{\mathcal{P}ol_2} = x_1 + 1$, and $f_{\mathcal{P}ol_2} = 0$ for all other symbols f. Then DP (9) would be strictly decreasing and could be removed. The resulting DP problem $\{(7)\}$ is easy to solve by $\mathcal{P}ol_3$ with $\mathsf{MINUS}_{\mathcal{P}ol_3} = 1$ and $f_{\mathcal{P}ol_3} = 0$ for all other symbols f.

But such integer interpretations may not be used, since $(\succsim_{\mathcal{P}ol_2}, \succ_{\mathcal{P}ol_2})$ is no reduction pair: $\succsim_{\mathcal{P}ol_2}$ is not monotonic (e.g., $\mathsf{s}(\mathsf{0}) \succsim_{\mathcal{P}ol_2} \mathsf{0}$, but $\mathsf{MINUS}(\mathsf{s}(\mathsf{0}), \mathsf{s}(\mathsf{0})) \not\succsim_{\mathcal{P}ol_2} \mathsf{MINUS}(\mathsf{s}(\mathsf{0}), \mathsf{0})$). Moreover, $\succ_{\mathcal{P}ol_2}$ is not well founded (e.g., $\mathsf{MINUS}(\mathsf{0}, \mathsf{0}) \succ_{\mathcal{P}ol_2} \mathsf{MINUS}(\mathsf{0}, \mathsf{s}(\mathsf{0})) \succ_{\mathcal{P}ol_2} \mathsf{MINUS}(\mathsf{0}, \mathsf{s}(\mathsf{s}(\mathsf{0}))) \succ_{\mathcal{P}ol_2} \ldots$). So integer interpretations violate both requirements (a) and (b) for reduction pairs, cf. Sect. 1.

Indeed, using such polynomial interpretations in Thm. 3 is unsound. As $\succ_{\mathcal{P}ol_2}$ is not well founded (i.e., as it violates requirement (b)), $\mathcal{P}ol_2$ could be used for a wrong innermost termination proof of the TRS $\{\mathsf{minus}(x, y) \to \mathsf{minus}(x, \mathsf{s}(y))\}$. But even if requirement (b) were not violated, a violation of requirement (a) would still render Thm. 3 unsound. We demonstrate this in Ex. 4.

Example 4. Consider the following TRS which is not innermost terminating. Here, $\mathsf{round}(x) = x$ *if* x *is even and* $\mathsf{round}(x) = \mathsf{s}(x)$ *if* x *is odd.*

$$\mathsf{minus}(\mathsf{s}(x), x) \to \mathsf{minus}(\mathsf{s}(x), \mathsf{round}(x)) \quad (11)$$

$$\mathsf{round}(\mathsf{0}) \to \mathsf{0} \quad (12)$$

$$\mathsf{round}(\mathsf{s}(\mathsf{0})) \to \mathsf{s}(\mathsf{s}(\mathsf{0})) \quad (13)$$

$$\mathsf{round}(\mathsf{s}(\mathsf{s}(x))) \to \mathsf{s}(\mathsf{s}(\mathsf{round}(x))) \quad (14)$$

We use a modification $\mathcal{P}ol_2'$ *of* $\mathcal{P}ol_2$*, where* $\mathsf{MINUS}_{\mathcal{P}ol_2'} = (x_1 - x_2)^2$, $\mathsf{round}_{\mathcal{P}ol_2'} = x_1 + 1$, *and* $\mathsf{ROUND}_{\mathcal{P}ol_2'} = 0$. *Now requirement (b) is satisfied. The* MINUS*-DPs are strictly decreasing (i.e.,* $\mathsf{MINUS}(\mathsf{s}(x), x) \succ_{\mathcal{P}ol_2'} \mathsf{MINUS}(\mathsf{s}(x), \mathsf{round}(x))$ *and* $\mathsf{MINUS}(\mathsf{s}(x), x) \succ_{\mathcal{P}ol_2'} \mathsf{ROUND}(x)$*) and the* ROUND*-DP and the usable rules are weakly decreasing. Thus, if we were allowed to use* $\mathcal{P}ol_2'$ *in Thm. 3, then we could remove the* MINUS*-DPs. The remaining DP problem is easily solved and thus, we would falsely prove innermost termination of this TRS.*

Ex. 4 shows the reason for the unsoundness when dropping requirement (a). Thm. 3 requires $\ell \succsim r$ for all usable rules $\ell \to r$. This is meant to ensure that all reductions with usable rules will weakly decrease the reduced term (w.r.t. $\succsim$). However, this only holds if the quasi-order $\succsim$ is monotonic. In Ex. 4, we have $\mathsf{round}(x) \succsim_{\mathcal{P}ol_2'} x$, but $\mathsf{MINUS}(\mathsf{s}(x), \mathsf{round}(x)) \not\succsim_{\mathcal{P}ol_2'} \mathsf{MINUS}(\mathsf{s}(x), x)$.

Therefore, one should take into account on which positions the used quasi-order $\succsim$ is monotonically *increasing* and on which positions it is monotonically *decreasing*. If a defined function symbol f occurs at a monotonically *increasing* position in the right-hand side of a dependency pair, then one should require $\ell \succsim r$ for all f-rules. If f occurs at a monotonically *decreasing* position, then one should require $r \succsim \ell$. Finally, if f occurs at a position which is neither monotonically increasing nor decreasing, one should require $\ell \approx r$. Here, $\approx$ is the equivalence relation associated with $\succsim$, i.e., $\approx \, = \, \succsim \cap \precsim$.

So we modify our definition of usable rules.[4] When computing $\mathcal{U}(f(t_1, ..., t_n))$, for any $i \in \{1, ..., n\}$ we first check how the quasi-order $\succsim$ treats f's i-th argument. We say that f is $\succsim$*-dependent* on i iff there exist terms $t_1, ..., t_n, t'_i$ where $f(t_1, \dots, t_i, ..., t_n) \not\approx f(t_1, ..., t'_i, \dots, t_n)$. Moreover, f is $\succsim$*-monotonically increasing* (resp. *decreasing*) on i iff $t_i \succsim t'_i$ implies $f(t_1, ..., t_i, ..., t_n) \succsim f(t_1, ..., t'_i, ..., t_n)$ (resp. $f(t_1, ..., t_i, ..., t_n) \precsim f(t_1, ..., t'_i, ..., t_n)$) for all terms $t_1, ..., t_n$ and t'_i.

Now if f is not $\succsim$-dependent on i, then $\mathcal{U}(t_i)$ does not have to be included in $\mathcal{U}(f(t_1, \dots, t_n))$ at all. (This idea was already used in recent refined definitions of the "usable rules", cf. [11].) Otherwise, we include the usable rules $\mathcal{U}(t_i)$ if f is $\succsim$-monotonically increasing on i. If it is $\succsim$-monotonically decreasing, we include the reversed rules $\mathcal{U}^{-1}(t_i)$ instead. Finally, if f is $\succsim$-dependent on i, but neither $\succsim$-monotonically increasing nor decreasing, then we include the usable rules of t_i in both directions, i.e., we include $\mathcal{U}^2(t_i)$ which is defined to be $\mathcal{U}(t_i) \cup \mathcal{U}^{-1}(t_i)$.

Definition 5 (General Usable Rules). *For any function symbol f and any $i \in \{1, \dots, \mathrm{arity}(f)\}$, we define*

$$\mathrm{ord}(f,i) = \begin{cases} 0, & \textit{if } f \textit{ is not } \succsim\textit{-dependent on } i \\ 1, & \textit{otherwise, if } f \textit{ is } \succsim\textit{-monotonically increasing on } i \\ -1, & \textit{otherwise, if } f \textit{ is } \succsim\textit{-monotonically decreasing on } i \\ 2, & \textit{otherwise} \end{cases}$$

For any TRS U, we define $U^0 = \varnothing$, $U^1 = U$, $U^{-1} = \{r \to \ell \mid \ell \to r \in U\}$, and $U^2 = U \cup U^{-1}$. For any term t, we define $\mathcal{U}(t)$ as the smallest set such that[5]

- $\mathcal{U}(x) = \varnothing$ *for every variable x and*
- $\mathcal{U}(f(t_1, \dots, t_n)) = Rls(f) \;\cup\; \bigcup_{\ell \to r \in Rls(f)} \mathcal{U}(r) \;\cup\; \bigcup_{i=1}^{n} \mathcal{U}^{\mathrm{ord}(f,i)}(t_i)$

For a set of dependency pairs $\mathcal{P}$, we again define $\mathcal{U}(\mathcal{P}) = \bigcup_{s \to t \in \mathcal{P}} \mathcal{U}(t)$.

So in Ex. 4, if $\mathsf{MINUS}_{\mathcal{P}ol'_2} = (x_1 - x_2)^2$ then MINUS is $\succsim_{\mathcal{P}ol'_2}$-dependent on 2, but neither $\succsim_{\mathcal{P}ol'_2}$-monotonically increasing nor decreasing. Hence, the usable rules include $\ell \to r$ and $r \to \ell$ for all round-rules $\ell \to r \in \{(12), (13), (14)\}$. Thus, we cannot falsely prove innermost termination with $\mathcal{P}ol'_2$ anymore. Indeed, with the modified definition of usable rules above, Thm. 3 can also be used for reduction pairs where $\succsim$ is not monotonic, i.e., where requirement (a) is violated.

We now also show how to omit the requirement (b) that the order $\succ$ in a reduction pair has to be well founded. Instead, we replace well-foundedness by the weaker requirement of *non-infinitesimality.*

---

[4] Now $\mathcal{U}(t)$ is no longer a subset of $\mathcal{R}$. We nevertheless refer to $\mathcal{U}(t)$ as "usable" rules in order to keep the similarity to Thm. 3.

[5] To ease readability, for $k \in \{-1, 0, 1, 2\}$ we write "$\mathcal{U}^k(t)$" instead of "$(\mathcal{U}(t))^k$". Note that Def. 5 can also be combined with recent refined definitions of "usable rules" [9].

Definition 6 (Non-Infinitesimal). *A relation* $\succ$ *is* non-infinitesimal *if there do not exist any* $t, s_0, s_1, \ldots$ *with* $s_i \succ s_{i+1}$ *and* $s_i \succ t$ *for all* $i \in \mathbb{N}$.

Any well-founded relation is non-infinitesimal. Thm. 7 shows that integer polynomial orders (which are not well founded) are non-infinitesimal as well.

Theorem 7 (Non-Infinitesimality of Integer Polynomial Orders). *Let* $\mathcal{P}ol$ *be an integer polynomial interpretation. Then* $\succ_{\mathcal{P}ol}$ *is non-infinitesimal.*

Proof. Suppose that there exist terms $t, s_0, s_1, \ldots$ with $s_i \succ_{\mathcal{P}ol} s_{i+1}$ and $s_i \succ_{\mathcal{P}ol} t$. Thus, we have $[s_i] > [s_{i+1}]$ and $[s_i] > [t]$ for all i. This means that these inequalities hold for all instantiations of the variables with natural numbers. Hence, if we choose an arbitrary instantiation of the variables (e.g., if we instantiate them all with the number 0), then each polynomial $[s_i]$ results in a number n_i and the polynomial $[t]$ results in a number m where $n_i > n_{i+1}$ and $n_i > m$ for all i. However, there do not exist any such integers n_i and m. ☐

Note that non-infinitesimality of $\succ_{\mathcal{P}ol}$ does not hold for polynomial interpretations on *rational* numbers. To see this, let $\mathsf{a}_{\mathcal{P}ol} = 1$, $\mathsf{b}_{\mathcal{P}ol} = 0$, and $\mathsf{f}_{\mathcal{P}ol} = \frac{x_1}{2}$. For $s_i = \mathsf{f}^i(\mathsf{a})$ and $t = \mathsf{b}$, we get the infinite sequence $\mathsf{a} \succ_{\mathcal{P}ol} \mathsf{f}(\mathsf{a}) \succ_{\mathcal{P}ol} \mathsf{f}(\mathsf{f}(\mathsf{a})) \succ_{\mathcal{P}ol} \ldots$ (i.e., $s_i \succ_{\mathcal{P}ol} s_{i+1}$ for all i) and $\mathsf{f}^i(\mathsf{a}) \succ_{\mathcal{P}ol} \mathsf{b}$ (i.e., $s_i \succ_{\mathcal{P}ol} t$) for all i.

We now extend the reduction pair processor from Thm. 3 to *general* reduction pairs. A *general* reduction pair $(\succsim, \succ)$ consists of an $\mathcal{F}$-stable quasi-order $\succsim$ and a compatible $\mathcal{F}$-stable non-infinitesimal order $\succ$, where $\mathcal{F}$ is the original signature of the TRS, i.e., without tuple symbols. Moreover, the equivalence relation $\approx$ associated with $\succsim$ must be monotonic (i.e., $s \approx t$ implies $u[s]_\pi \approx u[t]_\pi$ for any position π of any term u). But we do not require monotonicity of $\succsim$ or well-foundedness of $\succ$, i.e., both requirements (a) and (b) are dropped. So for any integer polynomial interpretation $\mathcal{P}ol$, $(\succsim_{\mathcal{P}ol}, \succ_{\mathcal{P}ol})$ is a general reduction pair.

In contrast to the reduction pair processor from Thm. 3, the new processor transforms a DP problem into *two* new problems. As before, the first problem results from removing all strictly decreasing dependency pairs. The second DP problem results from removing all DPs $s \to t$ from $\mathcal{P}$ that are *bounded from below*, i.e., DPs which satisfy the inequality $s \succsim \mathsf{c}$ for a fresh constant c.

Theorem 8 (General Reduction Pair Processor). *Let* $(\succsim, \succ)$ *be a general reduction pair. Let* c *be a fresh constant not occurring in the signature and let* $\mathcal{P}_{bound} = \{s \to t \in \mathcal{P} \mid s \succsim \mathsf{c}\}$. *Then the following DP processor Proc is sound. Here,* $\mathcal{U}(\mathcal{P})$ *is defined as in Def. 5.*

$$Proc(\mathcal{P}) = \begin{cases} \{\, \mathcal{P} \setminus \succ,\ \mathcal{P} \setminus \mathcal{P}_{bound} \,\} & \text{if } \mathcal{P} \subseteq \succ \cup \succsim \text{ and } \mathcal{U}(\mathcal{P}) \subseteq \succsim \\ \{\, \mathcal{P} \,\} & \text{otherwise} \end{cases}$$

Proof. Let $\mathcal{P} \subseteq \succ \cup \succsim$ and $\mathcal{U}(\mathcal{P}) \subseteq \succsim$. Suppose that there is an infinite minimal innermost $(\mathcal{P}, \mathcal{R})$-chain $s_1 \to t_1, s_2 \to t_2, \ldots$ where $t_i\sigma \xrightarrow{\mathrm{i}}{}^*_{\mathcal{R}} s_{i+1}\sigma$, $s_i\sigma$ is in normal form, and $t_i\sigma$ is innermost terminating for all i. We will show that $t_i\sigma \succsim s_{i+1}\sigma$ holds for all i. Then by non-infinitesimality, there cannot be both infinitely many i with $s_i \succ t_i$ and also infinitely many j with $s_j \succsim \mathsf{c}$. Thus, the chain has an infinite tail from $\mathcal{P} \setminus \succ$ or from $\mathcal{P} \setminus \mathcal{P}_{bound}$. It suffices to prove the following claim, where a *normal* substitution instantiates all variables by normal forms.

$$\text{If } t\sigma \xrightarrow{\mathsf{i}}^*_{\mathcal{R}} v \text{ for a normal substitution } \sigma \text{ and } \mathcal{U}(t) \subseteq \succsim, \text{ then } t\sigma \succsim v. \tag{15}$$

To prove (15) we show the following auxiliary claim for all $k \in \{-1, 0, 1, 2\}$ which implies (15) as can be shown by a straightforward induction on the length of the reduction.[6]

$$\begin{array}{l}\text{If } t\sigma \xrightarrow{\mathsf{i}}_{\mathcal{R}} v \text{ for a normal substitution } \sigma \text{ and } \mathcal{U}^k(t) \subseteq \succsim, \text{ then } \{t\sigma \to v\}^k \subseteq \succsim. \\ \text{Moreover, } v = u\sigma' \text{ and } \mathcal{U}^k(u) \subseteq \mathcal{U}^k(t) \text{ for a term } u \text{ and a normal substitution } \sigma'.\end{array} \tag{16}$$

We prove (16) by induction on the position of the redex. This position must be in t because σ is normal. So t has the form $f(t_1, \ldots, t_n)$.

If the reduction is on the root position, then we have $t\sigma = \ell\sigma' \xrightarrow{\mathsf{i}}_{\mathcal{R}} r\sigma' = v$ where $\ell \to r \in \mathit{Rls}(f)$. As $\mathit{Rls}^k(f) \subseteq \mathcal{U}^k(t) \subseteq \succsim$, we have $\{t\sigma \to v\}^k \subseteq \succsim$ by stability of $\succsim$. Note that σ' is a normal substitution due to the innermost strategy and by choosing $u = r$ we obtain $v = u\sigma'$ and $\mathcal{U}^k(u) = \mathcal{U}^k(r) \subseteq \mathcal{U}^k(t)$.

Now we regard the case where the reduction is not on the root position. Then $t\sigma = f(t_1\sigma, \ldots, t_i\sigma, \ldots, t_n\sigma) \xrightarrow{\mathsf{i}}_{\mathcal{R}} f(t_1\sigma, \ldots, v_i, \ldots, t_n\sigma) = v$ where $t_i\sigma \xrightarrow{\mathsf{i}}_{\mathcal{R}} v_i$.

We first prove $\{t\sigma \to v\}^k \subseteq \succsim$. The claim is trivial for $k = 0$, thus we now regard $k \neq 0$. If f is not $\succsim$-dependent on i, then we have $t\sigma \approx v$ which implies $\{t\sigma \to v\}^k \subseteq \succsim$. Otherwise, if f is $\succsim$-monotonically increasing on i, then $\mathcal{U}^k(t_i) \subseteq \mathcal{U}^k(t)$. By the induction hypothesis we have $\{t_i\sigma \to v_i\}^k \subseteq \succsim$, which implies $\{t\sigma \to v\}^k \subseteq \succsim$. Otherwise, if f is $\succsim$-monotonically decreasing on i and $k \in \{-1, 1\}$, then $\mathcal{U}^{-k}(t_i) \subseteq \mathcal{U}^k(t)$ By the induction hypothesis we have $\{v_i \to t_i\sigma\}^k \subseteq \succsim$, which implies $\{t\sigma \to v\}^k \subseteq \succsim$. Otherwise, we obtain $\mathcal{U}^2(t_i) \subseteq \mathcal{U}^k(t)$. By the induction hypothesis we have $t_i\sigma \approx v_i$, which implies $\{t\sigma \to v\}^k \subseteq \succsim$ due to the monotonicity of $\approx$.

Now we show the existence of a term u and a normal substitution σ' with $v = u\sigma'$ and $\mathcal{U}^k(u) \subseteq \mathcal{U}^k(t)$. By the induction hypothesis there is some term u_i and some normal substitution σ_i with $v_i = u_i\sigma_i$. Let u'_i result from u_i by replacing its variables x by corresponding fresh variables x'. We define $\sigma'(x') = \sigma_i(x)$ for all these fresh variables and $\sigma'(x) = \sigma(x)$ for all $x \in \mathcal{V}(t)$. Then for $u = f(t_1, \ldots, u'_i, \ldots, t_n)$ we obtain $v = u\sigma'$. We also have $\mathcal{U}^j(u'_i) = \mathcal{U}^j(u_i) \subseteq \mathcal{U}^j(t_i)$ for all $j \in \{-1, 0, 1, 2\}$ by the induction hypothesis.

It remains to show $\mathcal{U}^k(u) \subseteq \mathcal{U}^k(t)$. If f is not $\succsim$-dependent on i, then we even have $\mathcal{U}^k(u) = \mathcal{U}^k(t)$. Otherwise, the only difference between $\mathcal{U}^k(u)$ and $\mathcal{U}^k(t)$ is that $\mathcal{U}^k(t)$ contains $\mathcal{U}^j(t_i)$ and that $\mathcal{U}^k(u)$ contains $\mathcal{U}^j(u'_i)$ instead, for some $j \in \{-1, 0, 1, 2\}$. So by the above observation $\mathcal{U}^j(u'_i) \subseteq \mathcal{U}^j(t_i)$ for all j, we also have $\mathcal{U}^k(u) \subseteq \mathcal{U}^k(t)$. ☐

Example 9. To modify Ex. 4 into an innermost terminating TRS, we replace rule (11) by $\mathsf{minus}(\mathsf{s}(x), x) \to \mathsf{minus}(\mathsf{s}(x), \mathsf{round}(\mathsf{s}(x)))$. *We regard the interpretation* $\mathcal{P}ol''_2$ *with* $\mathsf{MINUS}_{\mathcal{P}ol''_2} = x_1 - x_2$, $\mathsf{s}_{\mathcal{P}ol''_2} = x_1 + 1$, $\mathsf{0}_{\mathcal{P}ol''_2} = 0$, $\mathsf{round}_{\mathcal{P}ol''_2} = x_1$,

[6] This claim corresponds to [11, Lemma 23].

$\mathsf{ROUND}_{\mathcal{P}ol''_2} = 0$, *and* $\mathsf{c}_{\mathcal{P}ol''_2} = 0$. *Then the* MINUS*-DPs are strictly decreasing and the* ROUND*-DP and the usable rules are weakly decreasing. Here, the usable rules are the* reversed round*-rules, since* MINUS *is* $\succsim$*-monotonically decreasing on 2. Moreover, all dependency pairs are bounded from below (i.e.,* $\mathsf{MINUS}(\mathsf{s}(x), x) \succsim_{\mathcal{P}ol''_2} \mathsf{c}$ *and* $\mathsf{ROUND}(\mathsf{s}(\mathsf{s}(x))) \succsim_{\mathcal{P}ol''_2} \mathsf{c}$*). Thus, we can transform the initial DP problem* $\mathcal{P} = DP(\mathcal{R})$ *into* $\mathcal{P} \setminus \mathcal{P}_{bound} = \varnothing$ *and into* $\mathcal{P} \setminus \succ$*, which only contains the* ROUND*-DP. This remaining DP problem is easily solved and thus, we can prove innermost termination of the TRS.*

Since $\mathcal{U}(\mathcal{P})$ now depends on $\succsim$, the constraints that the reduction pair has to satisfy in Thm. 8 depend on the reduction pair itself. Nevertheless, if one uses reduction pairs based on polynomial interpretations, then the search for suitable reduction pairs can still be mechanized efficiently. More precisely, one can reformulate Thm. 8 in a way where one first generates constraints (that are independent of $\succsim$) and searches for a reduction pair satisfying the constraints afterwards. We showed in [11, Sect. 7.1] how to reformulate "f is $\succsim$-dependent on i" accordingly and "f is $\succsim$-monotonically increasing on i" can be reformulated by requiring that the partial derivative of $f_{\mathcal{P}ol}$ w.r.t. x_i is non-negative, cf. [1, Footnote 11].

There have already been previous approaches to extend the DP method to non-monotonic reduction pairs. Hirokawa and Middeldorp [13] allowed interpretations like $\mathsf{MINUS}_{\mathcal{P}ol} = \max(x_1 - x_2, 0)$.[7] However, instead of detecting $\succsim$-monotonically increasing and decreasing positions, they always require $\ell \approx r$ for the usable rules. Therefore, their technique fails on Ex. 9, since their constraints cannot be fulfilled by the interpretations considered in their approach.[8]

Another approach was presented in [1, Thm. 33] and further extended in [6]. Essentially, here one permits non-monotonic quasi-orders $\succsim$ provided that f is $\succsim$-monotonically increasing on a position i whenever there is a subterm $f(t_1, ..., t_i, ..., t_n)$ in a right-hand side of a dependency pair or of a usable rule where t_i contains a defined symbol. Then Thm. 3 is still sound (this also follows from Def. 5 and Thm. 8). However, this approach would not allow us to handle arbitrary non-monotonic reduction pairs and therefore, it also fails on Ex. 9.

4 Conditions for Bounded Increase

With Thm. 8 we still cannot use our desired integer polynomial interpretation $\mathcal{P}ol_2$ with $\mathsf{MINUS}_{\mathcal{P}ol_2} = x_1 - x_2$, $\mathsf{COND}_{\mathcal{P}ol_2} = x_2 - x_3$, $\mathsf{s}_{\mathcal{P}ol_2} = x_1 + 1$, and $f_{\mathcal{P}ol_2} = 0$ for all other function symbols f to prove innermost termination of Ex. 1. When trying to solve the remaining DP problem $\{(7), (9)\}$, the DP (9) would be strictly decreasing but none of the two DPs would be bounded. The reason is that we have neither $\mathsf{MINUS}(x, y) \succsim_{\mathcal{P}ol_2} \mathsf{c}$ nor $\mathsf{COND}(\mathsf{true}, x, y) \succsim_{\mathcal{P}ol_2} \mathsf{c}$ for any

[7] While such interpretations always result in well-founded orders, they are difficult to generate automatically. In contrast, the search for integer polynomial interpretations is as for ordinary polynomial interpretations, e.g., by using SAT solving as in [7].

[8] The reason is that constructor ground terms built from from 0 and s must be mapped to infinitely many different numbers. Therefore, the polynomial for round (which cannot be a constant) would have infinitely many extrema, which is a contradiction.

possible value of $\mathsf{c}_{\mathcal{P}ol_2}$. Thus, the reduction pair processor would return the two[9] DP problems $\{(7)\}$ and $\{(7), (9)\}$, i.e., it would not simplify the DP problem.

The solution is to consider *conditions* when requiring inequalities like $s \underset{(\sim)}{\succ} t$ or $s \succsim \mathsf{c}$. For example, to include the DP (7) in $\mathcal{P}_{bound}$, we do not have to demand $\mathsf{MINUS}(x, y) \succsim \mathsf{c}$ for *all* instantiations of x and y. Instead, it suffices to require the inequality only for those instantiations of x and y which can be used in potentially infinite minimal innermost chains. So we require $\mathsf{MINUS}(x, y) \succsim \mathsf{c}$ only for instantiations σ where (7)'s instantiated right-hand side $\mathsf{COND}(\mathsf{gt}(x, y), x, y)\sigma$ reduces to an instantiated left-hand side $u\sigma$ for some DP $u \to v$.[10] Here, $u \to v$ should again be variable renamed. As our DP problem contains two DPs (7) and (9), we get the following two constraints (by considering all possibilities $u \to v \in \{(7), (9)\}$). If both constraints are satisfied, then we can include (7) in $\mathcal{P}_{bound}$.

$$\mathsf{COND}(\mathsf{gt}(x, y), x, y) = \mathsf{MINUS}(x', y') \quad \Rightarrow \quad \mathsf{MINUS}(x, y) \succsim \mathsf{c} \tag{17}$$
$$\mathsf{COND}(\mathsf{gt}(x, y), x, y) = \mathsf{COND}(\mathsf{true}, x', y') \quad \Rightarrow \quad \mathsf{MINUS}(x, y) \succsim \mathsf{c} \tag{18}$$

Def. 10 introduces the syntax and semantics of such conditional constraints.

Definition 10 (Conditional Constraint). *For given relations $\succsim$ and $\succ$, the set $\mathcal{C}$ of conditional constraints is the smallest set with*

- $\{TRUE,\ s \succsim t,\ s \succ t,\ s = t\} \subseteq \mathcal{C}$ *for all terms s and t*
- *if $\{\varphi_1, \varphi_2\} \subseteq \mathcal{C}$, then $\varphi_1 \Rightarrow \varphi_2 \in \mathcal{C}$ and $\varphi_1 \wedge \varphi_2 \in \mathcal{C}$*
- *if $\varphi \in \mathcal{C}$ and $y \in \mathcal{V}$, then $\forall y \;\; \varphi \in \mathcal{C}$*

Now we define which normal $\mathcal{F}$-substitutions[11] σ satisfy a constraint $\varphi \in \mathcal{C}$, denoted "$\sigma \models \varphi$":

- $\sigma \models TRUE$ *for all normal $\mathcal{F}$-substitutions σ*
- $\sigma \models s \succsim t$ *iff* $s\sigma \succsim t\sigma$ *and* $\sigma \models s \succ t$ *iff* $s\sigma \succ t\sigma$
- $\sigma \models s = t$ *iff $s\sigma$ is innermost terminating, $s\sigma \xrightarrow{\mathrm{i}}^*_{\mathcal{R}} t\sigma$, $t\sigma$ is a normal form*
- $\sigma \models \varphi_1 \Rightarrow \varphi_2$ *iff* $\sigma \not\models \varphi_1$ *or* $\sigma \models \varphi_2$
- $\sigma \models \varphi_1 \wedge \varphi_2$ *iff* $\sigma \models \varphi_1$ *and* $\sigma \models \varphi_2$
- $\sigma \models \forall y\, \varphi$ *iff $\sigma' \models \varphi$ for all normal $\mathcal{F}$-substitutions σ' where $\sigma'(x) = \sigma(x)$ for all $x \neq y$*

A constraint φ is valid *("$\models \varphi$") iff $\sigma \models \varphi$ holds for* all *normal $\mathcal{F}$-substitutions σ.*

Now we refine the reduction pair processor by taking conditions into account. To this end, we modify the definition of $\mathcal{P}_{bound}$ and introduce $\mathcal{P}_{\succ}$ and $\mathcal{P}_{\succsim}$.

[9] Since $\{(7)\} \subseteq \{(7), (9)\}$, then it suffices to regard just the DP problem $\{(7), (9)\}$.

[10] Moreover, $\mathsf{COND}(\mathsf{gt}(x, y), x, y)\sigma$ must be innermost terminating, $\mathsf{COND}(\mathsf{gt}(x, y), x, y)\sigma \xrightarrow{\mathrm{i}}^*_{\mathcal{R}} u\sigma$, and $u\sigma$ must be in normal form, since we consider *minimal innermost* chains.

[11] A *normal $\mathcal{F}$-substitution* σ instantiates all variables by normal forms that do not contain tuple symbols (i.e., for any $x \in \mathcal{V}$, all function symbols in $\sigma(x)$ are from $\mathcal{F}$).

Theorem 11 (Conditional General Reduction Pair Processor). *Let $(\succsim, \succ)$ be a general reduction pair. Let c be a fresh constant and let*

$$\begin{aligned}\mathcal{P}_{\succ} &= \{\, s \to t \in \mathcal{P} \quad | \quad \models \textstyle\bigwedge_{u \to v \in \mathcal{P}} (t = u' \;\Rightarrow\; s \succ t) \,\}\\ \mathcal{P}_{\succsim} &= \{\, s \to t \in \mathcal{P} \quad | \quad \models \textstyle\bigwedge_{u \to v \in \mathcal{P}} (t = u' \;\Rightarrow\; s \succsim t) \,\}\\ \mathcal{P}_{bound} &= \{\, s \to t \in \mathcal{P} \quad | \quad \models \textstyle\bigwedge_{u \to v \in \mathcal{P}} (t = u' \;\Rightarrow\; s \succsim \mathsf{c}) \,\}\end{aligned}$$

where u' results from u by renaming its variables into fresh variables. Then the following DP processor Proc is sound. Here, $\mathcal{U}(\mathcal{P})$ is defined as in Def. 5.

$$Proc(\mathcal{P}) = \begin{cases} \{\, \mathcal{P} \setminus \mathcal{P}_{\succ},\ \mathcal{P} \setminus \mathcal{P}_{bound} \,\} & \text{if } \mathcal{P}_{\succ} \cup \mathcal{P}_{\succsim} = \mathcal{P} \text{ and } \mathcal{U}(\mathcal{P}) \subseteq \succsim \\ \{\, \mathcal{P} \,\} & \text{otherwise} \end{cases}$$

Proof. Let $\mathcal{P}_{\succ} \cup \mathcal{P}_{\succsim} = \mathcal{P}$ and $\mathcal{U}(\mathcal{P}) \subseteq \succsim$. Suppose that there is an infinite minimal innermost $(\mathcal{P}, \mathcal{R})$-chain $s_1 \to t_1, s_2 \to t_2, \ldots$ So there is a normal substitution σ where $t_i\sigma \xrightarrow{\text{i}}{}^{*}_{\mathcal{R}} s_{i+1}\sigma$ and where $t_i\sigma$ is innermost terminating for all i. Clearly, one can choose σ to be an $\mathcal{F}$-substitution. Assume that there are both infinitely many i with $s_i \to t_i \in \mathcal{P}_{\succ}$ and also infinitely many j with $s_j \to t_j \in \mathcal{P}_{bound}$. Since $t_i\sigma \xrightarrow{\text{i}}{}^{*}_{\mathcal{R}} s_{i+1}\sigma$ and $t_j\sigma \xrightarrow{\text{i}}{}^{*}_{\mathcal{R}} s_{j+1}\sigma$, we have $s_i\sigma \succ t_i\sigma$ and $s_j\sigma \succsim \mathsf{c}$. Similarly, we have $s_i\sigma \succsim t_i\sigma$ whenever $s_i\sigma \not\succ t_i\sigma$. As in the proof of Thm. 8, this leads to a contradiction. Thus, we cannot have both infinitely many strictly decreasing and infinitely many bounded DPs. Thus, the chain has an infinite tail from $\mathcal{P} \setminus \mathcal{P}_{\succ}$ or from $\mathcal{P} \setminus \mathcal{P}_{bound}$. □

To ease readability, in Thm. 11 we only consider the conditions resulting from *two* DPs $s \to t$ and $u \to v$ which follow each other in minimal innermost chains. To consider also conditions resulting from $n+1$ adjacent DPs, one would have to modify $\mathcal{P}_{\succ}$ as follows (of course, $\mathcal{P}_{\succsim}$ and $\mathcal{P}_{bound}$ have to be modified analogously).

$$\mathcal{P}_{\succ} = \{ s \to t \in \mathcal{P} \mid\ \models \bigwedge\nolimits_{u_1 \to v_1, \ldots, u_n \to v_n \in \mathcal{P}} (t = u_1' \wedge v_1' = u_2' \wedge \ldots \wedge v_{n-1}' = u_n' \;\Rightarrow\; s \succ t) \}$$

Here, the variables in u_i' and v_i' must be renamed in order to be disjoint to the variables in u_j' and v_j' for $j \neq i$. Moreover, instead of regarding DPs which *follow* $s \to t$ in chains, one could also regards DPs which *precede* $s \to t$. Then instead of (or in addition to) the premise "$t = u'$", one would have the premise "$v' = s$".

The question remains how to check whether conditional constraints are valid, since this requires reasoning about reductions resp. reachability. We now introduce a calculus of seven rules to simplify conditional constraints. For example, the constraint (17) is trivially valid, since its condition is unsatisfiable. The reason is that there is no substitution σ with $\sigma \models \mathsf{COND}(\mathsf{gt}(x, y), x, y) = \mathsf{MINUS}(x', y')$, since COND is no defined function symbol (i.e., it is a *constructor*) and therefore, COND-terms can only be reduced to COND-terms.

This leads to the first inference rule. In a conjunction $\varphi_1 \wedge \ldots \wedge \varphi_n$ of conditional constraints φ_i, these rules can be used to replace a conjunct φ_i by a new formula φ_i'. Of course, $TRUE \wedge \varphi$ can always be simplified to φ. Eventually, the goal is to remove all equalities "$p = q$" from the constraints. The soundness of the rules is shown in Thm. 14: if φ_i is replaced by φ_i', then $\models \varphi_i'$ implies $\models \varphi_i$.

I. **Constructor and Different Function Symbol**	
$\dfrac{f(p_1,...,p_n) = g(q_1,...,q_m) \wedge \varphi \;\Rightarrow\; \psi}{\mathit{TRUE}}$	if f is a constructor and $f \neq g$

Rule (II) handles conditions like $\mathsf{COND}(\mathsf{gt}(x,y),x,y) = \mathsf{COND}(\mathsf{true},x',y')$ where both terms start with the constructor COND. So (18) is transformed to

$$\mathsf{gt}(x,y) = \mathsf{true} \wedge x = x' \wedge y = y' \quad \Rightarrow \quad \mathsf{MINUS}(x,y) \succsim \mathsf{c} \tag{19}$$

II. **Same Constructors on Both Sides**	
$\dfrac{f(p_1,...,p_n) = f(q_1,...,q_n) \wedge \varphi \;\Rightarrow\; \psi}{p_1 = q_1 \wedge \ldots \wedge p_n = q_n \wedge \varphi \;\Rightarrow\; \psi}$	if f is a constructor

Rule (III) removes conditions of the form "$x = q$" or "$q = x$" by applying the substitution $[x/q]$ to the constraint.[12] So (19) is transformed to

$$\mathsf{gt}(x,y) = \mathsf{true} \quad \Rightarrow \quad \mathsf{MINUS}(x,y) \succsim \mathsf{c} \tag{20}$$

III. **Variable in Equation**			
$\dfrac{x{=}q \,\wedge\, \varphi \quad \Rightarrow\; \psi}{\varphi\,\sigma \;\Rightarrow\; \psi\,\sigma}$	if $x \in \mathcal{V}$ and $\sigma = [x/q]$	$\dfrac{q{=}x \,\wedge\, \varphi \quad \Rightarrow\; \psi}{\varphi\,\sigma \;\Rightarrow\; \psi\,\sigma}$	if $x{\in}\mathcal{V}$, q has no defined symbols, $\sigma{=}[x/q]$

Of course, one can also omit arbitrary conjuncts from the premise of an implication. To ease notation, we regard a conjunction as a set of formulas. So their order is irrelevant and we write $\varphi' \subseteq \varphi$ iff all conjuncts of φ' are also conjuncts of φ. The empty conjunction is TRUE (i.e., $\mathit{TRUE} \Rightarrow \psi$ can be simplified to ψ).

IV. **Delete Conditions**	
$\dfrac{\varphi \;\Rightarrow\; \psi}{\varphi' \;\Rightarrow\; \psi}$	if $\varphi' \subseteq \varphi$

Rule (IV) is especially useful for omitting conditions $q = x$ where x is a variable which does not occur anywhere else. So one could also transform (19) to (20) by Rule (IV). The meaning of (20) is that $\mathsf{MINUS}(x,y)\sigma \succsim \mathsf{c}$ must hold

[12] To remove the condition $q = x$, we must ensure that for any normal $\mathcal{F}$-substitution δ, the term $q\delta$ is normal, too. Otherwise, Rule (III) would not be sound: Consider the TRS $\{\mathsf{f}(\mathsf{a}) \to \mathsf{b}\}$ and the constraint "$y = \mathsf{a} \wedge \mathsf{f}(y) = x \wedge x = x \Rightarrow \mathsf{a} = \mathsf{b}$". This constraint is not valid, since it is not satisfied by the substitution δ with $\delta(y) = \mathsf{a}$ and $\delta(x) = \mathsf{b}$. But if Rule (III) were allowed to omit the condition $\mathsf{f}(y) = x$ and to replace x by the normal form $\mathsf{f}(y)$, then we would obtain "$y{=}\mathsf{a}{\wedge}\mathsf{f}(y){=}\mathsf{f}(y) \Rightarrow \mathsf{a}{=}\mathsf{b}$". This constraint is valid since "$y{=}\mathsf{a}{\wedge}\mathsf{f}(y){=}\mathsf{f}(y)$" is unsatisfiable (as $\mathsf{f}(\mathsf{a})$ is no normal form).

whenever $\mathsf{gt}(x,y)\sigma$ is innermost terminating and $\mathsf{gt}(x,y)\sigma \xrightarrow{\mathrm{i}}{}^*_{\mathcal{R}} \mathsf{true}$ holds for a normal $\mathcal{F}$-substitution σ. To simplify this constraint further, the next inference rule performs an *induction* on the length of $\mathsf{gt}(x,y)\sigma$'s reduction.[13] Since $\mathsf{gt}(x,y)$ and true do not unify, at least one reduction step is needed, i.e., some rule $\mathsf{gt}(\ell_1,\ell_2) \to r$ must be applied. To detect all possibilities for the first reduction step, we consider all *narrowings* of the term $\mathsf{gt}(x,y)$. We obtain

$$\mathsf{gt}(x,y) \rightsquigarrow_{[x/\mathsf{0},y/v]} \mathsf{false}, \quad \mathsf{gt}(x,y) \rightsquigarrow_{[x/\mathsf{s}(u),y/\mathsf{0}]} \mathsf{true}, \quad \mathsf{gt}(x,y) \rightsquigarrow_{[x/\mathsf{s}(u),y/\mathsf{s}(v)]} \mathsf{gt}(u,v)$$

Thus, we could replace (20) by the following three new constraints where we always apply the respective narrowing substitution to the whole constraint:

$$\mathsf{false} = \mathsf{true} \quad \Rightarrow \quad \mathsf{MINUS}(\mathsf{0},v) \succsim \mathsf{c} \tag{21}$$
$$\mathsf{true} = \mathsf{true} \quad \Rightarrow \quad \mathsf{MINUS}(\mathsf{s}(u),\mathsf{0}) \succsim \mathsf{c} \tag{22}$$
$$\mathsf{gt}(u,v) = \mathsf{true} \quad \Rightarrow \quad \mathsf{MINUS}(\mathsf{s}(u),\mathsf{s}(v)) \succsim \mathsf{c} \tag{23}$$

So to transform a constraint $f(x_1,\ldots,x_n) = q \wedge \varphi \;\Rightarrow\; \psi$, we consider all rules $f(\ell_1,\ldots,\ell_n) \to r$. Then the constraint could be replaced by the new constraints

$$r = q\sigma \wedge \varphi\sigma \;\Rightarrow\; \psi\sigma, \qquad \text{where } \sigma = [x_1/\ell_1,\ldots,x_n/\ell_n]. \tag{24}$$

However, we perform a better transformation. Suppose that r contains a recursive call, i.e., a subterm $f(r_1,\ldots,r_n)$, and that the r_i do not contain defined symbols. Obviously, $f(r_1,\ldots,r_n)\sigma$'s reduction is shorter than the reduction of $f(x_1,\ldots,x_n)\sigma$. Thus for $\mu = [x_1/r_1,\ldots,x_n/r_n]$ one can assume

$$\forall y_1,\ldots,y_m \quad f(r_1,\ldots,r_n) = q\mu \wedge \varphi\mu \;\Rightarrow\; \psi\mu \tag{25}$$

as *induction hypothesis* when requiring (24).[14] Here, $y_1,\ldots,y_m$ are all occurring variables except those in r. Of course, we may assume that variables in rewrite

[13] More precisely, we use an induction on $\xrightarrow{\mathrm{i}}_{\mathcal{R}} \circ \trianglerighteq$, where $\trianglerighteq$ is the subterm relation. The idea for this inference rule was inspired by our earlier work on termination of simple first-order functional programs [3]. But [3] only considered a very restricted form of functional programs (left-linear, sufficiently complete, non-overlapping constructor systems without defined symbols in arguments of recursive calls), whereas we regard arbitrary TRSs. Moreover, we integrate this idea of performing induction into the whole framework of termination techniques and tools available for TRSs. Finally, in contrast to [3], we do not need an underlying induction theorem prover. Nevertheless, our approach is significantly stronger (e.g., [3] fails on examples like Ex. 12, cf. the appendix).

[14] If there is more than one recursive call in r, then one can obtain a corresponding induction hypothesis (25) for each recursive call. But for a similar reason as in Footnote 12, if the r_i contain defined symbols, then one may not assume (25) as induction hypothesis. The reason is that we only prove the claims for *normal* substitutions σ. For that reason, (25) is only an instance of our original claim, if $\mu\sigma$ is normal whenever σ is normal. Otherwise, we could falsely prove innermost termination of the TRS $\{\mathsf{f}(\mathsf{a},x,y) \to \mathsf{f}(\mathsf{g}(x,x,x),x,\mathsf{s}(y)),\ \mathsf{g}(\mathsf{s}(z),\mathsf{s}(z),\mathsf{s}(z)) \to \mathsf{g}(\mathsf{e},\mathsf{e},\mathsf{e}),\ \mathsf{g}(\mathsf{a},\mathsf{b},z) \to \mathsf{a},\ \mathsf{e} \to \mathsf{a},\ \mathsf{e} \to \mathsf{b}\}$. We use a polynomial interpretation $\mathcal{P}ol$ with $\mathsf{F}_{\mathcal{P}ol} = -x_3$, $\mathsf{G}_{\mathcal{P}ol} = x_1$, $\mathsf{s}_{\mathcal{P}ol} = x_1 + 1$, and $\mathsf{e}_{\mathcal{P}ol} = 0$.

rules (i.e., in r) are disjoint from variables in constraints (i.e., in q, φ, and ψ). So instead of (24), it suffices to demand (25) $\Rightarrow$ (24), or equivalently

$$r = q\sigma \wedge \varphi\sigma \wedge (25) \;\Rightarrow\; \psi\sigma. \tag{26}$$

This leads to Rule (V). Here, $x_1, \ldots, x_n$ denote pairwise different variables.

V. **Induction (Defined Symbol with Pairwise Different Variables)**

$$\frac{f(x_1, ..., x_n) = q \quad \wedge\, \varphi \qquad\qquad \Rightarrow\; \psi}{\bigwedge\limits_{f(\ell_1,...,\ell_n) \to r \in \mathcal{R}} \quad (\, r = q\,\sigma \wedge \varphi\,\sigma \wedge \varphi' \;\Rightarrow\; \psi\,\sigma\,)}$$

if f is a defined symbol and $f(x_1, ..., x_n)$ does not unify with q

where $\sigma = [x_1/\ell_1, ..., x_n/\ell_n]$

$$\text{and} \quad \varphi' = \begin{cases} \forall y_1, ..., y_m \;\; f(r_1, \ldots, r_n) = q\mu \wedge \varphi\,\mu \Rightarrow \psi\,\mu, & \text{if} \\ \quad \bullet\ r \text{ contains the subterm } f(r_1, ..., r_n), \\ \quad \bullet\ \text{there is no defined symbol in any } r_i, \\ \quad \bullet\ \mu = [x_1/r_1, ..., x_n/r_n], \text{ and} \\ \quad \bullet\ y_1, ..., y_m \text{ are all occurring variables except } \mathcal{V}(r) \\ \mathit{TRUE}, \quad \text{otherwise} \end{cases}$$

In our example, the above rule transforms the original constraint (20) into the three new constraints (21), (22), and (27). Here, (27) is obtained from the narrowing step $\mathsf{gt}(x, y) \rightsquigarrow_{[x/\mathsf{s}(u), y/\mathsf{s}(v)]} \mathsf{gt}(u, v)$, i.e., we have $\sigma = [x/\mathsf{s}(u), y/\mathsf{s}(v)]$, $r_1 = u$, $r_2 = v$, and $\mu = [x/u, y/v]$. There are no variables $y_1, \ldots, y_m$.

$$\begin{array}{l} \mathsf{gt}(u, v) = \mathsf{true} \\ \wedge\, (\mathsf{gt}(u, v) = \mathsf{true} \;\Rightarrow\; \mathsf{MINUS}(u, v) \succsim \mathsf{c}) \quad \Rightarrow \quad \mathsf{MINUS}(\mathsf{s}(u), \mathsf{s}(v)) \succsim \mathsf{c} \end{array} \tag{27}$$

To simplify (27) further, now we can "apply" the induction hypothesis, since its condition $\mathsf{gt}(u, v) = \mathsf{true}$ is guaranteed to hold. So we can transform (27) to

$$\mathsf{gt}(u, v) = \mathsf{true} \;\wedge\; \mathsf{MINUS}(u, v) \succsim \mathsf{c} \;\Rightarrow\; \mathsf{MINUS}(\mathsf{s}(u), \mathsf{s}(v)) \succsim \mathsf{c}. \tag{28}$$

In general, to simplify conditions one may of course also instantiate universally

---

Both DPs are strictly decreasing and the G-DP is obviously bounded from below. For the boundedness of the F-DP we would obtain the constraint $\mathsf{g}(x, x, x) = \mathsf{a} \Rightarrow \mathsf{F}(\mathsf{a}, x, y) \succsim \mathsf{c}$. The only narrowing is $\mathsf{g}(x, x, x) \rightsquigarrow_{[x/\mathsf{s}(z)]} \mathsf{g}(\mathsf{e}, \mathsf{e}, \mathsf{e})$. If we allowed such non-constructor substitutions μ, then we would only get the new constraint

$$\mathsf{g}(\mathsf{e}, \mathsf{e}, \mathsf{e}) = \mathsf{a} \wedge (\mathsf{g}(\mathsf{e}, \mathsf{e}, \mathsf{e}) = \mathsf{a} \Rightarrow \mathsf{F}(\mathsf{a}, \mathsf{e}, y) \succsim \mathsf{c}) \Rightarrow \mathsf{F}(\mathsf{a}, \mathsf{s}(z), y) \succsim \mathsf{c}.$$

This can be simplified to $\mathsf{F}(\mathsf{a}, \mathsf{e}, y) \succsim \mathsf{c} \Rightarrow \mathsf{F}(\mathsf{a}, \mathsf{s}(z), y) \succsim \mathsf{c}$ which is satisfied by the above polynomial interpretation.

quantified variables.[15] This leads to the following rule.

VI. **Simplify Condition**

$$\frac{\varphi \,\wedge\, (\forall y_1,\ldots,y_m \;\; \varphi' \Rightarrow \psi' \;) \;\Rightarrow\; \psi}{\varphi \,\wedge\, \qquad\qquad\qquad \psi'\,\sigma \;\Rightarrow\; \psi}$$

if $DOM(\sigma) \subseteq \{y_1,\ldots,y_m\}$, there is no defined symbol and no tuple symbol in any $\sigma(y_i)$, and $\varphi'\,\sigma \subseteq \varphi$

To simplify the remaining constraints (21), (22), and (28), note that (21) can be eliminated by Rule (I) since it has an unsatisfiable condition $\mathsf{false} = \mathsf{true}$. Moreover, Rule (II) can delete the trivial condition $\mathsf{true} = \mathsf{true}$ of the constraint (22). For (28), with Rule (IV) one can of course always omit conditions like $\mathsf{gt}(u,v) = \mathsf{true}$ from conditional constraints. In this way, all conditions with equalities $p = q$ are removed in the end.

So to finish the termination proof of Ex. 1, we can include the DP (7) in $\mathcal{P}_{bound}$ if the constraints $\mathsf{MINUS}(\mathsf{s}(u),\mathsf{0}) \succsim \mathsf{c}$ and $\mathsf{MINUS}(u,v) \succsim \mathsf{c} \;\Rightarrow\; \mathsf{MINUS}(\mathsf{s}(u),\mathsf{s}(v)) \succsim \mathsf{c}$ are satisfied. Of course, these constraints obviously hold for $\mathcal{P}ol_2$ if we choose $\mathsf{c}_{\mathcal{P}ol_2} = 1$. Then the DP (9) is strictly decreasing and (7) is bounded from below and thus, the reduction pair processor transforms the remaining DP problem $\{(7),(9)\}$ into $\{(7)\}$ and $\{(9)\}$. Now the resulting DP problems are easy to solve and thus, innermost termination of Ex. 1 is proved.

The rules (I) - (VI) are not always sufficient to exploit the conditions of a constraint. We demonstrate this with the following example.

Example 12. We regard a TRS $\mathcal{R}$ containing the gt*-rules (4) - (6) together with*

$$\begin{array}{rlrl} \mathsf{plus}(n,\mathsf{0}) & \to n & \mathsf{f}(\mathsf{true},x,y,z) & \to \mathsf{f}(\mathsf{gt}(x,\mathsf{plus}(y,z)),x,\mathsf{s}(y),z) \\ \mathsf{plus}(n,\mathsf{s}(m)) & \to \mathsf{s}(\mathsf{plus}(n,m)) & \mathsf{f}(\mathsf{true},x,y,z) & \to \mathsf{f}(\mathsf{gt}(x,\mathsf{plus}(y,z)),x,y,\mathsf{s}(z)) \end{array}$$

The termination of gt and of plus is easy to show. So the initial DP problem can easily be transformed into $\{(29),(30)\}$ with

$$\mathsf{F}(\mathsf{true},x,y,z) \to \mathsf{F}(\mathsf{gt}(x,\mathsf{plus}(y,z)),x,\mathsf{s}(y),z) \tag{29}$$

$$\mathsf{F}(\mathsf{true},x,y,z) \to \mathsf{F}(\mathsf{gt}(x,\mathsf{plus}(y,z)),x,y,\mathsf{s}(z)) \tag{30}$$

To include (29) in $\mathcal{P}_{bound}$, we have to impose the following constraint:

$$\mathsf{F}(\mathsf{gt}(x,\mathsf{plus}(y,z)),x,\mathsf{s}(y),z) = \mathsf{F}(\mathsf{true},x',y',z') \;\;\Rightarrow\;\; \mathsf{F}(\mathsf{true},x,y,z) \succsim \mathsf{c} \tag{31}$$

[15] For a similar reason as in Footnote 12, one may only instantiate them by terms without defined symbols. Otherwise, we lose soundness. To see this, consider the TRS $\{\mathsf{f}(\mathsf{a}) \to \mathsf{a}\}$ and the constraint $x = \mathsf{a} \wedge (\forall y \; \mathit{TRUE} \Rightarrow y = y) \Rightarrow \mathsf{a} = \mathsf{b}$. Clearly, this constraint is not valid (for example, the substitution $\delta = [x/\mathsf{a}]$ is not a model of this formula). However, if we were allowed to apply rule (VI) with the normal substitution $\sigma = [y/\mathsf{f}(x)]$ then we would obtain $x = \mathsf{a} \wedge \mathsf{f}(x) = \mathsf{f}(x) \Rightarrow \mathsf{a} = \mathsf{b}$. This constraint is valid, i.e., it is satisfied by every normal $\mathcal{F}$-substitution δ. The reason is that $\delta(x) = \mathsf{a}$ would imply that the right-hand side $\mathsf{f}(\mathsf{a})$ of $(\mathsf{f}(x) = \mathsf{f}(x))\delta$ is not in normal form. Thus, then the premise of the implication does not hold.

With the rules (II) and (IV), it can be transformed into

$$\mathsf{gt}(x, \mathsf{plus}(y, z)) = \mathsf{true} \;\;\Rightarrow\;\; \mathsf{F}(\mathsf{true}, x, y, z) \succsim \mathsf{c} \tag{32}$$

Now we want to use induction. However, Rule (V) is only applicable for conditions $f(x_1, \ldots, x_n) = q$ where $x_1, \ldots, x_n$ are *pairwise different variables*. To obtain such conditions, we use the following rule. Here, x denotes a fresh variable.

VII. **Defined Symbol without Pairwise Different Variables**	
$\dfrac{f(p_1, \ldots, p_i, \ldots, p_n) = q \,\wedge\, \varphi \;\Rightarrow\; \psi}{p_i = x \wedge f(p_1, \ldots, \; x \;, \ldots, p_n) = q \,\wedge\, \varphi \;\Rightarrow\; \psi}$	if f is a defined symbol and ($p_i \notin \mathcal{V}$ or $p_i = p_j$ for a $j \neq i$)

So the constraint (32) is transformed into

$$\mathsf{plus}(y, z) = w \wedge \mathsf{gt}(x, w) = \mathsf{true} \;\Rightarrow\; \mathsf{F}(\mathsf{true}, x, y, z) \succsim \mathsf{c}$$

Example 13. To continue, we can now perform induction on gt *which yields*

$$\mathsf{plus}(y, z) = v \wedge \mathsf{false} = \mathsf{true} \Rightarrow \mathsf{F}(\mathsf{true}, 0, y, z) \succsim \mathsf{c} \tag{33}$$
$$\mathsf{plus}(y, z) = 0 \wedge \mathsf{true} = \mathsf{true} \Rightarrow \mathsf{F}(\mathsf{true}, \mathsf{s}(u), y, z) \succsim \mathsf{c} \tag{34}$$
$$\mathsf{plus}(y, z) = \mathsf{s}(v) \wedge \mathsf{gt}(u, v) = \mathsf{true} \wedge (36) \Rightarrow \mathsf{F}(\mathsf{true}, \mathsf{s}(u), y, z) \succsim \mathsf{c} \tag{35}$$

Here, (36) is the induction hypothesis:

$$\forall y, z \;\; \mathsf{plus}(y, z) = v \wedge \mathsf{gt}(u, v) = \mathsf{true} \;\Rightarrow\; \mathsf{F}(\mathsf{true}, u, y, z) \succsim \mathsf{c} \tag{36}$$

With Rule (I) we delete constraint (33) and Rule (II) simplifies constraint (34) to "$\mathsf{plus}(y, z) = 0 \;\Rightarrow\; \mathsf{F}(\mathsf{true}, \mathsf{s}(u), y, z) \succsim \mathsf{c}$*". Similar to our previous example, by induction via* plus *and by removing the constraint with the unsatisfiable condition* $\mathsf{s}(\mathsf{plus}(n, m)) = 0$*, we finally transform it to*

$$\mathsf{F}(\mathsf{true}, \mathsf{s}(u), 0, 0) \succsim \mathsf{c} \tag{37}$$

The other constraint (35) is simplified further by induction via plus *as well:*

$$n = \mathsf{s}(v) \wedge \mathsf{gt}(u, v) = \mathsf{true} \wedge (36) \Rightarrow \mathsf{F}(\mathsf{true}, \mathsf{s}(u), n, 0) \succsim \mathsf{c} \tag{38}$$
$$\mathsf{s}(\mathsf{plus}(n, m)) = \mathsf{s}(v) \wedge \mathsf{gt}(u, v) = \mathsf{true} \wedge (36) \wedge \varphi' \Rightarrow \mathsf{F}(\mathsf{true}, \mathsf{s}(u), n, \mathsf{s}(m)) \succsim \mathsf{c} \tag{39}$$

where φ' *is the new induction hypothesis. We apply Rules (III) and (IV) on (38) to obtain "*$\mathsf{gt}(u, v) = \mathsf{true} \;\Rightarrow\; \mathsf{F}(\mathsf{true}, \mathsf{s}(u), \mathsf{s}(v), 0) \succsim \mathsf{c}$*". By another induction on* gt *and by applying Rules (I), (II), (IV), and (VI) we get the final constraints*

$$\mathsf{F}(\mathsf{true}, \mathsf{s}(\mathsf{s}(i)), \mathsf{s}(0), 0) \succsim \mathsf{c} \tag{40}$$
$$\mathsf{F}(\mathsf{true}, \mathsf{s}(i), \mathsf{s}(j), 0) \succsim \mathsf{c} \Rightarrow \mathsf{F}(\mathsf{true}, \mathsf{s}(\mathsf{s}(i)), \mathsf{s}(\mathsf{s}(j)), 0) \succsim \mathsf{c} \tag{41}$$

In the only remaining constraint (39) we delete φ' *with Rule (IV) and by removing the outermost* s *in the first condition with Rule (II), we get*

$$\mathsf{plus}(n, m) = v \wedge \mathsf{gt}(u, v) = \mathsf{true} \wedge (36) \;\Rightarrow\; \mathsf{F}(\mathsf{true}, \mathsf{s}(u), n, \mathsf{s}(m)) \succsim \mathsf{c}$$

Now we can simplify the condition by applying the induction hypothesis (36). In (36), the variables y and z were universally quantified. We instantiate y with n and z with m. With Rule (VI) we replace (36) by the new condition $\mathsf{F}(\mathsf{true}, u, n, m) \succsim \mathsf{c}$. By deleting the first two remaining conditions we finally get

$$\mathsf{F}(\mathsf{true}, u, n, m) \succsim \mathsf{c} \;\Rightarrow\; \mathsf{F}(\mathsf{true}, \mathsf{s}(u), n, \mathsf{s}(m)) \succsim \mathsf{c} \tag{42}$$

So to summarize, the constraint (31) can be transformed into (37), (40), (41), and (42). These constraints are satisfied by the interpretation $\mathcal{P}ol$ where $\mathsf{F}_{\mathcal{P}ol} = x_2 - x_3 - x_4$, $\mathsf{s}_{\mathcal{P}ol} = x_1 + 1$, $\mathsf{0}_{\mathcal{P}ol} = 0$, and $\mathsf{c}_{\mathcal{P}ol} = 1$. Therefore, we can include the DP (29) in $\mathcal{P}_{bound}$. For a similar reason, the other DP (30) is also bounded. Moreover, both DPs are strictly decreasing and there are no usable rules since F is not $\succsim_{\mathcal{P}ol}$-dependent on 1. Hence, the reduction pair processor can remove both DPs and innermost termination of Ex. 12 is proved.

We define $\varphi \vdash \varphi'$ iff φ' results from φ by repeatedly applying the above inference rules to the conjuncts of φ. Thm. 14 states that these rules are sound.

Theorem 14 (Soundness). *If $\varphi \vdash \varphi'$, then $\models \varphi'$ implies $\models \varphi$.*

Proof. Rule (I) is sound, since $\delta \not\models f(p_1, \ldots, p_n) = g(q_1, \ldots, q_m)$ holds for all substitutions δ if f is a constructor and $f \neq g$. The reason is that $f(\ldots)\delta$ cannot reduce to $g(\ldots)\delta$.

The soundness of Rule (II) follows since $\delta \models p_1 = q_1 \wedge \ldots \wedge p_n = q_n$ holds iff $\delta \models f(p_1, \ldots, p_n) = f(q_1, \ldots, q_n)$ for any constructor f. The reason is that every $p_i\delta$ is innermost terminating iff $f(p_1, \ldots, p_n)\delta$ is innermost terminating, that $f(p_1, \ldots, p_n)\delta \xrightarrow{\mathrm{i}}{}^*_{\mathcal{R}} f(q_1, \ldots, q_n)\delta$ is equivalent to $p_i\delta \xrightarrow{\mathrm{i}}{}^*_{\mathcal{R}} q_i\delta$, and that every $q_i\delta$ is in normal form iff $f(q_1, \ldots, q_n)\delta$ is in normal form.

For Rule (III), note that if $\delta \models x = q$, then we have $x\delta = q\delta$ (since δ must be normal substitution) and $q\delta$ is a normal form. Similarly, if $\delta \models q = x$ and q does not contain defined symbols, then $q\delta$ is a normal form and $x\delta = q\delta$. For any normal $\mathcal{F}$-substitution δ where $x\delta = q\delta$ and where $q\delta$ is in normal form and for any constraint φ, we have $\delta \models \varphi[x/q]$ iff $\delta \models \varphi$.[16] This implies the soundness of Rule (III).

[16] Note that

$$\sigma_2 \models \varphi\sigma_1 \text{ iff } \sigma_1\sigma_2 \models \varphi \tag{43}$$

holds for all constraints φ and all $\mathcal{F}$-substitutions σ_i where σ_2 and $\sigma_1\sigma_2$ are normal. With this observation, the above claim immediately follows since $[x/q]\delta = \delta$ and since δ and $q\delta$ are normal.

The observation (43) can be proved by a straightforward structural induction on φ. It is clear for $\varphi = TRUE$.

If φ has the form $s \,{}_{(}\!\succsim\!{}_{)}\, t$, then we have $\sigma_2 \models s\sigma_1 \,{}_{(}\!\succsim\!{}_{)}\, t\sigma_1$ iff $s\sigma_1\sigma_2 \,{}_{(}\!\succsim\!{}_{)}\, t\sigma_1\sigma_2$ iff $\sigma_1\sigma_2 \models s \,{}_{(}\!\succsim\!{}_{)}\, t$.

If φ has the form $s = t$, then we have $\sigma_2 \models s\sigma_1 = t\sigma_1$ iff $s\sigma_1\sigma_2$ is innermost terminating, $s\sigma_1\sigma_2 \xrightarrow{\mathrm{i}}{}^*_{\mathcal{R}} t\sigma_1\sigma_2$, and $t\sigma_1\sigma_2$ is a normal form. This is equivalent to $\sigma_1\sigma_2 \models s = t$.

If φ has the form $s \wedge t$ or $s \Rightarrow t$, then (43) immediately follows from the induction hypothesis.

The soundness of Rule (IV) is obvious, since $\delta \models \varphi' \;\Rightarrow\; \psi$ implies $\delta \models \varphi \;\Rightarrow\; \psi$ whenever $\varphi' \subseteq \varphi$.

For Rule (V), we have to prove that $\delta \models f(x_1,\ldots,x_n) = q \wedge \varphi \;\Rightarrow\; \psi$ holds for all normal $\mathcal{F}$-substitutions δ. As this is obviously true when $\delta \not\models f(x_1,\ldots,x_n) = q$ we only have to consider substitutions δ where $f(x_1,\ldots,x_n)\delta$ is innermost terminating and where $f(x_1,\ldots,x_n)\delta \xrightarrow{\mathrm{i}}{}^*_{\mathcal{R}} q\delta$. Innermost termination allows us to perform induction on $f(x_1,\ldots,x_n)\delta$ w.r.t. the induction relation $\xrightarrow{\mathrm{i}}_{\mathcal{R}} \circ \trianglerighteq$, where $\trianglerighteq$ is the subterm relation, cf. Footnote 13. Since $f(x_1,\ldots,x_n)$ and q do not unify, at least one reduction step is needed for $f(x_1,\ldots,x_n)\delta \xrightarrow{\mathrm{i}}{}^*_{\mathcal{R}} q\delta$. Since δ is a normal substitution, the first reduction step takes place on the root position, i.e., it is performed with a rule $f(\ell_1,\ldots,\ell_n) \to r$. Let $\sigma = [x_1/\ell_1,\ldots,x_n/\ell_n]$. Then there is a normal $\mathcal{F}$-substitution σ' with $\delta = \sigma\sigma'$ and $f(x_1,\ldots,x_n)\delta = f(\ell_1,\ldots,\ell_n)\sigma' \xrightarrow{\mathrm{i}}_{\mathcal{R}} r\sigma' \xrightarrow{\mathrm{i}}{}^*_{\mathcal{R}} q\delta = q\sigma\sigma'$, where $r\sigma'$ is also innermost terminating. If "$\models\; r = q\sigma \wedge \varphi\sigma \wedge \varphi' \Rightarrow \psi\sigma$", then this constraint is also satisfied by σ'. Since $\sigma' \models r = q\sigma$ as shown above, we have

$$\sigma' \models \varphi\sigma \wedge \varphi' \Rightarrow \psi\sigma. \tag{44}$$

It remains to show that $\sigma' \models \varphi'$. Then (44) implies $\sigma' \models \varphi\sigma \Rightarrow \psi\sigma$, i.e.,[17] $\sigma\sigma' \models \varphi \Rightarrow \psi$ and thus, $\delta \models \varphi \;\Rightarrow\; \psi$.

Now we show that $\sigma' \models \varphi'$. If $\varphi' \neq \mathit{TRUE}$, then we have to prove $\tau \models f(x_1,\ldots,x_n)\mu = q\mu \wedge \varphi\mu \Rightarrow \psi\mu$ for all normal $\mathcal{F}$-substitutions τ that differ from σ' only on $y_1,\ldots,y_m$. As r does not contain any variable y_i we obtain $r_i\tau = r_i\sigma'$. Note that μ instantiates all terms by terms without defined or tuple symbols. So $\mu\tau$ is a normal $\mathcal{F}$-substitution and by the observation in Footnote 43 it is equivalent to prove $\mu\tau \models f(x_1,\ldots,x_n) = q \wedge \varphi \Rightarrow \psi$. But as $f(x_1,\ldots,x_n)\delta \xrightarrow{\mathrm{i}}_{\mathcal{R}} r\sigma' \trianglerighteq f(r_1,\ldots,r_n)\sigma' = f(r_1,\ldots,r_n)\tau = f(x_1,\ldots,x_n)\mu\tau$ this follows from the induction hypothesis.

For Rule (VI), let $\delta \models \varphi \wedge (\forall y_1,\ldots y_m \;\; \varphi' \Rightarrow \psi')$. We define $\delta' = \sigma\delta$. Since $\mathit{DOM}(\sigma) \subseteq \{y_1,\ldots,y_m\}$, δ' and δ differ at most on the variables $y_1,\ldots,y_m$. Moreover, as σ is a constructor $\mathcal{F}$-substitution and as δ is a normal $\mathcal{F}$-substitution we also know that δ' is a normal $\mathcal{F}$-substitution. Hence, by definition $\delta' \models \varphi' \Rightarrow \psi'$. Using the result in Footnote 43 we have $\delta \models \varphi \wedge (\varphi'\sigma \Rightarrow \psi'\sigma)$. Thus, $\varphi'\,\sigma \subseteq \varphi$ implies $\delta \models \varphi \wedge \psi'\sigma$. Now $\models \varphi \wedge \psi'\sigma \Rightarrow \psi$ implies $\delta \models \psi$.

Finally, for Rule (VII) let $\delta \models f(p_1,\ldots,p_i,\ldots,p_n) = q$. Then we have $f(p_1,\ldots,p_i,\ldots,p_n)\delta = f(p_1\delta,\ldots,p_i\delta,\ldots,p_n\delta) \xrightarrow{\mathrm{i}}{}^*_{\mathcal{R}} f(p_1\delta,\ldots,q',\ldots,p_n\delta) \xrightarrow{\mathrm{i}}{}^*_{\mathcal{R}} q\delta$ where $p_i\delta \xrightarrow{\mathrm{i}}{}^*_{\mathcal{R}} q'$ and q' is a normal form. Moreover, $f(p_1,\ldots,p_i,\ldots,p_n)\delta$ is innermost terminating and thus, the terms $p_i\delta$ and $f(p_1\delta,\ldots,q',\ldots,p_n\delta)$ are

---

If φ has the form $\forall y\ \varphi'$, then w.l.o.g. we may assume that y is a fresh variable which does not occur in $\mathit{DOM}(\sigma_1)$ or in $\sigma_1(x)$ for $x \in \mathit{DOM}(\sigma_1)$. Thus, $\sigma_2 \models (\forall y\ \varphi')\sigma_1$ iff $\sigma_2 \models \forall y\ (\varphi'\sigma_1)$. This is equivalent to the requirement that $\sigma_2' \models \varphi'\sigma_1$ for all substitutions σ_2' which are like σ_2 on all variables except y. By the induction hypothesis, this holds iff $\sigma_1\sigma_2' \models \varphi'$ for all such substitutions σ_2'. Hence, this holds iff $\sigma \models \varphi'$ for all substitutions σ which are like $\sigma_1\sigma_2$ on all variables except y. This is equivalent to $\sigma_1\sigma_2 \models \forall y\ \varphi'$.

[17] This again follows from the observation (43) since σ' and $\delta = \sigma\sigma'$ are normal.

innermost terminating as well. Let δ' be like δ on all variables except x and let $x\delta' = q'$. Then we have $\delta' \models p_i = x$ and $\delta' \models f(p_1, \ldots, x, \ldots, p_n) = q$. Thus we also obtain $\delta' \models \varphi \Rightarrow \psi$ and hence, $\delta \models \varphi \Rightarrow \psi$. $\square$

With Thm. 14 we can now refine the reduction pair processor from Thm. 11.

Corollary 15 (Conditional General Reduction Pair Processor with Inference). *Let $(\succsim, \succ)$ be a general reduction pair and let c be a fresh constant. For every $s \to t \in \mathcal{P}$ and every inequality $\psi \in \{\, s \succ t,\ s \succsim t,\ s \succsim \mathsf{c}\,\}$, let φ_ψ be a constraint with $\bigwedge_{u \to v \in \mathcal{P}} (t = u' \;\Rightarrow\; \psi) \;\vdash\; \varphi_\psi$. Here, u' results from u by renaming its variables into fresh variables. Then the processor Proc from Thm. 11 is still sound if we define $\mathcal{P}_\succ = \{s \to t \in \mathcal{P} \mid\ \models \varphi_{s \succ t}\,\}$, $\mathcal{P}_\succsim = \{s \to t \in \mathcal{P} \mid\ \models \varphi_{s \succsim t}\,\}$, and $\mathcal{P}_{bound} = \{s \to t \in \mathcal{P} \mid\ \models \varphi_{s \succsim \mathsf{c}}\,\}$.*

For automation, one of course needs a strategy for the application of the rules (I) - (VII). We propose the following heuristic:

- Apply the rules with the priority (I), (II), (IV)$'$, (VI)$'$, (III)$'$, (VII), (V), where
  - (IV)$'$ is a restriction of (IV) which only deletes conditions $q = x$ where x is a variable which does not occur anywhere else.
  - (VI)$'$ is a restriction of (VI), i.e., it simplifies conditions $\varphi \wedge (\forall \ldots \varphi' \Rightarrow \psi')$. Hence, every conjunct of φ' must be mapped to a conjunct of φ that is an instance of it. The difference to (VI) is that this mapping must be *injective*. In other words, in the condition "$\varphi'\sigma \subseteq \varphi$" we do not regard conjunctions as sets, but as *multisets*.
  - (III)$'$ is a restriction of (III) which is not applied for conditions $x = q$ or $q = x$ where x is the argument of a defined symbol in the left-hand side of an equation. The reason is that then its effect would be again "revised" by Rule (VII).
- When applying induction (Rule (V)), we prefer those conditions of the form $f(x_1, \ldots, x_n) = q$ which were used least often for induction up to now. Moreover, if $f(x_1, \ldots, x_n) = q$ already resulted from a previous induction, then before applying induction again, we first delete its induction hypothesis with Rule (IV).
- To ensure termination of the inference rules, one has to impose some limit on the number of possible inductions with Rule (V). In the end, one applies the rules (I), (II), (III), and (VI) as often as possible. Afterwards, we use Rule (IV) to remove all remaining conditions containing "$=$" or "$\Rightarrow$". Moreover, if there are several conditions of the form $s \underset{(\sim)}{\succ} t$, we remove all but one of them.

Thus, the constraints φ_ψ in Cor. 15 are conjunctions where the conjuncts have the form "$t_1 \underset{(\sim)}{\succ} t_2$" or "$s_1 \underset{(\sim)}{\succ} s_2 \Rightarrow t_1 \underset{(\sim)}{\succ} t_2$". However, most existing methods and tools for the generation of orders and of polynomial interpretations can only handle *unconditional* inequalities [4, 7]. To transform such conditional constraints into unconditional ones, note that any constraint "$s \succsim \mathsf{c} \Rightarrow t \succsim \mathsf{c}$" can be replaced by "$t \succsim s$". More generally, if one uses polynomial orders, then any constraint "$s_1 \,_{(}\!\succsim\!_{)}\, s_2 \Rightarrow t_1 \,_{(}\!\succsim\!_{)}\, t_2$" can be replaced by "$[t_1] - [t_2] \geq [s_1] - [s_2]$". So in Ex. 13, instead of (41) and (42), we would require $[\mathsf{F}(\mathsf{true}, \mathsf{s}(\mathsf{s}(i)), \mathsf{s}(\mathsf{s}(j)), \mathsf{0})] \geq [\mathsf{F}(\mathsf{true}, \mathsf{s}(i), \mathsf{s}(j), \mathsf{0})]$ and $[\mathsf{F}(\mathsf{true}, \mathsf{s}(u), n, \mathsf{s}(m))] \geq [\mathsf{F}(\mathsf{true}, u, n, m)]$.

In practice, it is not recommendable to fix the reduction pair $(\succsim, \succ)$ in advance and to check the validity of the constraints of the reduction pair processor afterwards. Instead, one should leave the reduction pair open and first simplify the constraints of the reduction pair processor using the above inference rules. Afterwards, one uses the existing techniques to generate a reduction pair (e.g., based on integer polynomial interpretations) satisfying the resulting constraints.

More precisely, we start the following procedure $\mathsf{REDUCTION\_PAIR}(\mathcal{P})$ with $\mathcal{P} = DP(\mathcal{R})$. If $\mathsf{REDUCTION\_PAIR}(DP(\mathcal{R}))$ returns "*Yes*", then innermost termination is proved. Of course, this procedure can be refined by also applying other DP processors than just the reduction pair processor to $\mathcal{P}$.

Procedure $\mathsf{REDUCTION\_PAIR}(\mathcal{P})$

1. If $\mathcal{P} = \varnothing$ then stop and return "*Yes*".
2. Choose non-empty subsets $\mathcal{P}_\succ \subseteq \mathcal{P}$ and $\mathcal{P}_{bound} \subseteq \mathcal{P}$. Let $\mathcal{P}_\succsim = \mathcal{P} \setminus \mathcal{P}_\succ$.
3. Generate the following constraint φ (where $\succsim$ and $\succ$ are *not yet fixed*):

$$\bigwedge_{s \to t \in \mathcal{P}_\succ,\; u \to v \in \mathcal{P}} (t = u' \;\Rightarrow\; s \succ t) \wedge \bigwedge_{s \to t \in \mathcal{P}_{bound},\; u \to v \in \mathcal{P}} (t = u' \;\Rightarrow\; s \succsim \mathsf{c}) \wedge$$
$$\bigwedge_{s \to t \in \mathcal{P}_\succsim,\; u \to v \in \mathcal{P}} (t = u' \;\Rightarrow\; s \succsim t) \wedge \bigwedge_{\ell \to r \in \mathcal{U}(\mathcal{P})} (\ell \succsim r)$$

4. Use Rules (I) - (VII) to transform φ to a constraint φ' without "=".
5. Generate an integer polynomial interpretation satisfying φ', cf. e.g. [7].
6. If $\mathsf{REDUCTION\_PAIR}(\mathcal{P}_\succsim)$ = "*Yes*" and $\mathsf{REDUCTION\_PAIR}(\mathcal{P} \setminus \mathcal{P}_{bound})$ = "*Yes*", then return "*Yes*". Otherwise, return "*Maybe*".

5 Conclusion

We have extended the reduction pair processor of the DP method in order to handle TRSs that terminate because of bounded increase. To be able to measure the *increase* of arguments, we permitted the use of general reduction pairs (e.g., based on integer polynomial interpretations). Moreover, to exploit the *bounds* given by conditions, we developed a calculus based on induction which simplifies the constraints needed for the reduction pair processor.

We implemented the new reduction pair processor of Cor. 15 in our termination prover AProVE [10]. Here, we used the heuristic from the end of Sect. 4 for the application of the inference rules (I) - (VII). To demonstrate the power of our method, the appendix contains a collection of typical TRSs with bounded increase. These include examples with non-boolean (possibly nested) functions in the bound, examples with combinations of bounds, examples containing increasing or decreasing defined symbols, examples with bounds on lists, examples with different increases in different arguments, increasing TRSs that go beyond the shape of functional programs, etc. Although AProVE was the most powerful tool for termination analysis of TRSs in the *International Competition of Termination Tools*, up to now AProVE (as well as all other tools participating in the competition) failed on all TRSs from our collection. In contrast, with the results from this paper, the new version of AProVE can prove innermost termination for all of them. Thus, these results represent a substantial advance in automated termination proving. To experiment with our implementation, the new version of AProVE can be accessed via the web at `http://aprove.informatik.rwth-aachen.de/eval/Increasing/`.

References

1. T. Arts and J. Giesl. Termination of term rewriting using dependency pairs. *Theoretical Computer Science*, 236:133–178, 2000.
2. F. Baader and T. Nipkow. *Term Rewriting and All That.* Cambridge, 1998.
3. J. Brauburger and J. Giesl. Termination analysis by inductive evaluation. In *Proc. 15th CADE*, LNAI 1421, pages 254–269, 1998.
4. E. Contejean, C. Marché, A. P. Tomás, and X. Urbain. Mechanically proving termination using polynomial interpretations. *J. Aut. Reason.*, 34(4):325–363, 2005.
5. B. Cook, A. Podelski, and A. Rybalchenko. Terminator: Beyond safety. In *Proc. CAV '06*, LNCS 4144, pages 415–418, 2006.
6. M.-L. Fernández. Relaxing monotonicity for innermost termination. *Information Processing Letters*, 93(3):117–123, 2005.
7. C. Fuhs, J. Giesl, A. Middeldorp, P. Schneider-Kamp, R. Thiemann, and H. Zankl. SAT solving for termination analysis with polynomial interpretations. In *Proc. SAT '07*, LNCS, 2007. To appear.
8. J. Giesl, R. Thiemann, and P. Schneider-Kamp. The dependency pair framework: Combining techniques for automated termination proofs. In *Proc. LPAR'04*, LNAI 3452, pages 301–331, 2005.
9. J. Giesl, R. Thiemann, P. Schneider-Kamp. Proving and disproving termination of higher-order functions. In *Proc. 5th FroCoS*, LNAI 3717, pages 216-231, 2005.
10. J. Giesl, P. Schneider-Kamp, and R. Thiemann. AProVE 1.2: Automatic termination proofs in the DP framework. *Proc. IJCAR '06*, LNAI 4130, p. 281-286, 2006.
11. J. Giesl, R. Thiemann, P. Schneider-Kamp, and S. Falke. Mechanizing and improving dependency pairs. *Journal of Automated Reasoning*, 37(3):155–203, 2006.
12. N. Hirokawa and A. Middeldorp. Automating the dependency pair method. *Information and Computation*, 199(1,2):172–199, 2005.
13. N. Hirokawa and A. Middeldorp. Tyrolean termination tool: Techniques and features. *Information and Computation*, 205(4):474–511, 2007.
14. D. Lankford. On proving term rewriting systems are Noetherian. Technical Report MTP-3, Louisiana Technical University, Ruston, LA, USA, 1979.
15. P. Manolios and D. Vroon. Termination analysis with calling context graphs. In *Proc. CAV '06*, LNCS 4144, pages 401–414, 2006.
16. C. Marché and H. Zantema. The termination competition. In *Proc. RTA '07*, 2007. To appear.

A Examples

The following examples demonstrate the power of our new method for termination analysis of increasing TRSs. Their innermost termination can be proved automatically by the new version of AProVE which is available at `http://aprove.informatik.rwth-aachen.de/eval/Increasing/`. In contrast, the tools participating in the *International Competition of Termination Tools* 2006 fail on all of these examples. This is also true for the previous version of AProVE which did not feature the results of this paper.

A.1 Simple Increasing Example with Bound

This is the first leading example from our paper (Ex. 1).

$$\begin{aligned}
\mathsf{minus}(x, y) &\to \mathsf{cond}(\mathsf{gt}(x, y), x, y)\\
\mathsf{cond}(\mathsf{false}, x, y) &\to \mathsf{0}\\
\mathsf{cond}(\mathsf{true}, x, y) &\to \mathsf{s}(\mathsf{minus}(x, \mathsf{s}(y)))\\
\mathsf{gt}(\mathsf{0}, v) &\to \mathsf{false}\\
\mathsf{gt}(\mathsf{s}(u), \mathsf{0}) &\to \mathsf{true}\\
\mathsf{gt}(\mathsf{s}(u), \mathsf{s}(v)) &\to \mathsf{gt}(u, v)
\end{aligned}$$

One can easily reduce the initial DP problem to the problem consisting just of the following two DPs.

$$\begin{aligned}
\mathsf{MINUS}(x, y) &\to \mathsf{COND}(\mathsf{gt}(x, y), x, y)\\
\mathsf{COND}(\mathsf{true}, x, y) &\to \mathsf{MINUS}(x, \mathsf{s}(y))
\end{aligned}$$

We use the polynomial interpretation $\mathcal{P}ol$ where $\mathsf{MINUS}_{\mathcal{P}ol} = x_1 - x_2$, $\mathsf{COND}_{\mathcal{P}ol} = x_2 - x_3$, $\mathsf{0}_{\mathcal{P}ol} = 0$, and $\mathsf{s}_{\mathcal{P}ol} = x_1 + 1$. Then the first DP is bounded and weakly decreasing and the second DP is strictly decreasing. Boundedness of the first DP is shown in Sect. 4.

Note that our method of course also succeeds on examples where instead of the bound $\mathsf{gt}(x, y)$ one has a *fixed* bound (e.g., $\mathsf{gt}(\mathsf{s}(\mathsf{0}), y)$).

A.2 Challenge from the Termination Competition

This example ("`TRS/Zantema06-while`") was identified as one of the main challenges for automated termination proving in [16, p. 38].

$$\begin{aligned}
\mathsf{f}(\mathsf{t}, x, y) &\to \mathsf{f}(\mathsf{g}(x, y), x, \mathsf{s}(y))\\
\mathsf{g}(\mathsf{s}(x), \mathsf{0}) &\to \mathsf{t}\\
\mathsf{g}(\mathsf{s}(x), \mathsf{s}(y)) &\to \mathsf{g}(x, y)
\end{aligned}$$

One can easily reduce the initial DP problem to $\{(45)\}$, with

$$\mathsf{F}(\mathsf{t}, x, y) \to \mathsf{F}(\mathsf{g}(x, y), x, \mathsf{s}(y)) \tag{45}$$

We use the polynomial interpretation $\mathcal{P}ol$ with $\mathsf{F}_{\mathcal{P}ol} = x_2 - x_3$, $\mathsf{0}_{\mathcal{P}ol} = \mathsf{0}$, and $\mathsf{s}_{\mathcal{P}ol} = x_1 + 1$. Then the DP is strictly decreasing. Moreover, it is bounded. We have to consider the constraint

$$\mathsf{F}(\mathsf{g}(x,y), x, \mathsf{s}(y)) = \mathsf{F}(\mathsf{t}, x', y') \;\Rightarrow\; \mathsf{F}(\mathsf{t}, x, y) \succsim \mathsf{c}$$

This can be done as in the previous example.

A.3 Bound which is a Constructor Term

The following example has the bound $\mathsf{ge}(x, \mathsf{s}(y))$. So here, one argument of ge is a constructor term $\mathsf{s}(y)$.

$$\begin{aligned}
\mathsf{minus}(x,y) &\to \mathsf{cond}(\mathsf{ge}(x,\mathsf{s}(y)), x, y)\\
\mathsf{cond}(\mathsf{false}, x, y) &\to \mathsf{0}\\
\mathsf{cond}(\mathsf{true}, x, y) &\to \mathsf{s}(\mathsf{minus}(x, \mathsf{s}(y)))\\
\mathsf{ge}(u, \mathsf{0}) &\to \mathsf{true}\\
\mathsf{ge}(\mathsf{0}, \mathsf{s}(v)) &\to \mathsf{false}\\
\mathsf{ge}(\mathsf{s}(u), \mathsf{s}(v)) &\to \mathsf{ge}(u, v)
\end{aligned}$$

One can easily reduce the initial DP problem to $\{(46), (47)\}$, with

$$\mathsf{MINUS}(x,y) \to \mathsf{COND}(\mathsf{ge}(x,\mathsf{s}(y)), x, y) \tag{46}$$

$$\mathsf{COND}(\mathsf{true}, x, y) \to \mathsf{MINUS}(x, \mathsf{s}(y)) \tag{47}$$

We use the polynomial interpretation $\mathcal{P}ol$ with $\mathsf{MINUS}_{\mathcal{P}ol} = x_1 - x_2$, $\mathsf{COND}_{\mathcal{P}ol} = x_2 - x_3$, $\mathsf{0}_{\mathcal{P}ol} = \mathsf{0}$, and $\mathsf{s}_{\mathcal{P}ol} = x_1 + 1$. Then (47) is strictly decreasing. Moreover, we show that (46) is bounded (afterwards the remaining DP problems $\{(46)\}$ and $\{(47)\}$ are easy to solve). We have to consider the constraint

$$\mathsf{COND}(\mathsf{ge}(x, \mathsf{s}(y)), x, y) = \mathsf{COND}(\mathsf{true}, x', y') \;\Rightarrow\; \mathsf{MINUS}(x, y) \succsim \mathsf{c}$$

By Rule (II) and (IV) we obtain

$$\mathsf{ge}(x, \mathsf{s}(y)) = \mathsf{true} \;\Rightarrow\; \mathsf{MINUS}(x, y) \succsim \mathsf{c}$$

Rule (VII) yields

$$\mathsf{s}(y) = z \wedge \mathsf{ge}(x, z) = \mathsf{true} \;\Rightarrow\; \mathsf{MINUS}(x, y) \succsim \mathsf{c}$$

Now we perform induction with Rule (V). If we omit the unsatisfiable constraints (which are deleted by Rule (I)), then we only result in

$$\begin{aligned}
&\mathsf{s}(y) = \mathsf{s}(v) \;\wedge\; \mathsf{ge}(u, v) = \mathsf{true}\\
\wedge\; &(\,\forall y\;\; \mathsf{s}(y) = v \wedge \mathsf{ge}(u, v) = \mathsf{true} \;\Rightarrow\; \mathsf{MINUS}(u, y) \succsim \mathsf{c}\,) \;\;\Rightarrow\; \mathsf{MINUS}(\mathsf{s}(u), y) \succsim \mathsf{c}
\end{aligned}$$

By Rule (II) we can replace the condition $\mathsf{s}(y) = \mathsf{s}(v)$ by $y = v$ and then substitute y by v using Rule (III). Moreover, we remove the induction hypothesis with Rule (IV). This yields

$$\mathsf{ge}(u, v) = \mathsf{true} \;\Rightarrow\; \mathsf{MINUS}(\mathsf{s}(u), v) \succsim \mathsf{c}$$

Now we perform induction with Rule (V) again. By Rule (I), one resulting constraint is omitted and thus, we result in the constraint

$$\mathsf{true} = \mathsf{true} \;\Rightarrow\; \mathsf{MINUS}(\mathsf{s}(n), \mathsf{0}) \succsim \mathsf{c}$$

and in

$$\begin{array}{ll} \;\;\mathsf{ge}(n, m) = \mathsf{true} & \\ \wedge\; (\,\mathsf{ge}(n, m) = \mathsf{true} \;\Rightarrow\; \mathsf{MINUS}(\mathsf{s}(n), m) \succsim \mathsf{c}\,) & \Rightarrow\; \mathsf{MINUS}(\mathsf{s}(\mathsf{s}(n)), \mathsf{s}(m)) \succsim \mathsf{c} \end{array}$$

By Rule (II), (VI), and (IV), we finally obtain

$$\begin{aligned} \mathsf{MINUS}(\mathsf{s}(n), \mathsf{0}) &\succsim \mathsf{c} \\ \mathsf{MINUS}(\mathsf{s}(n), m) \succsim \mathsf{c} \Rightarrow \mathsf{MINUS}(\mathsf{s}(\mathsf{s}(n)), \mathsf{s}(m)) &\succsim \mathsf{c} \end{aligned}$$

If we have $\mathsf{c}_{\mathcal{P}ol} = 1$, then these constraints are obviously satisfied.

A.4 Bound with Non-Boolean Function

The following example uses the function min in its bound which computes the minimum of two numbers. Note that this TRS is neither confluent, nor sufficiently complete, nor left-linear (i.e., it does not have the shape of functional programs).

$$\begin{aligned} \mathsf{minus}(x, x) &\to \mathsf{0} \\ \mathsf{minus}(x, y) &\to \mathsf{cond}(\mathsf{min}(x, y), x, y) \\ \mathsf{cond}(y, x, y) &\to \mathsf{s}(\mathsf{minus}(x, \mathsf{s}(y))) \\ \mathsf{min}(\mathsf{0}, v) &\to \mathsf{0} \\ \mathsf{min}(u, \mathsf{0}) &\to \mathsf{0} \\ \mathsf{min}(\mathsf{s}(u), \mathsf{s}(v)) &\to \mathsf{s}(\mathsf{min}(u, v)) \end{aligned}$$

One can easily reduce the initial DP problem to $\{(48), (49)\}$, with

$$\mathsf{MINUS}(x, y) \to \mathsf{COND}(\mathsf{min}(x, y), x, y) \tag{48}$$

$$\mathsf{COND}(y, x, y) \to \mathsf{MINUS}(x, \mathsf{s}(y)) \tag{49}$$

We use the polynomial interpretation $\mathcal{P}ol$ with $\mathsf{MINUS}_{\mathcal{P}ol} = x_1 - x_2$, $\mathsf{COND}_{\mathcal{P}ol} = x_2 - x_3$, $\mathsf{0}_{\mathcal{P}ol} = \mathsf{0}$, and $\mathsf{s}_{\mathcal{P}ol} = x_1 + 1$. Then (49) is strictly decreasing. Moreover, we show that (48) is bounded (afterwards the remaining DP problems $\{(48)\}$ and $\{(49)\}$ are easy to solve). We have to consider the constraint

$$\mathsf{COND}(\mathsf{min}(x, y), x, y) = \mathsf{COND}(y', x', y') \;\Rightarrow\; \mathsf{MINUS}(x, y) \succsim \mathsf{c}$$

By Rule (II), (III), and (IV) we obtain

$$\mathsf{min}(x, y') = y' \;\Rightarrow\; \mathsf{MINUS}(x, y') \gtrsim \mathsf{c}$$

Now we perform induction with Rule (V) and result in

$$\mathsf{0} = v \;\Rightarrow\; \mathsf{MINUS}(\mathsf{0}, v) \gtrsim \mathsf{c}$$
$$\mathsf{0} = \mathsf{0} \;\Rightarrow\; \mathsf{MINUS}(u, \mathsf{0}) \gtrsim \mathsf{c}$$

and the formula

$$\begin{aligned}&\mathsf{s}(\mathsf{min}(u, v)) = \mathsf{s}(v)\\ \wedge\ &(\mathsf{min}(u, v) = v \Rightarrow \mathsf{MINUS}(u, v) \gtrsim \mathsf{c}) \qquad \Rightarrow\; \mathsf{MINUS}(\mathsf{s}(u), \mathsf{s}(v)) \gtrsim \mathsf{c}\end{aligned}$$

The first constraint is simplified with Rule (III), the second constraint is simplified with Rule (II), and in the third one we first remove the outermost s from the first condition with Rule (II). Then we simplify the condition with Rule (VI) and afterwards remove conditions with Rule (IV). So we finally result in

$$\begin{aligned}\mathsf{MINUS}(\mathsf{0}, \mathsf{0}) &\gtrsim \mathsf{c}\\ \mathsf{MINUS}(u, \mathsf{0}) &\gtrsim \mathsf{c}\\ \mathsf{MINUS}(u, v) \gtrsim \mathsf{c} \Rightarrow \mathsf{MINUS}(\mathsf{s}(u), \mathsf{s}(v)) &\gtrsim \mathsf{c}\end{aligned}$$

If we have $\mathsf{c}_{\mathcal{P}ol} = 0$, then these constraints are obviously satisfied.

A.5 Bound with Nested Non-Boolean Function

The following example is similar to the previous one, but uses nested function symbols in the bound.

$$\begin{aligned}\mathsf{minus}(x, x) &\to \mathsf{0}\\ \mathsf{minus}(x, y) &\to \mathsf{cond}(\mathsf{equal}(\mathsf{min}(x, y), y), x, y)\\ \mathsf{cond}(\mathsf{true}, x, y) &\to \mathsf{s}(\mathsf{minus}(x, \mathsf{s}(y)))\\ \mathsf{min}(\mathsf{0}, v) &\to \mathsf{0}\\ \mathsf{min}(u, \mathsf{0}) &\to \mathsf{0}\\ \mathsf{min}(\mathsf{s}(u), \mathsf{s}(v)) &\to \mathsf{s}(\mathsf{min}(u, v))\\ \mathsf{equal}(\mathsf{0}, \mathsf{0}) &\to \mathsf{true}\\ \mathsf{equal}(\mathsf{s}(x), \mathsf{0}) &\to \mathsf{false}\\ \mathsf{equal}(\mathsf{0}, \mathsf{s}(y)) &\to \mathsf{false}\\ \mathsf{equal}(\mathsf{s}(x), \mathsf{s}(y)) &\to \mathsf{equal}(x, y)\end{aligned}$$

One can easily reduce the initial DP problem to $\{(50), (51)\}$, with

$$\mathsf{MINUS}(x, y) \to \mathsf{COND}(\mathsf{equal}(\mathsf{min}(x, y), y), x, y) \tag{50}$$
$$\mathsf{COND}(\mathsf{true}, x, y) \to \mathsf{MINUS}(x, \mathsf{s}(y)) \tag{51}$$

We use the polynomial interpretation $\mathcal{P}ol$ with $\mathsf{MINUS}_{\mathcal{P}ol} = x_1 - x_2$, $\mathsf{COND}_{\mathcal{P}ol} = x_2 - x_3$, $\mathsf{0}_{\mathcal{P}ol} = 0$, and $\mathsf{s}_{\mathcal{P}ol} = x_1 + 1$. Then (51) is strictly decreasing. Moreover, we show that (50) is bounded (afterwards the remaining DP problems $\{(50)\}$ and $\{(51)\}$ are easy to solve). We have to consider the constraint

$$\mathsf{COND}(\mathsf{equal}(\mathsf{min}(x,y),y),x,y) = \mathsf{COND}(\mathsf{true},x',y') \;\Rightarrow\; \mathsf{MINUS}(x,y) \gtrsim \mathsf{c}$$

By Rules (II) and (IV) we obtain

$$\mathsf{equal}(\mathsf{min}(x,y),y) = \mathsf{true} \;\Rightarrow\; \mathsf{MINUS}(x,y) \gtrsim \mathsf{c}$$

Now Rule (VII) yields

$$\mathsf{min}(x,y) = z \wedge \mathsf{equal}(z,y) = \mathsf{true} \;\Rightarrow\; \mathsf{MINUS}(x,y) \gtrsim \mathsf{c}$$

We use induction on equal. After removing the constraints with the unsatisfiable condition $\mathsf{false} = \mathsf{true}$ we obtain

$$\mathsf{min}(x,\mathsf{0}) = \mathsf{0} \wedge \mathsf{true} = \mathsf{true} \Rightarrow \mathsf{MINUS}(x,\mathsf{0}) \gtrsim \mathsf{c} \tag{52}$$
$$\mathsf{min}(x,\mathsf{s}(v)) = \mathsf{s}(u) \;\wedge\; \mathsf{equal}(u,v) = \mathsf{true} \;\wedge\; (54) \Rightarrow \mathsf{MINUS}(x,\mathsf{s}(v)) \gtrsim \mathsf{c} \tag{53}$$

where

$$\forall x\; \mathsf{min}(x,v) = u \wedge \mathsf{equal}(u,v) = \mathsf{true} \;\Rightarrow\; \mathsf{MINUS}(x,v) \gtrsim \mathsf{c} \tag{54}$$

is the induction hypothesis.

We first continue with (52) where we remove the condition by Rule (IV) and obtain the final constraint

$$\mathsf{MINUS}(x,\mathsf{0}) \gtrsim \mathsf{c} \tag{55}$$

For (53), we apply Rule (VII) to get

$$\mathsf{s}(v) = w \;\wedge\; \mathsf{min}(x,w) = \mathsf{s}(u) \;\wedge\; \mathsf{equal}(u,v) = \mathsf{true} \;\wedge\; (54) \Rightarrow\; \mathsf{MINUS}(x,\mathsf{s}(v)) \gtrsim \mathsf{c}$$

and process this constraint further by induction on min by Rule (V). After removing constraints with unsatisfiable premises by Rule (I), the following constraint remains where φ' is the new induction hypothesis.

$$\begin{aligned}\mathsf{s}(v) = \mathsf{s}(m) \;\wedge\; \mathsf{s}(\mathsf{min}(n,m)) = \mathsf{s}(u) \;\wedge\; \mathsf{equal}(u,v) = \mathsf{true} \;\wedge\; (54) \;\wedge\; \varphi' \\ \Rightarrow\; \mathsf{MINUS}(\mathsf{s}(n),\mathsf{s}(v)) \gtrsim \mathsf{c}\end{aligned} \tag{56}$$

We can now use Rules (II), (III), and (VI). When applying the induction hypothesis (54), we instantiate x by n. We obtain

$$\begin{aligned}\mathsf{min}(n,m) = u \;\wedge\; \mathsf{equal}(u,m) = \mathsf{true} \;\wedge\; \mathsf{MINUS}(n,m) \gtrsim \mathsf{c} \;\wedge\; \varphi' \\ \Rightarrow\; \mathsf{MINUS}(\mathsf{s}(n),\mathsf{s}(m)) \gtrsim \mathsf{c}\end{aligned}$$

After removing all conditions containing "=" with Rule (IV), we therefore obtain

$$\mathsf{MINUS}(n,m) \gtrsim \mathsf{c} \;\Rightarrow\; \mathsf{MINUS}(\mathsf{s}(n),\mathsf{s}(m)) \gtrsim \mathsf{c}. \tag{57}$$

If we have $\mathsf{c}_{\mathcal{P}ol} = 0$, then the constraints (55) and (57) are obviously satisfied.

A.6 Bound with Nested Defined Symbols

The second leading example from our paper (Ex. 12) has nested defined symbols in its bound. While our induction inference rule was inspired by our earlier work in [3], the approach of [3] would not succeed on such examples, since it would only perform induction on plus, but not on gt. For similar reasons, the approach of [3] fails on most of the examples from this collection. (Moreover, several of these examples do not have the shape of functional programs and thus, [3] is not applicable at all.)

$$\begin{aligned}
\mathsf{f}(\mathsf{true}, x, y, z) &\to \mathsf{f}(\mathsf{gt}(x, \mathsf{plus}(y, z)), x, \mathsf{s}(y), z)\\
\mathsf{f}(\mathsf{true}, x, y, z) &\to \mathsf{f}(\mathsf{gt}(x, \mathsf{plus}(y, z)), x, y, \mathsf{s}(z))\\
\mathsf{plus}(n, \mathsf{0}) &\to n\\
\mathsf{plus}(n, \mathsf{s}(m)) &\to \mathsf{s}(\mathsf{plus}(n, m))\\
\mathsf{gt}(\mathsf{0}, v) &\to \mathsf{false}\\
\mathsf{gt}(\mathsf{s}(u), \mathsf{0}) &\to \mathsf{true}\\
\mathsf{gt}(\mathsf{s}(u), \mathsf{s}(v)) &\to \mathsf{gt}(u, v)
\end{aligned}$$

The initial DP problem can easily be reduced to

$$\begin{aligned}
\mathsf{F}(\mathsf{true}, x, y, z) &\to \mathsf{F}(\mathsf{gt}(x, \mathsf{plus}(y, z)), x, \mathsf{s}(y), z)\\
\mathsf{F}(\mathsf{true}, x, y, z) &\to \mathsf{F}(\mathsf{gt}(x, \mathsf{plus}(y, z)), x, y, \mathsf{s}(z))
\end{aligned}$$

We used the interpretation $\mathcal{P}ol$ where $\mathsf{F}_{\mathcal{P}ol} = x_2 - x_3 - x_4$, $\mathsf{s}_{\mathcal{P}ol} = x_1 + 1$, and $\mathsf{0}_{\mathcal{P}ol} = 0$. Then both DPs are strictly decreasing. To show that they are bounded, we proceed as in Sect. 4.

A.7 Bound with Two Conditions

The next example is a variant of Ex. 12. Instead of using plus, now we use two conditions with gt.

$$\begin{aligned}
\mathsf{f}(\mathsf{true}, x, y, z) &\to \mathsf{g}(\mathsf{gt}(x, y), x, y, z)\\
\mathsf{g}(\mathsf{true}, x, y, z) &\to \mathsf{f}(\mathsf{gt}(x, z), x, \mathsf{s}(y), z)\\
\mathsf{g}(\mathsf{true}, x, y, z) &\to \mathsf{f}(\mathsf{gt}(x, z), x, y, \mathsf{s}(z))\\
\mathsf{gt}(\mathsf{0}, v) &\to \mathsf{false}\\
\mathsf{gt}(\mathsf{s}(u), \mathsf{0}) &\to \mathsf{true}\\
\mathsf{gt}(\mathsf{s}(u), \mathsf{s}(v)) &\to \mathsf{gt}(u, v)
\end{aligned}$$

One can easily reduce the initial DP problem to $\{(58), (59), (60)\}$, with

$$\begin{aligned}
\mathsf{F}(\mathsf{true}, x, y, z) &\to \mathsf{G}(\mathsf{gt}(x, y), x, y, z) && (58)\\
\mathsf{G}(\mathsf{true}, x, y, z) &\to \mathsf{F}(\mathsf{gt}(x, z), x, \mathsf{s}(y), z) && (59)\\
\mathsf{G}(\mathsf{true}, x, y, z) &\to \mathsf{F}(\mathsf{gt}(x, z), x, y, \mathsf{s}(z)) && (60)
\end{aligned}$$

We use the polynomial interpretation $\mathcal{P}ol$ with $\mathsf{F}_{\mathcal{P}ol} = \mathsf{G}_{\mathcal{P}ol} = 2 \cdot x_2 - x_3 - x_4$, $\mathsf{0}_{\mathcal{P}ol} = 0$, and $\mathsf{s}_{\mathcal{P}ol} = x_1 + 1$. Then (59) and (60) are strictly decreasing. Moreover, we show that they are also both bounded (afterwards the remaining DP problem $\{(58)\}$ is easy to solve). Here, we need the refinement of Thm. 11 which considers more than two adjacent pairs in innermost chains. We only show the boundedness of (60) (the proof for (59) is analogous). We have to consider the constraint

$$\begin{array}{ll} & \mathsf{F}(\mathsf{gt}(x,z), x, y, \mathsf{s}(z)) = \mathsf{F}(\mathsf{true}, x', y', z') \\ \wedge & \mathsf{G}(\mathsf{gt}(x',y'), x', y', z') = \mathsf{G}(\mathsf{true}, x'', y'', z'') \qquad \Rightarrow \; \mathsf{G}(\mathsf{true}, x, y, z) \succsim \mathsf{c}. \end{array}$$

After simplification with Rules (II), (III), and (IV) we obtain

$$\mathsf{gt}(x', z) = \mathsf{true} \wedge \mathsf{gt}(x', y') = \mathsf{true} \;\Rightarrow\; \mathsf{G}(\mathsf{true}, x', y', z) \succsim \mathsf{c}$$

We use Rule (V) to perform induction on $\mathsf{gt}(x', z)$. After simplification with Rule (II) and (I) we obtain

$$\mathsf{gt}(\mathsf{s}(u), y') = \mathsf{true} \Rightarrow \mathsf{G}(\mathsf{true}, \mathsf{s}(u), y', \mathsf{0}) \succsim \mathsf{c} \tag{61}$$

$$\mathsf{gt}(u, v) = \mathsf{true} \wedge \mathsf{gt}(\mathsf{s}(u), y') = \mathsf{true} \wedge \varphi' \Rightarrow \mathsf{G}(\mathsf{true}, \mathsf{s}(u), y', \mathsf{s}(v)) \succsim \mathsf{c} \tag{62}$$

where

$$\varphi' : \qquad \forall y' \;\; \mathsf{gt}(u, v) = \mathsf{true} \wedge \mathsf{gt}(u, y') = \mathsf{true} \;\Rightarrow\; \mathsf{G}(\mathsf{true}, u, y', v) \succsim \mathsf{c}.$$

To transform (61) further, we first apply Rule (VII) to replace $\mathsf{gt}(\mathsf{s}(u), y') = \mathsf{true}$ by $\mathsf{s}(u) = w \wedge \mathsf{gt}(w, y') = \mathsf{true}$. Then we perform induction on $\mathsf{gt}(w, y')$. After simplification with Rule (II), (I), and (III) we obtain

$$\mathsf{G}(\mathsf{true}, \mathsf{s}(n), \mathsf{0}, \mathsf{0}) \succsim \mathsf{c} \tag{63}$$

$$\mathsf{gt}(n, m) = \mathsf{true} \wedge \varphi'' \Rightarrow \mathsf{G}(\mathsf{true}, \mathsf{s}(n), \mathsf{s}(m), \mathsf{0}) \succsim \mathsf{c} \tag{64}$$

for a new induction hypothesis φ''. For (64) we now perform another induction on $\mathsf{gt}(n, m)$ and according to our heuristic, we first delete the condition φ'' with Rule (IV). Then in the end, we result in

$$\mathsf{G}(\mathsf{true}, \mathsf{s}(\mathsf{s}(i)), \mathsf{s}(\mathsf{0}), \mathsf{0}) \succsim \mathsf{c} \tag{65}$$

$$\mathsf{G}(\mathsf{true}, \mathsf{s}(i), \mathsf{s}(j), \mathsf{0}) \succsim \mathsf{c} \Rightarrow \mathsf{G}(\mathsf{true}, \mathsf{s}(\mathsf{s}(i)), \mathsf{s}(\mathsf{s}(j)), \mathsf{0}) \succsim \mathsf{c} \tag{66}$$

Now we transform (62) further. We again apply Rule (VII) to replace $\mathsf{gt}(\mathsf{s}(u), y') = \mathsf{true}$ by $\mathsf{s}(u) = w \wedge \mathsf{gt}(w, y') = \mathsf{true}$. Then we perform induction on $\mathsf{gt}(w, y')$. In the case where w is instantiated with $\mathsf{s}(n)$ and y' is instantiated with $\mathsf{0}$, we obtain

$$\begin{array}{ll} & \mathsf{gt}(n, v) = \mathsf{true} \\ \wedge & (\forall y' \;\; \mathsf{gt}(n, v) = \mathsf{true} \wedge \mathsf{gt}(n, y') = \mathsf{true} \;\Rightarrow\; \mathsf{G}(\mathsf{true}, n, y', v) \succsim \mathsf{c}) \\ & \qquad\qquad \Rightarrow \; \mathsf{G}(\mathsf{true}, \mathsf{s}(n), \mathsf{0}, \mathsf{s}(v)) \succsim \mathsf{c}. \end{array}$$

Now we finally apply Rule (VI) by instantiating y' with v. Then we end up with

$$\mathsf{G}(\mathsf{true}, n, v, \mathsf{s}(v)) \succsim \mathsf{c} \;\Rightarrow\; \mathsf{G}(\mathsf{true}, \mathsf{s}(n), \mathsf{0}, \mathsf{s}(v)) \succsim \mathsf{c} \tag{67}$$

In the case where w is instantiated with $\mathsf{s}(n)$ and y' is instantiated with $\mathsf{s}(m)$, we obtain

$$\mathsf{gt}(n, v) = \mathsf{true} \wedge \mathsf{gt}(n, m) = \mathsf{true} \wedge \varphi'[u/n] \wedge \varphi'' \;\Rightarrow\; \mathsf{G}(\mathsf{true}, \mathsf{s}(n), \mathsf{s}(m), \mathsf{s}(v)) \succsim \mathsf{c}$$

for a new induction hypothesis φ''. Now we apply the first induction hypothesis $\varphi'[u/n]$ by Rule (VI) where we instantiate the universally quantified variable y' with m. We delete the remaining conditions and end up with

$$\mathsf{G}(\mathsf{true}, n, m, v) \succsim \mathsf{c} \;\Rightarrow\; \mathsf{G}(\mathsf{true}, \mathsf{s}(n), \mathsf{s}(m), \mathsf{s}(v)) \succsim \mathsf{c} \tag{68}$$

If we have $\mathsf{c}_{\mathcal{P}ol} = 0$, then the resulting constraints (63), (65), (66), (67), and (68) are obviously satisfied.

A.8 Boolean Combination in Condition

The next example is similar to the previous one, but uses a conjunction in the condition.

$$\begin{aligned}
\mathsf{f}(\mathsf{true}, x, y, z) &\to \mathsf{f}(\mathsf{and}(\mathsf{gt}(x, y), \mathsf{gt}(x, z)), x, \mathsf{s}(y), z) \\
\mathsf{f}(\mathsf{true}, x, y, z) &\to \mathsf{f}(\mathsf{and}(\mathsf{gt}(x, y), \mathsf{gt}(x, z)), x, y, \mathsf{s}(z)) \\
\mathsf{gt}(\mathsf{0}, v) &\to \mathsf{false} \\
\mathsf{gt}(\mathsf{s}(u), \mathsf{0}) &\to \mathsf{true} \\
\mathsf{gt}(\mathsf{s}(u), \mathsf{s}(v)) &\to \mathsf{gt}(u, v) \\
\mathsf{and}(x, \mathsf{true}) &\to x \\
\mathsf{and}(x, \mathsf{false}) &\to \mathsf{false}
\end{aligned}$$

One can easily reduce the initial DP problem to $\{(69), (70)\}$, with

$$\mathsf{F}(\mathsf{true}, x, y, z) \to \mathsf{F}(\mathsf{and}(\mathsf{gt}(x, y), \mathsf{gt}(x, z)), x, \mathsf{s}(y), z) \tag{69}$$
$$\mathsf{F}(\mathsf{true}, x, y, z) \to \mathsf{F}(\mathsf{and}(\mathsf{gt}(x, y), \mathsf{gt}(x, z)), x, y, \mathsf{s}(z)) \tag{70}$$

We use the polynomial interpretation $\mathcal{P}ol$ with $\mathsf{F}_{\mathcal{P}ol} = 2 \cdot x_2 - x_3 - x_4$, $\mathsf{0}_{\mathcal{P}ol} = \mathsf{0}$, and $\mathsf{s}_{\mathcal{P}ol} = x_1 + 1$. Then (69) and (70) are strictly decreasing. Moreover, we show that they are also both bounded. We only show the boundedness of (70) (the proof for (69) is analogous). We have to consider the constraint

$$\mathsf{F}(\mathsf{and}(\mathsf{gt}(x, y), \mathsf{gt}(x, z)), x, y, \mathsf{s}(z)) = \mathsf{F}(\mathsf{true}, x', y', z') \;\Rightarrow\; \mathsf{F}(\mathsf{true}, x, y, z) \succsim \mathsf{c}.$$

After simplification with Rule (II) and (IV) we obtain

$$\mathsf{and}(\mathsf{gt}(x, y), \mathsf{gt}(x, z)) = \mathsf{true} \;\Rightarrow\; \mathsf{F}(\mathsf{true}, x, y, z) \succsim \mathsf{c}.$$

Rule (VII) yields

$$\mathsf{gt}(x,y) = u \wedge \mathsf{gt}(x,z) = v \wedge \mathsf{and}(u,v) = \mathsf{true} \;\Rightarrow\; \mathsf{F}(\mathsf{true},x,y,z) \succsim \mathsf{c}.$$

Now induction on and ends up with

$$\mathsf{gt}(x,y) = \mathsf{true} \wedge \mathsf{gt}(x,z) = \mathsf{true} \;\Rightarrow\; \mathsf{F}(\mathsf{true},x,y,z) \succsim \mathsf{c}.$$

The remainder of the proof is similar to the previous example.

A.9 Increasing Defined Symbols

The next example uses the defined symbol round from Ex. 9 to increase an argument.

$$\begin{aligned}
\mathsf{f}(\mathsf{true},x,y) &\to \mathsf{f}(\mathsf{gt}(x,y),x,\mathsf{round}(\mathsf{s}(y)))\\
\mathsf{round}(\mathsf{0}) &\to \mathsf{0}\\
\mathsf{round}(\mathsf{s}(\mathsf{0})) &\to \mathsf{s}(\mathsf{s}(\mathsf{0}))\\
\mathsf{round}(\mathsf{s}(\mathsf{s}(x))) &\to \mathsf{s}(\mathsf{s}(\mathsf{round}(x)))\\
\mathsf{gt}(\mathsf{0},v) &\to \mathsf{false}\\
\mathsf{gt}(\mathsf{s}(u),\mathsf{0}) &\to \mathsf{true}\\
\mathsf{gt}(\mathsf{s}(u),\mathsf{s}(v)) &\to \mathsf{gt}(u,v)
\end{aligned}$$

One can easily reduce the initial DP problem to $\{(71)\}$, with

$$\mathsf{F}(\mathsf{true},x,y) \;\to\; \mathsf{F}(\mathsf{gt}(x,y),x,\mathsf{round}(\mathsf{s}(y))) \tag{71}$$

We use the polynomial interpretation $\mathcal{P}ol$ with $\mathsf{F}_{\mathcal{P}ol} = x_2 - x_3$, $\mathsf{0}_{\mathcal{P}ol} = \mathsf{0}$, $\mathsf{s}_{\mathcal{P}ol} = x_1 + 1$, and $\mathsf{round}_{\mathcal{P}ol} = x_1$. The usable rules are the reversed round-rules, since F is $\succsim$-monotonically decreasing on 3. The DP is clearly strictly decreasing. To show that it is bounded, we have to consider the constraint

$$\mathsf{F}(\mathsf{gt}(x,y),x,\mathsf{round}(\mathsf{s}(y))) = \mathsf{F}(\mathsf{true},x',y') \;\Rightarrow\; \mathsf{F}(\mathsf{true},x,y) \succsim \mathsf{c}.$$

By Rule (II) we get

$$\mathsf{gt}(x,y) = \mathsf{true} \wedge x = x' \wedge \mathsf{round}(\mathsf{s}(y)) = y' \;\Rightarrow\; \mathsf{F}(\mathsf{true},x,y) \succsim \mathsf{c}.$$

Now we delete the conditions $x = x'$ and $\mathsf{round}(\mathsf{s}(y)) = y'$ by Rule (IV) since x' and y' do not occur anywhere else. We obtain

$$\mathsf{gt}(x,y) = \mathsf{true} \;\Rightarrow\; \mathsf{F}(\mathsf{true},x,y) \succsim \mathsf{c}.$$

Now the remainder of the proof works similar to our leading example (Ex. 1).

A.10 Decreasing Defined Symbols

The next example uses the defined symbol trunc to (possibly) decrease an argument. Here, $\mathsf{trunc}(x)$ is the highest even number that is less than or equal to x.

$$\begin{aligned}
\mathsf{f}(\mathsf{true}, x, y) &\to \mathsf{f}(\mathsf{gt}(x, y), \mathsf{trunc}(x), \mathsf{s}(y))\\
\mathsf{trunc}(\mathsf{0}) &\to \mathsf{0}\\
\mathsf{trunc}(\mathsf{s}(\mathsf{0})) &\to \mathsf{0}\\
\mathsf{trunc}(\mathsf{s}(\mathsf{s}(x))) &\to \mathsf{s}(\mathsf{s}(\mathsf{trunc}(x)))\\
\mathsf{gt}(\mathsf{0}, v) &\to \mathsf{false}\\
\mathsf{gt}(\mathsf{s}(u), \mathsf{0}) &\to \mathsf{true}\\
\mathsf{gt}(\mathsf{s}(u), \mathsf{s}(v)) &\to \mathsf{gt}(u, v)
\end{aligned}$$

One can easily reduce the initial DP problem to $\{(72)\}$, with

$$\mathsf{F}(\mathsf{true}, x, y) \;\to\; \mathsf{F}(\mathsf{gt}(x, y), \mathsf{trunc}(x), \mathsf{s}(y)) \tag{72}$$

We use the polynomial interpretation $\mathcal{P}ol$ with $\mathsf{F}_{\mathcal{P}ol} = x_2 - x_3$, $\mathsf{0}_{\mathcal{P}ol} = \mathsf{0}$, $\mathsf{s}_{\mathcal{P}ol} = x_1 + 1$, and $\mathsf{trunc}_{\mathcal{P}ol} = x_1$. The usable rules are the trunc-rules, since F is $\succsim$-monotonically increasing on 2. The DP is clearly strictly decreasing. To show that it is bounded, we have to consider the constraint

$$\mathsf{F}(\mathsf{gt}(x, y), \mathsf{trunc}(x), \mathsf{s}(y)) = \mathsf{F}(\mathsf{true}, x', y') \;\Rightarrow\; \mathsf{F}(\mathsf{true}, x, y) \succsim \mathsf{c}.$$

By Rule (II) we get

$$\mathsf{gt}(x, y) = \mathsf{true} \wedge \mathsf{trunc}(x) = x' \wedge \mathsf{s}(y) = y' \;\Rightarrow\; \mathsf{F}(\mathsf{true}, x, y) \succsim \mathsf{c}.$$

Now we delete the conditions $\mathsf{trunc}(x) = x'$ and $\mathsf{s}(y) = y'$ by Rule (IV) since x' and y' do not occur anywhere else. We obtain

$$\mathsf{gt}(x, y) = \mathsf{true} \;\Rightarrow\; \mathsf{F}(\mathsf{true}, x, y) \succsim \mathsf{c}.$$

Now the remainder of the proof works similar to our leading example (Ex. 1).

A.11 Increase in all Arguments

The next example increases the third argument more than the second argument.

$$\begin{aligned}
\mathsf{f}(\mathsf{true}, x, y) &\to \mathsf{f}(\mathsf{gt}(x, y), \mathsf{s}(x), \mathsf{s}(\mathsf{s}(y)))\\
\mathsf{gt}(\mathsf{0}, v) &\to \mathsf{false}\\
\mathsf{gt}(\mathsf{s}(u), \mathsf{0}) &\to \mathsf{true}\\
\mathsf{gt}(\mathsf{s}(u), \mathsf{s}(v)) &\to \mathsf{gt}(u, v)
\end{aligned}$$

One can easily reduce the initial DP problem to $\{(73)\}$, with

$$\mathsf{F}(\mathsf{true}, x, y) \;\to\; \mathsf{F}(\mathsf{gt}(x, y), \mathsf{s}(x), \mathsf{s}(\mathsf{s}(y))) \tag{73}$$

We use the polynomial interpretation $\mathcal{P}ol$ with $\mathsf{F}_{\mathcal{P}ol} = x_2 - x_3$, $\mathsf{0}_{\mathcal{P}ol} = \mathsf{0}$, and $\mathsf{s}_{\mathcal{P}ol} = x_1 + 1$. The DP is clearly strictly decreasing. To show that it is bounded, we have to consider the constraint

$$\mathsf{F}(\mathsf{gt}(x,y), \mathsf{s}(x), \mathsf{s}(\mathsf{s}(y))) = \mathsf{F}(\mathsf{true}, x', y') \;\Rightarrow\; \mathsf{F}(\mathsf{true}, x, y) \succsim \mathsf{c}$$

which can be reduced to

$$\mathsf{gt}(x,y) = \mathsf{true} \;\Rightarrow\; \mathsf{F}(\mathsf{true}, x, y) \succsim \mathsf{c}$$

as in the previous examples. Now the remainder of the proof works similar to our leading example (Ex. 1).

A.12 Increase in all Arguments with Defined Symbol

Similarly to the previous example, this TRS again increases the third argument more than the second. But the example is more involved, since it uses defined symbols for this increase. Note also that due to the associativity rule of plus, this TRS is no constructor system (i.e., it does not have the shape of functional programs).

$$\begin{aligned}
\mathsf{f}(\mathsf{true}, x, y) &\to \mathsf{f}(\mathsf{and}(\mathsf{gt}(x,y), \mathsf{gt}(y, \mathsf{s}(\mathsf{s}(\mathsf{0})))), \mathsf{plus}(\mathsf{s}(\mathsf{0}), x), \mathsf{double}(y)) \\
\mathsf{gt}(\mathsf{0}, v) &\to \mathsf{false} \\
\mathsf{gt}(\mathsf{s}(u), \mathsf{0}) &\to \mathsf{true} \\
\mathsf{gt}(\mathsf{s}(u), \mathsf{s}(v)) &\to \mathsf{gt}(u, v) \\
\mathsf{and}(x, \mathsf{true}) &\to x \\
\mathsf{and}(x, \mathsf{false}) &\to \mathsf{false} \\
\mathsf{plus}(n, \mathsf{0}) &\to n \\
\mathsf{plus}(n, \mathsf{s}(m)) &\to \mathsf{s}(\mathsf{plus}(n, m)) \\
\mathsf{plus}(\mathsf{plus}(n, m), u) &\to \mathsf{plus}(n, \mathsf{plus}(m, u)) \\
\mathsf{double}(\mathsf{0}) &\to \mathsf{0} \\
\mathsf{double}(\mathsf{s}(x)) &\to \mathsf{s}(\mathsf{s}(\mathsf{double}(x)))
\end{aligned}$$

One can easily reduce the initial DP problem to $\{(74)\}$, with

$$\mathsf{F}(\mathsf{true}, x, y) \;\to\; \mathsf{F}(\mathsf{and}(\mathsf{gt}(x,y), \mathsf{gt}(y, \mathsf{s}(\mathsf{s}(\mathsf{0})))), \mathsf{plus}(\mathsf{s}(\mathsf{0}), x), \mathsf{double}(y)) \tag{74}$$

We use the polynomial interpretation $\mathcal{P}ol$ with $\mathsf{F}_{\mathcal{P}ol} = x_2 - x_3$, $\mathsf{0}_{\mathcal{P}ol} = \mathsf{0}$, $\mathsf{s}_{\mathcal{P}ol} = x_1 + 1$, $\mathsf{plus}_{\mathcal{P}ol} = x_1 + x_2$, and $\mathsf{double}_{\mathcal{P}ol} = 2 \cdot x_1$. The usable rules are the plus-rules and the reversed double-rules, both of which are weakly decreasing. To show that the DP is strictly decreasing, after simplification we have to consider the constraint

$$\begin{aligned}
&\mathsf{and}(\mathsf{gt}(x,y), \mathsf{gt}(y, \mathsf{s}(\mathsf{s}(\mathsf{0})))) = \mathsf{true} \\
&\qquad \Rightarrow\; \mathsf{F}(\mathsf{true}, x, y) \succ \mathsf{F}(\mathsf{and}(\mathsf{gt}(x,y), \mathsf{gt}(y, \mathsf{s}(\mathsf{s}(\mathsf{0})))), \mathsf{plus}(\mathsf{s}(\mathsf{0}), x), \mathsf{double}(y))
\end{aligned}$$

Rule (VII) and induction on and yield

$$\begin{aligned}&\mathsf{gt}(x,y) = \mathsf{true} \wedge \mathsf{s}(\mathsf{s}(0)) = v \wedge \mathsf{gt}(y,v) = \mathsf{true}\\ &\qquad \Rightarrow\ \mathsf{F}(\mathsf{true},x,y) \succ \mathsf{F}(\mathsf{and}(\mathsf{gt}(x,y),\mathsf{gt}(y,\mathsf{s}(\mathsf{s}(0)))),\mathsf{plus}(\mathsf{s}(0),x),\mathsf{double}(y))\end{aligned}$$

Induction on $\mathsf{gt}(y,v)$ twice and omitting all conditions yields

$$\mathsf{F}(\mathsf{true},x,\mathsf{s}(\mathsf{s}(z))) \succ \mathsf{F}(\ldots,\mathsf{plus}(\mathsf{s}(0),x),\mathsf{double}(\mathsf{s}(\mathsf{s}(z))))$$

which is satisfied by the above polynomial interpretation.

To show that the DP is also bounded we proceed in a similar way to the example in Sect. A.8.

A.13 Increase by Addition

As in the previous example, we use plus to increase an argument, but here we add two terms containing variables. Note that the TRS is neither sufficiently complete nor a constructor system.

$$\begin{aligned}\mathsf{div}(x,\mathsf{s}(y)) &\to \mathsf{d}(x,\mathsf{s}(y),0)\\ \mathsf{d}(x,\mathsf{s}(y),z) &\to \mathsf{cond}(\mathsf{ge}(x,z),x,y,z)\\ \mathsf{cond}(\mathsf{true},x,y,z) &\to \mathsf{s}(\mathsf{d}(x,\mathsf{s}(y),\mathsf{plus}(\mathsf{s}(y),z)))\\ \mathsf{cond}(\mathsf{false},x,y,z) &\to 0\\ \mathsf{ge}(u,0) &\to \mathsf{true}\\ \mathsf{ge}(0,\mathsf{s}(v)) &\to \mathsf{false}\\ \mathsf{ge}(\mathsf{s}(u),\mathsf{s}(v)) &\to \mathsf{ge}(u,v)\\ \mathsf{plus}(n,0) &\to n\\ \mathsf{plus}(n,\mathsf{s}(m)) &\to \mathsf{s}(\mathsf{plus}(n,m))\\ \mathsf{plus}(\mathsf{plus}(n,m),u) &\to \mathsf{plus}(n,\mathsf{plus}(m,u))\end{aligned}$$

One can easily reduce the initial DP problem to $\{(75),(76)\}$, with

$$\mathsf{D}(x,\mathsf{s}(y),z) \to \mathsf{COND}(\mathsf{ge}(x,z),x,y,z) \tag{75}$$

$$\mathsf{COND}(\mathsf{true},x,y,z) \to \mathsf{D}(x,\mathsf{s}(y),\mathsf{plus}(\mathsf{s}(y),z)) \tag{76}$$

We use the polynomial interpretation $\mathcal{P}ol$ with $\mathsf{D}_{\mathcal{P}ol} = x_1+x_2-x_3$, $\mathsf{COND}_{\mathcal{P}ol} = x_2+x_3-x_4+1$, $0_{\mathcal{P}ol} = 0$, $\mathsf{s}_{\mathcal{P}ol} = x_1+1$, and $\mathsf{plus}_{\mathcal{P}ol} = x_1+x_2$. The usable rules are the reversed plus-rules which are weakly decreasing. Moreover, (75) is weakly decreasing and (76) is strictly decreasing. To show that (75) is bounded, we have to consider the constraint

$$\mathsf{COND}(\mathsf{ge}(x,z),x,y,z) = \mathsf{COND}(\mathsf{true},x',y',z') \Rightarrow \mathsf{D}(x,\mathsf{s}(y),z) \succsim \mathsf{c}$$

which can be handled as before.

A.14 Increase in Different Arguments

The following TRS computes $|x - y|$.

$$\begin{aligned}
\mathsf{diff}(x, y) &\to \mathsf{cond}_1(\mathsf{equal}(x, y), x, y)\\
\mathsf{cond}_1(\mathsf{true}, x, y) &\to \mathsf{0}\\
\mathsf{cond}_1(\mathsf{false}, x, y) &\to \mathsf{cond}_2(\mathsf{gt}(x, y), x, y)\\
\mathsf{cond}_2(\mathsf{true}, x, y) &\to \mathsf{s}(\mathsf{diff}(x, \mathsf{s}(y)))\\
\mathsf{cond}_2(\mathsf{false}, x, y) &\to \mathsf{s}(\mathsf{diff}(\mathsf{s}(x), y))\\
\mathsf{gt}(\mathsf{0}, v) &\to \mathsf{false}\\
\mathsf{gt}(\mathsf{s}(u), \mathsf{0}) &\to \mathsf{true}\\
\mathsf{gt}(\mathsf{s}(u), \mathsf{s}(v)) &\to \mathsf{gt}(u, v)\\
\mathsf{equal}(\mathsf{0}, \mathsf{0}) &\to \mathsf{true}\\
\mathsf{equal}(\mathsf{s}(x), \mathsf{0}) &\to \mathsf{false}\\
\mathsf{equal}(\mathsf{0}, \mathsf{s}(y)) &\to \mathsf{false}\\
\mathsf{equal}(\mathsf{s}(x), \mathsf{s}(y)) &\to \mathsf{equal}(x, y)
\end{aligned}$$

One can easily reduce the initial DP problem to $\{(77), (78), (79), (80)\}$, with

$$\mathsf{DIFF}(x, y) \to \mathsf{COND}_1(\mathsf{equal}(x, y), x, y) \tag{77}$$

$$\mathsf{COND}_1(\mathsf{false}, x, y) \to \mathsf{COND}_2(\mathsf{gt}(x, y), x, y) \tag{78}$$

$$\mathsf{COND}_2(\mathsf{true}, x, y) \to \mathsf{DIFF}(x, \mathsf{s}(y)) \tag{79}$$

$$\mathsf{COND}_2(\mathsf{false}, x, y) \to \mathsf{DIFF}(\mathsf{s}(x), y) \tag{80}$$

We use the polynomial interpretation $\mathcal{P}ol$ with $\mathsf{DIFF}_{\mathcal{P}ol} = (x_1 - x_2)^2$, $\mathsf{COND1}_{\mathcal{P}ol} = \mathsf{COND2}_{\mathcal{P}ol} = (x_2 - x_3)^2$, $\mathsf{0}_{\mathcal{P}ol} = 0$, and $\mathsf{s}_{\mathcal{P}ol} = x_1 + 1$. It is easy to prove that all DPs are bounded. Moreover, (79) and (80) are strictly decreasing. Then the remaining DP problems can easily be solved.

We only show that (80) is strictly decreasing. The proof for (79) is analogous. Again, we need the refinement of Thm. 11 which considers more than two adjacent dependency pairs in innermost chains. Here we regard the dependency pairs which *precede* (80), cf. the corresponding refinement of Thm. 11. We consider the constraint

$$\begin{aligned}
&\mathsf{COND}_1(\mathsf{equal}(x'', y''), x'', y'') = \mathsf{COND}_1(\mathsf{false}, x', y')\\
\wedge\ &\mathsf{COND}_2(\mathsf{gt}(x', y'), x', y') = \mathsf{COND}_2(\mathsf{false}, x, y)\\
&\qquad\qquad\qquad\Rightarrow \mathsf{COND}_2(\mathsf{false}, x, y) \succ \mathsf{DIFF}(\mathsf{s}(x), y)
\end{aligned}$$

This can be simplified to

$$\mathsf{equal}(x, y) = \mathsf{false} \wedge \mathsf{gt}(x, y) = \mathsf{false} \quad\Rightarrow\quad \mathsf{COND}_2(\mathsf{false}, x, y) \succ \mathsf{DIFF}(\mathsf{s}(x), y)$$

Induction on $\mathsf{gt}(x, y)$ and simplifications with Rules (I) and (II) yield

$$\begin{aligned}
&\mathsf{equal}(\mathsf{0}, v) = \mathsf{false} \\
&\qquad\qquad\qquad \Rightarrow \mathsf{COND}_2(\mathsf{false}, \mathsf{0}, v) \succ \mathsf{DIFF}(\mathsf{s}(\mathsf{0}), v) && (81)\\
&\mathsf{equal}(\mathsf{s}(u), \mathsf{s}(v)) = \mathsf{false} \ \wedge\ \mathsf{gt}(u, v) = \mathsf{false}\ \wedge \varphi \\
&\qquad\qquad\qquad \Rightarrow \mathsf{COND}_2(\mathsf{false}, \mathsf{s}(u), \mathsf{s}(v)) \succ \mathsf{DIFF}(\mathsf{s}(\mathsf{s}(u)), \mathsf{s}(v)) && (82)
\end{aligned}$$

with the following induction hypothesis φ:

$$\mathsf{equal}(u, v) = \mathsf{false} \wedge \mathsf{gt}(u, v) = \mathsf{false}\ \Rightarrow\ \mathsf{COND}_2(\mathsf{false}, u, v) \succ \mathsf{DIFF}(\mathsf{s}(u), v)$$

Constraint (81) is simplified by Rule (VII) and induction on equal yields the following final constraint after simplifications with Rules (I) and (II).

$$\mathsf{COND}_2(\mathsf{false}, \mathsf{0}, \mathsf{s}(m)) \succ \mathsf{DIFF}(\mathsf{s}(\mathsf{0}), \mathsf{s}(m)) \tag{83}$$

Constraint (82) is also simplified by Rule (VII) and then we perform induction on equal. After the application of Rules (I) and (II) we obtain

$$\begin{aligned}
&\mathsf{equal}(n, m) = \mathsf{false} \wedge u = n \wedge v = m\ \wedge\ \mathsf{gt}(u, v) = \mathsf{false}\ \wedge \varphi\ \wedge\ \varphi' \\
&\qquad\qquad\qquad \Rightarrow \mathsf{COND}_2(\mathsf{false}, \mathsf{s}(n), \mathsf{s}(m)) \succ \mathsf{DIFF}(\mathsf{s}(\mathsf{s}(n)), \mathsf{s}(m))
\end{aligned}$$

with a new induction hypothesis φ'. Simplifying with Rule (III) and (VI) using the induction hypothesis φ we result in the final constraint

$$\begin{aligned}
&\mathsf{COND}_2(\mathsf{false}, n, m) \succ \mathsf{DIFF}(\mathsf{s}(n), m) \\
\Rightarrow\ &\mathsf{COND}_2(\mathsf{false}, \mathsf{s}(n), \mathsf{s}(m)) \succ \mathsf{DIFF}(\mathsf{s}(\mathsf{s}(n)), \mathsf{s}(m))
\end{aligned} \tag{84}$$

Then constraints (83) and (84) are obviously satisfied.

A.15 Sorting Algorithm

The following TRS sorts a list. Here, the bound contains a list ℓ, where max computes the maximum of a list. The function $\mathsf{st}(n, \ell)$ returns the sorted list of elements of ℓ that are greater than or equal to n. Duplicates are removed.

$$\begin{aligned}
\mathsf{sort}(\ell) &\to \mathsf{st}(\mathsf{0}, \ell) \\
\mathsf{st}(n, \ell) &\to \mathsf{cond}_1(\mathsf{member}(n, \ell), n, \ell) \\
\mathsf{cond}_1(\mathsf{true}, n, \ell) &\to \mathsf{cons}(n, \mathsf{st}(\mathsf{s}(n), \ell)) \\
\mathsf{cond}_1(\mathsf{false}, n, \ell) &\to \mathsf{cond}_2(\mathsf{gt}(n, \mathsf{max}(\ell)), n, \ell) \\
\mathsf{cond}_2(\mathsf{true}, n, \ell) &\to \mathsf{nil} \\
\mathsf{cond}_2(\mathsf{false}, n, \ell) &\to \mathsf{st}(\mathsf{s}(n), \ell)
\end{aligned}$$

$$
\begin{aligned}
\mathsf{member}(n, \mathsf{nil}) &\to \mathsf{false}\\
\mathsf{member}(n, \mathsf{cons}(m, \ell)) &\to \mathsf{or}(\mathsf{equal}(n, m), \mathsf{member}(n, \ell))\\
\mathsf{or}(x, \mathsf{true}) &\to \mathsf{true}\\
\mathsf{or}(x, \mathsf{false}) &\to x\\
\mathsf{equal}(\mathsf{0}, \mathsf{0}) &\to \mathsf{true}\\
\mathsf{equal}(\mathsf{s}(x), \mathsf{0}) &\to \mathsf{false}\\
\mathsf{equal}(\mathsf{0}, \mathsf{s}(y)) &\to \mathsf{false}\\
\mathsf{equal}(\mathsf{s}(x), \mathsf{s}(y)) &\to \mathsf{equal}(x, y)\\
\mathsf{gt}(\mathsf{0}, v) &\to \mathsf{false}\\
\mathsf{gt}(\mathsf{s}(u), \mathsf{0}) &\to \mathsf{true}\\
\mathsf{gt}(\mathsf{s}(u), \mathsf{s}(v)) &\to \mathsf{gt}(u, v)\\
\mathsf{max}(\mathsf{nil}) &\to \mathsf{0}\\
\mathsf{max}(\mathsf{cons}(u, \ell)) &\to \mathsf{if}(\mathsf{gt}(u, \mathsf{max}(\ell)), u, \mathsf{max}(\ell))\\
\mathsf{if}(\mathsf{true}, u, v) &\to u\\
\mathsf{if}(\mathsf{false}, u, v) &\to v
\end{aligned}
$$

One can easily reduce the initial DP problem to $\{(85), (86), (87), (88)\}$, with

$$
\begin{aligned}
\mathsf{ST}(n, \ell) &\to \mathsf{COND}_1(\mathsf{member}(n, \ell), n, \ell) && (85)\\
\mathsf{COND}_1(\mathsf{true}, n, \ell) &\to \mathsf{ST}(\mathsf{s}(n), \ell) && (86)\\
\mathsf{COND}_1(\mathsf{false}, n, \ell) &\to \mathsf{COND}_2(\mathsf{gt}(n, \mathsf{max}(\ell)), n, \ell) && (87)\\
\mathsf{COND}_2(\mathsf{false}, n, \ell) &\to \mathsf{ST}(\mathsf{s}(n), \ell) && (88)
\end{aligned}
$$

We use the polynomial interpretation $\mathcal{P}ol$ with $\mathsf{ST}_{\mathcal{P}ol} = x_2 - x_1$, $\mathsf{COND}_{1\mathcal{P}ol} = \mathsf{COND}_{2\mathcal{P}ol} = x_3 - x_2$, $\mathsf{0}_{\mathcal{P}ol} = 0$, $\mathsf{s}_{\mathcal{P}ol} = x_1 + 1$, $\mathsf{nil}_{\mathcal{P}ol} = 0$, and $\mathsf{cons}_{\mathcal{P}ol} = x_1 + x_2$. It is easy to prove that (85) and (87) are weakly decreasing and (86) and (88) are strictly decreasing. The DP problem $\{(85), (87)\}$ without the strictly decreasing DPs can easily be handled.

We now show that (87) is bounded. Here we obtain the constraint

$$\mathsf{gt}(n, \mathsf{max}(\ell)) = \mathsf{false} \Rightarrow \mathsf{COND}_1(\mathsf{false}, n, \ell) \succsim \mathsf{c}$$

After application of Rule (VII) we have

$$\mathsf{max}(\ell) = m \wedge \mathsf{gt}(n, m) = \mathsf{false} \Rightarrow \mathsf{COND}_1(\mathsf{false}, n, \ell) \succsim \mathsf{c}$$

Now we can perform induction on gt which yields $\mathsf{COND}_1(\mathsf{false}, \mathsf{0}, \ell) \succsim \mathsf{c}$ (which is satisfied by $\mathcal{P}ol$) and

$$\mathsf{max}(\ell) = \mathsf{s}(v) \wedge \mathsf{gt}(u, v) = \mathsf{false} \wedge \varphi' \Rightarrow \mathsf{COND}_1(\mathsf{false}, \mathsf{s}(u), \ell) \succsim \mathsf{c}.$$

Here φ' is the induction hypothesis, which can be eliminated by Rule (IV) again. Next we perform induction on max. This yields

$$\mathsf{if}(\mathsf{gt}(x, \mathsf{max}(\ell')), x, \mathsf{max}(\ell')) = \mathsf{s}(v) \wedge \mathsf{gt}(u,v) = \mathsf{false} \wedge \varphi'' \\ \Rightarrow \mathsf{COND}_1(\mathsf{false}, \mathsf{s}(u), \mathsf{cons}(x, \ell')) \succsim \mathsf{c}.$$

Here φ'' is the induction hypothesis:

$$\varphi'' : \quad \forall\, u, v \;\; \mathsf{max}(\ell') = \mathsf{s}(v) \wedge \mathsf{gt}(u,v) = \mathsf{false} \;\Rightarrow\; \mathsf{COND}_1(\mathsf{false}, \mathsf{s}(u), \ell') \succsim \mathsf{c}.$$

We apply Rule (VII) to obtain

$$\mathsf{gt}(x, \mathsf{max}(\ell')) = a \wedge \mathsf{max}(\ell') = b \wedge \mathsf{if}(a, x, b) = \mathsf{s}(v) \wedge \mathsf{gt}(u,v) = \mathsf{false} \wedge \varphi'' \\ \Rightarrow \mathsf{COND}_1(\mathsf{false}, \mathsf{s}(u), \mathsf{cons}(x, \ell')) \succsim \mathsf{c}.$$

Now we perform induction on if. The first resulting constraint is

$$\mathsf{gt}(x, \mathsf{max}(\ell')) = \mathsf{true} \wedge \mathsf{max}(\ell') = b \wedge x = \mathsf{s}(v) \wedge \mathsf{gt}(u,v) = \mathsf{false} \wedge \varphi'' \\ \Rightarrow \mathsf{COND}_1(\mathsf{false}, \mathsf{s}(u), \mathsf{cons}(x, \ell')) \succsim \mathsf{c}$$

which is simplified to $\mathsf{gt}(u,v) = \mathsf{false} \Rightarrow \mathsf{COND}_1(\mathsf{false}, \mathsf{s}(u), \mathsf{cons}(\mathsf{s}(v), \ell')) \succsim \mathsf{c}$. This constraint can be transformed similar to previous examples.

The other resulting constraint is

$$\mathsf{gt}(x, \mathsf{max}(\ell')) = \mathsf{false} \wedge \mathsf{max}(\ell') = b \wedge b = \mathsf{s}(v) \wedge \mathsf{gt}(u,v) = \mathsf{false} \wedge \varphi'' \\ \Rightarrow \mathsf{COND}_1(\mathsf{false}, \mathsf{s}(u), \mathsf{cons}(x, \ell')) \succsim \mathsf{c}$$

which is simplified to

$$\mathsf{max}(\ell') = \mathsf{s}(v) \wedge \mathsf{gt}(u,v) = \mathsf{false} \wedge \varphi'' \Rightarrow \mathsf{ST}(\mathsf{s}(u), \mathsf{cons}(x, \ell')) \succsim \mathsf{c}$$

Now we apply the induction hypothesis φ'' and finally obtain

$$\mathsf{COND}_1(\mathsf{false}, \mathsf{s}(u), \ell') \succsim \mathsf{c} \Rightarrow \mathsf{COND}_1(\mathsf{false}, \mathsf{s}(u), \mathsf{cons}(x, \ell')) \succsim \mathsf{c}$$

which is obviously satisfied by $\mathcal{P}ol$.

Since (87) is bounded, we end up with the DP problem $\{(85), (86), (88)\}$. We now show that (in this reduced DP problem), (85) is bounded. Then all remaining DP problems are easy to solve. For boundedness of (85), we obtain the constraint

$$\mathsf{COND}_1(\mathsf{member}(n, \ell), n, \ell) = \mathsf{COND}_1(\mathsf{true}, n', \ell') \Rightarrow \mathsf{ST}(n, \ell) \succsim \mathsf{c}$$

which can be simplified to

$$\mathsf{member}(n, \ell) = \mathsf{true} \Rightarrow \mathsf{ST}(n, \ell) \succsim \mathsf{c}$$

By induction on member, we transform it to

$$\mathsf{or}(\mathsf{equal}(n, m), \mathsf{member}(n, \ell')) = \mathsf{true} \wedge \varphi \Rightarrow \mathsf{ST}(n, \mathsf{cons}(m, \ell')) \succsim \mathsf{c}$$

Here, φ is the induction hypothesis:

$$\varphi : \qquad \mathsf{member}(n, \ell') = \mathsf{true} \Rightarrow \mathsf{ST}(n, \ell') \succsim \mathsf{c}$$

By applying Rule (VII) and induction on or we obtain the following two constraints:

$$\mathsf{equal}(n,m) = x \wedge \mathsf{member}(n,\ell') = \mathsf{true} \wedge \varphi \Rightarrow \mathsf{ST}(n,\mathsf{cons}(m,\ell')) \succsim \mathsf{c} \quad (89)$$
$$\mathsf{equal}(n,m) = \mathsf{true} \wedge \mathsf{member}(n,\ell') = \mathsf{false} \wedge \varphi \Rightarrow \mathsf{ST}(n,\mathsf{cons}(m,\ell')) \succsim \mathsf{c} \quad (90)$$

For (89), we apply the induction hypothesis and end up with the following constraint that is obviously satisfied by $\mathcal{P}ol$.

$$\mathsf{ST}(n,\ell') \succsim \mathsf{c} \Rightarrow \mathsf{ST}(n,\mathsf{cons}(m,\ell')) \succsim \mathsf{c}$$

By induction on equal, Constraint (90) is also transformed into constraints that are clearly satisfied by $\mathcal{P}ol$.

A.16 Bound on List Length

The function $\mathsf{nthtail}(n,\ell)$ returns the last n elements of the list ℓ.

$$\begin{aligned}
\mathsf{nthtail}(n,\ell) &\to \mathsf{cond}(\mathsf{ge}(n,\mathsf{length}(\ell)),n,\ell)\\
\mathsf{cond}(\mathsf{true},n,\ell) &\to \ell\\
\mathsf{cond}(\mathsf{false},n,\ell) &\to \mathsf{tail}(\mathsf{nthtail}(\mathsf{s}(n),\ell))\\
\mathsf{tail}(\mathsf{nil}) &\to \mathsf{nil}\\
\mathsf{tail}(\mathsf{cons}(x,\ell)) &\to \ell\\
\mathsf{length}(\mathsf{nil}) &\to \mathsf{0}\\
\mathsf{length}(\mathsf{cons}(x,\ell)) &\to \mathsf{s}(\mathsf{length}(\ell))\\
\mathsf{ge}(u,\mathsf{0}) &\to \mathsf{true}\\
\mathsf{ge}(\mathsf{0},\mathsf{s}(v)) &\to \mathsf{false}\\
\mathsf{ge}(\mathsf{s}(u),\mathsf{s}(v)) &\to \mathsf{ge}(u,v)
\end{aligned}$$

One can easily reduce the initial DP problem to $\{(91),(92)\}$, with

$$\mathsf{NTHTAIL}(n,\ell) \to \mathsf{COND}(\mathsf{ge}(n,\mathsf{length}(\ell)),n,\ell) \quad (91)$$
$$\mathsf{COND}(\mathsf{false},n,\ell) \to \mathsf{NTHTAIL}(\mathsf{s}(n),\ell) \quad (92)$$

We use the polynomial interpretation $\mathcal{P}ol$ with $\mathsf{NTHTAIL}_{\mathcal{P}ol} = x_2 - x_1$, $\mathsf{COND}_{\mathcal{P}ol} = x_3 - x_2$, $\mathsf{nil}_{\mathcal{P}ol} = 0$, $\mathsf{cons}_{\mathcal{P}ol} = 1 + x_1$, $\mathsf{0}_{\mathcal{P}ol} = 0$, and $\mathsf{s}_{\mathcal{P}ol} = x_1 + 1$. Clearly, (91) is weakly decreasing and (92) is strictly decreasing. Moreover, (91) is bounded. We have to regard a constraint which can be simplified to

$$\mathsf{ge}(n,\mathsf{length}(\ell)) = \mathsf{false} \Rightarrow \mathsf{NTHTAIL}(n,\ell) \succsim \mathsf{c}$$

We apply Rule (VII) to obtain

$$\mathsf{length}(\ell) = m \wedge \mathsf{ge}(n,m) = \mathsf{false} \Rightarrow \mathsf{NTHTAIL}(n,\ell) \succsim \mathsf{c}$$

By induction on ge we get $\mathsf{NTHTAIL}(0,\ell) \succsim \mathsf{c}$ (which is obviously satisfied by $\mathcal{P}ol$) and

$$\mathsf{length}(\ell) = \mathsf{s}(v) \wedge \mathsf{ge}(u,v) = \mathsf{false} \wedge \varphi \Rightarrow \mathsf{NTHTAIL}(\mathsf{s}(u),\ell) \succsim \mathsf{c}$$

where φ is the following induction hypothesis:

$$\varphi: \qquad \forall\, \ell \;\; \mathsf{length}(\ell) = v \wedge \mathsf{ge}(u,v) = \mathsf{false} \Rightarrow \mathsf{NTHTAIL}(u,\ell) \succsim \mathsf{c}$$

Now we apply induction on length. We result in

$$\mathsf{length}(\ell') = v \wedge \mathsf{ge}(u,v) = \mathsf{false} \wedge \varphi \wedge \varphi' \Rightarrow \mathsf{NTHTAIL}(\mathsf{s}(u),\mathsf{cons}(x,\ell')) \succsim \mathsf{c}$$

where φ' is the new induction hypothesis. Now we apply the induction hypothesis φ' by instantiating ℓ with ℓ'. Finally, we end up with

$$\mathsf{NTHTAIL}(u,\ell') \succsim \mathsf{c} \succsim \mathsf{NTHTAIL}(\mathsf{s}(u),\mathsf{cons}(x,\ell'))$$

which is satisfied by $\mathcal{P}ol$.

A.17 Bound with Square Function

In the following TRS, $\mathsf{log}(x,y)$ computes the smallest power of 2 which is greater than or equal to $\log_y(x)$.

$$\begin{aligned}
\mathsf{log}(x,\mathsf{s}(\mathsf{s}(y))) &\to \mathsf{cond}(\mathsf{le}(x,\mathsf{s}(\mathsf{s}(y))),x,y)\\
\mathsf{cond}(\mathsf{true},x,y) &\to \mathsf{s}(0)\\
\mathsf{cond}(\mathsf{false},x,y) &\to \mathsf{double}(\mathsf{log}(x,\mathsf{square}(\mathsf{s}(\mathsf{s}(y)))))\\
\mathsf{le}(0,v) &\to \mathsf{true}\\
\mathsf{le}(\mathsf{s}(u),0)) &\to \mathsf{false}\\
\mathsf{le}(\mathsf{s}(u),\mathsf{s}(v)) &\to \mathsf{le}(u,v)\\
\mathsf{double}(0) &\to 0\\
\mathsf{double}(\mathsf{s}(x)) &\to \mathsf{s}(\mathsf{s}(\mathsf{double}(x)))\\
\mathsf{square}(0) &\to 0\\
\mathsf{square}(\mathsf{s}(x)) &\to \mathsf{s}(\mathsf{plus}(\mathsf{square}(x),\mathsf{double}(x)))\\
\mathsf{plus}(n,0) &\to n\\
\mathsf{plus}(n,\mathsf{s}(m)) &\to \mathsf{s}(\mathsf{plus}(n,m))
\end{aligned}$$

One can easily reduce the initial DP problem to $\{(93),(94)\}$, with

$$\mathsf{LOG}(x,\mathsf{s}(\mathsf{s}(y))) \to \mathsf{COND}(\mathsf{le}(x,\mathsf{s}(\mathsf{s}(y))),x,y) \tag{93}$$
$$\mathsf{COND}(\mathsf{false},x,y) \to \mathsf{LOG}(x,\mathsf{square}(\mathsf{s}(\mathsf{s}(y)))) \tag{94}$$

We use the polynomial interpretation $\mathcal{P}ol$ with $\mathsf{LOG}_{\mathcal{P}ol} = x_1 - x_2$, $\mathsf{COND}_{\mathcal{P}ol} = x_2 - x_3 - 2$, $0_{\mathcal{P}ol} = 0$, $\mathsf{s}_{\mathcal{P}ol} = x_1 + 1$, $\mathsf{square}_{\mathcal{P}ol} = x_1$, and $\mathsf{plus}_{\mathcal{P}ol} = x$. The usable rules are the reversed square- and the reversed plus-rules. Clearly, (93) and the usable rules are weakly decreasing. To show that (94) is strictly decreasing, one can use the rewriting technique from [8]. We show that (93) is bounded. The corresponding constraint is simplified to

$$\mathsf{le}(x, \mathsf{s}(\mathsf{s}(y))) = \mathsf{false} \Rightarrow \mathsf{LOG}(x, \mathsf{s}(\mathsf{s}(y))) \succsim \mathsf{c}$$

and by application of Rule (VII) we obtain

$$\mathsf{s}(\mathsf{s}(y)) = z \wedge \mathsf{le}(x, z) = \mathsf{false} \Rightarrow \mathsf{LOG}(x, \mathsf{s}(\mathsf{s}(y))) \succsim \mathsf{c}$$

Induction on le yields

$$\mathsf{s}(y) = v \wedge \mathsf{le}(u, v) = \mathsf{false} \wedge \varphi \Rightarrow \mathsf{LOG}(\mathsf{s}(u), \mathsf{s}(\mathsf{s}(y))) \succsim \mathsf{c}$$

where φ is the induction hypothesis. We eliminate it with Rule (IV) and perform another induction on le which results in

$$y = v' \wedge \mathsf{le}(u', v') = \mathsf{false} \wedge \varphi' \Rightarrow \mathsf{LOG}(\mathsf{s}(\mathsf{s}(u')), \mathsf{s}(\mathsf{s}(y))) \succsim \mathsf{c}$$

Now Rule (III) yields

$$\mathsf{le}(u', v') = \mathsf{false} \wedge \varphi' \Rightarrow \mathsf{LOG}(\mathsf{s}(\mathsf{s}(u')), \mathsf{s}(\mathsf{s}(v'))) \succsim \mathsf{c}.$$

After eliminating the induction hypothesis φ' one can apply another induction on le, similar to previous examples.

Aachener Informatik-Berichte

This list contains all technical reports published during the past five years. A complete list of reports dating back to 1987 is available from `http://aib.informatik.rwth-aachen.de/`**. To obtain copies consult the above URL or send your request to: Informatik-Bibliothek, RWTH Aachen, Ahornstr. 55, 52056 Aachen, Email:** `biblio@informatik.rwth-aachen.de`

2001-01 * Jahresbericht 2000
2001-02 Benedikt Bollig, Martin Leucker: Deciding LTL over Mazurkiewicz Traces
2001-03 Thierry Cachat: The power of one-letter rational languages
2001-04 Benedikt Bollig, Martin Leucker, Michael Weber: Local Parallel Model Checking for the Alternation Free mu-Calculus
2001-05 Benedikt Bollig, Martin Leucker, Thomas Noll: Regular MSC Languages
2001-06 Achim Blumensath: Prefix-Recognisable Graphs and Monadic Second-Order Logic
2001-07 Martin Grohe, Stefan Wöhrle: An Existential Locality Theorem
2001-08 Mareike Schoop, James Taylor (eds.): Proceedings of the Sixth International Workshop on the Language-Action Perspective on Communication Modelling
2001-09 Thomas Arts, Jürgen Giesl: A collection of examples for termination of term rewriting using dependency pairs
2001-10 Achim Blumensath: Axiomatising Tree-interpretable Structures
2001-11 Klaus Indermark, Thomas Noll (eds.): Kolloquium Programmiersprachen und Grundlagen der Programmierung
2002-01 * Jahresbericht 2001
2002-02 Jürgen Giesl, Aart Middeldorp: Transformation Techniques for Context-Sensitive Rewrite Systems
2002-03 Benedikt Bollig, Martin Leucker, Thomas Noll: Generalised Regular MSC Languages
2002-04 Jürgen Giesl, Aart Middeldorp: Innermost Termination of Context-Sensitive Rewriting
2002-05 Horst Lichter, Thomas von der Maßen, Thomas Weiler: Modelling Requirements and Architectures for Software Product Lines
2002-06 Henry N. Adorna: 3-Party Message Complexity is Better than 2-Party Ones for Proving Lower Bounds on the Size of Minimal Nondeterministic Finite Automata
2002-07 Jörg Dahmen: Invariant Image Object Recognition using Gaussian Mixture Densities
2002-08 Markus Mohnen: An Open Framework for Data-Flow Analysis in Java
2002-09 Markus Mohnen: Interfaces with Default Implementations in Java
2002-10 Martin Leucker: Logics for Mazurkiewicz traces
2002-11 Jürgen Giesl, Hans Zantema: Liveness in Rewriting

2003-01 * Jahresbericht 2002
2003-02 Jürgen Giesl, René Thiemann: Size-Change Termination for Term Rewriting
2003-03 Jürgen Giesl, Deepak Kapur: Deciding Inductive Validity of Equations
2003-04 Jürgen Giesl, René Thiemann, Peter Schneider-Kamp, Stephan Falke: Improving Dependency Pairs
2003-05 Christof Löding, Philipp Rohde: Solving the Sabotage Game is PSPACE-hard
2003-06 Franz Josef Och: Statistical Machine Translation: From Single-Word Models to Alignment Templates
2003-07 Horst Lichter, Thomas von der Maßen, Alexander Nyßen, Thomas Weiler: Vergleich von Ansätzen zur Feature Modellierung bei der Softwareproduktlinienentwicklung
2003-08 Jürgen Giesl, René Thiemann, Peter Schneider-Kamp, Stephan Falke: Mechanizing Dependency Pairs
2004-01 * Fachgruppe Informatik: Jahresbericht 2003
2004-02 Benedikt Bollig, Martin Leucker: Message-Passing Automata are expressively equivalent to EMSO logic
2004-03 Delia Kesner, Femke van Raamsdonk, Joe Wells (eds.): HOR 2004 – 2nd International Workshop on Higher-Order Rewriting
2004-04 Slim Abdennadher, Christophe Ringeissen (eds.): RULE 04 – Fifth International Workshop on Rule-Based Programming
2004-05 Herbert Kuchen (ed.): WFLP 04 – 13th International Workshop on Functional and (Constraint) Logic Programming
2004-06 Sergio Antoy, Yoshihito Toyama (eds.): WRS 04 – 4th International Workshop on Reduction Strategies in Rewriting and Programming
2004-07 Michael Codish, Aart Middeldorp (eds.): WST 04 – 7th International Workshop on Termination
2004-08 Klaus Indermark, Thomas Noll: Algebraic Correctness Proofs for Compiling Recursive Function Definitions with Strictness Information
2004-09 Joachim Kneis, Daniel Mölle, Stefan Richter, Peter Rossmanith: Parameterized Power Domination Complexity
2004-10 Zinaida Benenson, Felix C. Gärtner, Dogan Kesdogan: Secure Multi-Party Computation with Security Modules
2005-01 * Fachgruppe Informatik: Jahresbericht 2004
2005-02 Maximillian Dornseif, Felix C. Gärtner, Thorsten Holz, Martin Mink: An Offensive Approach to Teaching Information Security: "Aachen Summer School Applied IT Security"
2005-03 Jürgen Giesl, René Thiemann, Peter Schneider-Kamp: Proving and Disproving Termination of Higher-Order Functions
2005-04 Daniel Mölle, Stefan Richter, Peter Rossmanith: A Faster Algorithm for the Steiner Tree Problem
2005-05 Fabien Pouget, Thorsten Holz: A Pointillist Approach for Comparing Honeypots
2005-06 Simon Fischer, Berthold Vöcking: Adaptive Routing with Stale Information

2005-07 Felix C. Freiling, Thorsten Holz, Georg Wicherski: Botnet Tracking: Exploring a Root-Cause Methodology to Prevent Distributed Denial-of-Service Attacks
2005-08 Joachim Kneis, Peter Rossmanith: A New Satisfiability Algorithm With Applications To Max-Cut
2005-09 Klaus Kursawe, Felix C. Freiling: Byzantine Fault Tolerance on General Hybrid Adversary Structures
2005-10 Benedikt Bollig: Automata and Logics for Message Sequence Charts
2005-11 Simon Fischer, Berthold Vöcking: A Counterexample to the Fully Mixed Nash Equilibrium Conjecture
2005-12 Neeraj Mittal, Felix Freiling, S. Venkatesan, Lucia Draque Penso: Efficient Reductions for Wait-Free Termination Detection in Faulty Distributed Systems
2005-13 Carole Delporte-Gallet, Hugues Fauconnier, Felix C. Freiling: Revisiting Failure Detection and Consensus in Omission Failure Environments
2005-14 Felix C. Freiling, Sukumar Ghosh: Code Stabilization
2005-15 Uwe Naumann: The Complexity of Derivative Computation
2005-16 Uwe Naumann: Syntax-Directed Derivative Code (Part I: Tangent-Linear Code)
2005-17 Uwe Naumann: Syntax-directed Derivative Code (Part II: Intraprocedural Adjoint Code)
2005-18 Thomas von der Maßen, Klaus Müller, John MacGregor, Eva Geisberger, Jörg Dörr, Frank Houdek, Harbhajan Singh, Holger Wußmann, Hans-Veit Bacher, Barbara Paech: Einsatz von Features im Software-Entwicklungsprozess - Abschlußbericht des GI-Arbeitskreises "Features"
2005-19 Uwe Naumann, Andre Vehreschild: Tangent-Linear Code by Augmented LL-Parsers
2005-20 Felix C. Freiling, Martin Mink: Bericht über den Workshop zur Ausbildung im Bereich IT-Sicherheit Hochschulausbildung, berufliche Weiterbildung, Zertifizierung von Ausbildungsangeboten am 11. und 12. August 2005 in Köln organisiert von RWTH Aachen in Kooperation mit BITKOM, BSI, DLR und Gesellschaft fuer Informatik (GI) e.V.
2005-21 Thomas Noll, Stefan Rieger: Optimization of Straight-Line Code Revisited
2005-22 Felix Freiling, Maurice Herlihy, Lucia Draque Penso: Optimal Randomized Fair Exchange with Secret Shared Coins
2005-23 Heiner Ackermann, Alantha Newman, Heiko Röglin, Berthold Vöcking: Decision Making Based on Approximate and Smoothed Pareto Curves
2005-24 Alexander Becher, Zinaida Benenson, Maximillian Dornseif: Tampering with Motes: Real-World Physical Attacks on Wireless Sensor Networks
2006-01 * Fachgruppe Informatik: Jahresbericht 2005
2006-02 Michael Weber: Parallel Algorithms for Verification of Large Systems
2006-03 Michael Maier, Uwe Naumann: Intraprocedural Adjoint Code Generated by the Differentiation-Enabled NAGWare Fortran Compiler
2006-04 Ebadollah Varnik, Uwe Naumann, Andrew Lyons: Toward Low Static Memory Jacobian Accumulation

2006-05	Uwe Naumann, Jean Utke, Patrick Heimbach, Chris Hill, Derya Ozyurt, Carl Wunsch, Mike Fagan, Nathan Tallent, Michelle Strout: Adjoint Code by Source Transformation with OpenAD/F
2006-06	Joachim Kneis, Daniel Mölle, Stefan Richter, Peter Rossmanith: Divide-and-Color
2006-07	Thomas Colcombet, Christof Löding: Transforming structures by set interpretations
2006-08	Uwe Naumann, Yuxiao Hu: Optimal Vertex Elimination in Single-Expression-Use Graphs
2006-09	Tingting Han, Joost-Pieter Katoen: Counterexamples in Probabilistic Model Checking
2006-10	Mesut Günes, Alexander Zimmermann, Martin Wenig, Jan Ritzerfeld, Ulrich Meis: From Simulations to Testbeds - Architecture of the Hybrid MCG-Mesh Testbed
2006-11	Bastian Schlich, Michael Rohrbach, Michael Weber, Stefan Kowalewski: Model Checking Software for Microcontrollers
2006-12	Benedikt Bollig, Joost-Pieter Katoen, Carsten Kern, Martin Leucker: Replaying Play in and Play out: Synthesis of Design Models from Scenarios by Learning
2006-13	Wong Karianto, Christof Löding: Unranked Tree Automata with Sibling Equalities and Disequalities
2006-14	Danilo Beuche, Andreas Birk, Heinrich Dreier, Andreas Fleischmann, Heidi Galle, Gerald Heller, Dirk Janzen, Isabel John, Ramin Tavakoli Kolagari, Thomas von der Maßen, Andreas Wolfram: Report of the GI Work Group "Requirements Management Tools for Product Line Engineering"
2006-15	Sebastian Ullrich, Jakob T. Valvoda, Torsten Kuhlen: Utilizing optical sensors from mice for new input devices
2006-16	Rafael Ballagas, Jan Borchers: Selexels: a Conceptual Framework for Pointing Devices with Low Expressiveness
2006-17	Eric Lee, Henning Kiel, Jan Borchers: Scrolling Through Time: Improving Interfaces for Searching and Navigating Continuous Audio Timelines
2007-01 *	Fachgruppe Informatik: Jahresbericht 2006
2007-02	Carsten Fuhs, Jürgen Giesl, Aart Middeldorp, Peter Schneider-Kamp, René Thiemann, and Harald Zankl: SAT Solving for Termination Analysis with Polynomial Interpretations

* These reports are only available as a printed version.
Please contact `biblio@informatik.rwth-aachen.de` to obtain copies.