Polynomial Interpretations with Negative Coefficients

Nao Hirokawa and Aart Middeldorp

Institute of Computer Science
University of Innsbruck
6020 Innsbruck, Austria
{nao.hirokawa,aart.middeldorp}@uibk.ac.at

Abstract. Polynomial interpretations are a useful technique for proving termination of term rewrite systems. We show how polynomial interpretations with negative coefficients, like $x - 1$ for a unary function symbol or $x - y$ for a binary function symbol, can be used to extend the class of rewrite systems that can be automatically proved terminating.

1 Introduction

This paper is concerned with automatically proving termination of first-order rewrite systems by means of polynomial interpretations. In the classical approach, which goes back to Lankford [16], one associates with every n-ary function symbol f a polynomial P_f over the natural numbers in n indeterminates, which induces a mapping from terms to polynomials in the obvious way. Then one shows that for every rewrite rule $l \to r$ the polynomial P_l associated with the left-hand side l is strictly greater than the polynomial P_r associated with the right-hand side r, i.e., $P_l - P_r > 0$ for all values of the indeterminates. In order to conclude termination, the polynomial P_f associated with an n-ary function symbol f must be strictly monotone in all n indeterminates. Techniques for finding appropriate polynomials as well as approximating (in general undecidable) polynomial inequalities $P > 0$ are described in several papers (e.g. [4, 6, 9, 15, 19]). As a simple example, consider the rewrite rules

$$\begin{aligned} x + \mathsf{0} &\to x & x \times \mathsf{0} &\to \mathsf{0} \\ x + \mathsf{s}(y) &\to \mathsf{s}(x + y) & x \times \mathsf{s}(y) &\to (x \times y) + x \end{aligned}$$

Termination can be shown by the strictly monotone polynomial interpretations

$$\times_{\mathbb{N}}(x, y) = 2xy + y + 1 \qquad +_{\mathbb{N}}(x, y) = x + 2y \qquad \mathsf{s}_{\mathbb{N}}(x) = x + 1 \qquad \mathsf{0}_{\mathbb{N}} = 1$$

over the natural numbers:

$$\begin{aligned} x + 2 &> x & 2x + 2 &> 1 \\ x + 2y + 2 &> x + 2y + 1 & 2xy + 2x + y + 2 &> 2xy + 2x + y + 1 \end{aligned}$$

Compared to other classical methods for proving termination of rewrite systems (like recursive path orders and Knuth-Bendix orders), polynomial interpretations are rather weak. Numerous natural examples cannot be handled because of the strict monotonicity requirement which precludes interpretations like $x+1$ for binary function symbols. In connection with the dependency pair method of Arts and Giesl [1], polynomial interpretations become much more useful because strict monotonicity is no longer required; weak monotonicity is sufficient and hence $x+1$ or even 0 as interpretation of a binary function symbol causes no problems. Monotonicity is typically guaranteed by demanding that all coefficients are positive.

In this paper we go a step further. We show that polynomial interpretations over the integers with negative coefficients like $x-1$ and $x-y+1$ can also be used for termination proofs. To make the discussion more concrete, let us consider a somewhat artificial example: the recursive definition

$$f(x) = \text{if } x > 0 \text{ then } f(f(x-1)) + 1 \text{ else } 0$$

from [8]. It computes the identity function over the natural numbers. Termination of the rewrite system

$$1\colon\ \mathsf{f}(\mathsf{s}(x)) \to \mathsf{s}(\mathsf{f}(\mathsf{f}(\mathsf{p}(\mathsf{s}(x))))) \qquad 2\colon\ \mathsf{f}(\mathsf{0}) \to \mathsf{0} \qquad 3\colon\ \mathsf{p}(\mathsf{s}(x)) \to x$$

obtained after the obvious translation is not easily proved. The (manual) proof in [8] relies on forward closures whereas powerful automatic tools like AProVE [11] and CiME [5] that incorporate both polynomial interpretations and the dependency pair method fail to prove termination. There are three dependency pairs (here $\mathsf{f}^\sharp$ and $\mathsf{p}^\sharp$ are new function symbols):

$$4\colon\ \mathsf{f}^\sharp(\mathsf{s}(x)) \to \mathsf{f}^\sharp(\mathsf{f}(\mathsf{p}(\mathsf{s}(x)))) \quad 5\colon\ \mathsf{f}^\sharp(\mathsf{s}(x)) \to \mathsf{f}^\sharp(\mathsf{p}(\mathsf{s}(x))) \quad 6\colon\ \mathsf{f}^\sharp(\mathsf{s}(x)) \to \mathsf{p}^\sharp(\mathsf{s}(x))$$

By taking the *natural* polynomial interpretation

$$\mathsf{f}_{\mathbb{Z}}(x) = \mathsf{f}^\sharp_{\mathbb{Z}}(x) = x \qquad \mathsf{s}_{\mathbb{Z}}(x) = x+1 \qquad \mathsf{0}_{\mathbb{Z}} = 0 \qquad \mathsf{p}_{\mathbb{Z}}(x) = \mathsf{p}^\sharp_{\mathbb{Z}}(x) = x-1$$

over the integers, the rule and dependency pair constraints reduce to the following inequalities:

$$\begin{array}{llllll} 1\colon & x+1 \geqslant x+1 & 3\colon & x \geqslant x & 5\colon & x+1 > x \\ 2\colon & 0 \geqslant 0 & 4\colon & x+1 > x & 6\colon & x+1 > x \end{array}$$

These constraints are obviously satisfied. The question is whether we are allowed to conclude termination at this point. We will argue that the answer is affirmative and, moreover, that the search for appropriate natural polynomial interpretations can be efficiently implemented.

The approach described in this paper is inspired by the combination of the general path order and forward closures [8] as well as semantic labelling [24]. Concerning related work, Lucas [17, 18] considers polynomials with *real* coefficients for automatically proving termination of (context-sensitive) rewriting systems. He solves the problem of well-foundedness by replacing the standard order on $\mathbb{R}$ with $>_\delta$ for some fixed positive $\delta \in \mathbb{R}$: $x >_\delta y$ if and only if $x - y \geqslant \delta$. In addition, he demands that interpretations are uniformly bounded from below (i.e., there exists an $m \in \mathbb{R}$ such that $f_{\mathbb{R}}(x_1, \dots, x_n) \geqslant m$ for all function symbols f and $x_1, \dots, x_n \geqslant m$). The latter requirement entails that interpretations like $x - 1$ or $x - y + 1$ cannot be handled.

The remainder of the paper is organized as follows. In Section 3 we discuss polynomial interpretations with negative constants. Polynomial interpretations with negative coefficients require a different approach, which is detailed in Section 4. In Section 5 we discuss briefly how to find suitable polynomial interpretations automatically and we report on the many experiments that we performed.

2 Preliminaries

We assume familiarity with the basics of term rewriting [3, 21] and with the dependency pair method [1] for proving (innermost) termination. In the latter method a term rewrite system (TRS for short) is transformed into a collection of ordering constraints of the form $l \gtrsim r$ and $l > r$ that need to be solved in order to conclude termination. Solutions $(\gtrsim, >)$ must be *reduction pairs* which consist of a rewrite preorder $\gtrsim$ (i.e., a transitive and reflexive relation which is closed under contexts and substitutions) on terms and a compatible well-founded order $>$ which is closed under substitutions. Compatibility means that the inclusion $\gtrsim \cdot > \subseteq >$ or the inclusion $> \cdot \gtrsim \subseteq >$ holds. (Here $\cdot$ denotes relational composition.)

A general semantic construction of reduction pairs, which covers traditional polynomial interpretations, is based on the concept of algebra. If we equip the carrier A of an $\mathcal{F}$-algebra $\mathcal{A} = (A, \{f_{\mathcal{A}}\}_{f \in \mathcal{F}})$ with a well-founded order $>$ such that every interpretation function is weakly monotone in all arguments (i.e., $f_{\mathcal{A}}(x_1, \dots, x_n) \geqslant f_{\mathcal{A}}(y_1, \dots, y_n)$ whenever $x_i \geqslant y_i$ for all $1 \leqslant i \leqslant n$, for every n-ary function symbol $f \in \mathcal{F}$) then $(\geqslant_{\mathcal{A}}, >_{\mathcal{A}})$ is a reduction pair. Here the relations $\geqslant_{\mathcal{A}}$ and $>_{\mathcal{A}}$ are defined as follows: $s \geqslant_{\mathcal{A}} t$ if $[\alpha]_{\mathcal{A}}(s) \geqslant [\alpha]_{\mathcal{A}}(t)$ and $s >_{\mathcal{A}} t$ if $[\alpha]_{\mathcal{A}}(s) > [\alpha]_{\mathcal{A}}(t)$, for all assignments α of elements of A to the variables in s and t ($[\alpha]_{\mathcal{A}}(\cdot)$ denotes the usual evaluation function associated with the algebra $\mathcal{A}$). In general, the relation $>_{\mathcal{A}}$ is not closed under contexts, $\geqslant_{\mathcal{A}}$ is a preorder but not a partial order, and $>_{\mathcal{A}}$ is not the strict part of $\geqslant_{\mathcal{A}}$. Compatibility holds because of the identity $\geqslant_{\mathcal{A}} \cdot >_{\mathcal{A}} \, = \, >_{\mathcal{A}}$. We write $s =_{\mathcal{A}} t$ if $[\alpha]_{\mathcal{A}}(s) = [\alpha]_{\mathcal{A}}(t)$ for all assignments α. We say that $\mathcal{A}$ is a model for a TRS $\mathcal{R}$ if $l =_{\mathcal{A}} r$ for all rewrite rules in $\mathcal{R}$.

In this paper we use the following results from [10] concerning dependency pairs.

Theorem 1. *A TRS $\mathcal{R}$ is terminating if for every cycle $\mathcal{C}$ in its dependency graph there exists a reduction pair $(\gtrsim, >)$ such that $\mathcal{R} \subseteq \gtrsim$, $\mathcal{C} \subseteq \gtrsim \cup >$, and $\mathcal{C} \cap > \neq \varnothing$.* □

Theorem 2. *A TRS $\mathcal{R}$ is innermost terminating if for every cycle $\mathcal{C}$ in its innermost dependency graph there exists a reduction pair $(\gtrsim, >)$ such that $\mathcal{U}(\mathcal{C}) \subseteq \gtrsim$, $\mathcal{C} \subseteq \gtrsim \cup >$, and $\mathcal{C} \cap > \neq \varnothing$.* □

3 Negative Constants

3.1 Theoretical Framework

When using polynomial interpretations with negative constants like in the example of the introduction, the first challenge we face is that the standard order $>$ on $\mathbb{Z}$ is not well-founded. Restricting the domain to the set $\mathbb{N}$ of natural numbers makes an interpretation like $\mathsf{p}_{\mathbb{Z}}(x) = x - 1$ ill-defined. Dershowitz and Hoot observe in [8] that if all (instantiated) subterms in the rules of the TRS are interpreted as non-negative integers, such interpretations can work correctly. Following their observation, we propose to modify the interpretation of p to $\mathsf{p}_{\mathbb{N}}(x) = \max\{0, x - 1\}$.

Definition 3. *Let $\mathcal{F}$ be a signature and let $(\mathbb{Z}, \{f_{\mathbb{Z}}\}_{f \in \mathcal{F}})$ be an $\mathcal{F}$-algebra such that every interpretation function $f_{\mathbb{Z}}$ is weakly monotone in all its arguments. The interpretation functions of the* induced *algebra $(\mathbb{N}, \{f_{\mathbb{N}}\}_{f \in \mathcal{F}})$ are defined as follows: $f_{\mathbb{N}}(x_1, \ldots, x_n) = \max\{0, f_{\mathbb{Z}}(x_1, \ldots, x_n)\}$ for all $x_1, \ldots, x_n \in \mathbb{N}$.*

With respect to the interpretations in the introduction we obtain $\mathsf{s}_{\mathbb{N}}(\mathsf{p}_{\mathbb{N}}(x)) = \max\{0, \max\{0, x-1\} + 1\} = \max\{0, x-1\} + 1$, $\mathsf{p}_{\mathbb{N}}(\mathsf{0}_{\mathbb{N}}) = \max\{0, 0\} = 0$, and $\mathsf{p}_{\mathbb{N}}(\mathsf{s}_{\mathbb{N}}(x)) = \max\{0, \max\{0, x+1\} - 1\} = x$.

Lemma 4. *If $(\mathbb{Z}, \{f_{\mathbb{Z}}\}_{f \in \mathcal{F}})$ is an $\mathcal{F}$-algebra with weakly monotone interpretations then $(\geqslant_{\mathbb{N}}, >_{\mathbb{N}})$ is a reduction pair.*

Proof. It is easy to show that the interpretation functions of the induced algebra are weakly monotone in all arguments. Routine arguments reveal that the relation $>_{\mathbb{N}}$ is a well-founded order which is closed under substitutions and that $\geqslant_{\mathbb{N}}$ is a preorder closed under contexts and substitutions. Moreover, the identity $>_{\mathbb{N}} \cdot \geqslant_{\mathbb{N}} = >_{\mathbb{N}}$ holds. Hence $(\geqslant_{\mathbb{N}}, >_{\mathbb{N}})$ is a reduction pair. $\square$

It is interesting to remark that unlike usual polynomial interpretations, the relation $>_{\mathbb{N}}$ does not have the (weak) subterm property. For instance, with respect to the interpretations in the example of the introduction, we have $\mathsf{s}(\mathsf{0}) >_{\mathbb{N}} \mathsf{p}(\mathsf{s}(\mathsf{0}))$ and not $\mathsf{p}(\mathsf{s}(\mathsf{0})) >_{\mathbb{N}} \mathsf{p}(\mathsf{0})$.

In recent modular refinements of the dependency pair method [23, 13, 22] suitable reduction pairs $(\gtrsim, >)$ have to satisfy the additional property of $\mathcal{C}_{\mathcal{E}}$-*compatibility*: $\gtrsim$ must orient the rules of the TRS $\mathcal{C}_{\mathcal{E}}$ consisting of the two rewrite rules $\mathsf{cons}(x, y) \to x$ and $\mathsf{cons}(x, y) \to y$, where cons is a fresh function symbol, from left to right. This is not a problem because we can simply define $\mathsf{cons}_{\mathbb{N}}(x, y) = \max\{x, y\}$. In this way we obtain a reduction pair $(\succsim, \succ)$ on terms over the original signature extended with cons such that $\gtrsim \cup\, \mathcal{C}_{\mathcal{E}} \subseteq\, \succsim$ and $> \subseteq \succ$.

Example 5. Consider the TRS consisting of the following rewrite rules:

$$
\begin{array}{rlrl}
1: & \mathsf{half}(\mathsf{0}) \to \mathsf{0} & 4: & \mathsf{bits}(\mathsf{0}) \to \mathsf{0} \\
2: & \mathsf{half}(\mathsf{s}(\mathsf{0})) \to \mathsf{0} & 5: & \mathsf{bits}(\mathsf{s}(x)) \to \mathsf{s}(\mathsf{bits}(\mathsf{half}(\mathsf{s}(x)))) \\
3: & \mathsf{half}(\mathsf{s}(\mathsf{s}(x))) \to \mathsf{s}(\mathsf{half}(x)) & &
\end{array}
$$

The function $\mathsf{half}(x)$ computes $\lceil \frac{x}{2} \rceil$ and $\mathsf{bits}(x)$ computes the number of bits that are needed to represent all numbers less than or equal to x. Termination of this TRS is proved in [2] by using the dependency pair method together with the narrowing refinement. There are three dependency pairs:

$$\begin{aligned} 6\colon && \mathsf{half}^\sharp(\mathsf{s}(\mathsf{s}(x))) &\to \mathsf{half}^\sharp(x) \\ 7\colon && \mathsf{bits}^\sharp(\mathsf{s}(x)) &\to \mathsf{bits}^\sharp(\mathsf{half}(\mathsf{s}(x))) \\ 8\colon && \mathsf{bits}^\sharp(\mathsf{s}(x)) &\to \mathsf{half}^\sharp(\mathsf{s}(x)) \end{aligned}$$

By taking the interpretations $\mathsf{0}_\mathbb{Z} = 0$, $\mathsf{half}_\mathbb{Z}(x) = x - 1$, $\mathsf{bits}_\mathbb{Z}(x) = \mathsf{half}^\sharp_\mathbb{Z}(x) = x$, and $\mathsf{s}_\mathbb{Z}(x) = \mathsf{bits}^\sharp_\mathbb{Z}(x) = x + 1$, we obtain the following constraints over $\mathbb{N}$:

$$\begin{aligned} 1\colon && 0 &\geqslant 0 & \qquad 5\colon && x+1 &\geqslant x+1 \\ 2\colon && 0 &\geqslant 0 & 6\colon && x+2 &> x \\ 3\colon && x+1 &\geqslant \max\{0, x-1\}+1 & 7\colon && x+2 &> x+1 \\ 4\colon && 0 &\geqslant 0 & 8\colon && x+2 &> x+1 \end{aligned}$$

These constraints are satisfied, so the TRS is terminating, but how can an inequality like $x + 1 \geqslant \max\{0, x - 1\} + 1$ be verified automatically?

3.2 Towards Automation

Because the inequalities resulting from interpretations with negative constants may contain the max operator, we cannot use standard techniques for comparing polynomial expressions. In order to avoid reasoning by case analysis ($x - 1 > 0$ or $x - 1 \leqslant 0$ for constraint 3 in Example 5), we approximate the evaluation function of the induced algebra.

Definition 6. *Given a polynomial P with coefficients in $\mathbb{Z}$, we denote the constant part by $c(P)$ and the non-constant part $P - c(P)$ by $n(P)$. Let $(\mathbb{Z}, \{f_\mathbb{Z}\}_{f \in \mathcal{F}})$ be an $\mathcal{F}$-algebra such that every $f_\mathbb{Z}$ is a weakly monotone polynomial. With every term t we associate polynomials $P_{\mathit{left}}(t)$ and $P_{\mathit{right}}(t)$ with coefficients in $\mathbb{Z}$ and variables in t as indeterminates:*

$$P_{\mathit{left}}(t) = \begin{cases} t & \text{if } t \text{ is a variable} \\ 0 & \text{if } t = f(t_1, \ldots, t_n),\ n(P_1) = 0, \text{ and } c(P_1) < 0 \\ P_1 & \text{otherwise} \end{cases}$$

where $P_1 = f_\mathbb{Z}(P_{\mathit{left}}(t_1), \ldots, P_{\mathit{left}}(t_n))$ and

$$P_{\mathit{right}}(t) = \begin{cases} t & \text{if } t \text{ is a variable} \\ n(P_2) & \text{if } t = f(t_1, \ldots, t_n) \text{ and } c(P_2) < 0 \\ P_2 & \text{otherwise} \end{cases}$$

where $P_2 = f_\mathbb{Z}(P_{\mathit{right}}(t_1), \ldots, P_{\mathit{right}}(t_n))$. Let $\alpha\colon \mathcal{V} \to \mathbb{N}$ be an assignment. The result of evaluating $P_{\mathit{left}}(t)$ and $P_{\mathit{right}}(t)$ under α is denoted by $[\alpha]^l_\mathbb{Z}(t)$ and $[\alpha]^r_\mathbb{Z}(t)$. The result of evaluating a polynomial P under α is denoted by $\alpha(P)$.

According the following lemma, $P_{\mathit{left}}(t)$ is a lower bound and $P_{\mathit{right}}(t)$ is an upper bound of the interpretation of t in the induced algebra.

Lemma 7. *Let $(\mathbb{Z}, \{f_{\mathbb{Z}}\}_{f \in \mathcal{F}})$ be an $\mathcal{F}$-algebra such that every $f_{\mathbb{Z}}$ is a weakly monotone polynomial. Let t be a term. For every assignment $\alpha \colon \mathcal{V} \to \mathbb{N}$ we have $[\alpha]^r_{\mathbb{Z}}(t) \geqslant [\alpha]_{\mathbb{N}}(t) \geqslant [\alpha]^l_{\mathbb{Z}}(t)$.*

Proof. By induction on the structure of t. If $t \in \mathcal{V}$ then $[\alpha]^r_{\mathbb{Z}}(t) = [\alpha]^l_{\mathbb{Z}}(t) = \alpha(t) = [\alpha]_{\mathbb{N}}(t)$. Suppose $t = f(t_1, \dots, t_n)$. According to the induction hypothesis, $[\alpha]^r_{\mathbb{Z}}(t_i) \geqslant [\alpha]_{\mathbb{N}}(t_i) \geqslant [\alpha]^l_{\mathbb{Z}}(t_i)$ for all i. Since $f_{\mathbb{Z}}$ is weakly monotone,

$$f_{\mathbb{Z}}([\alpha]^r_{\mathbb{Z}}(t_1), \dots, [\alpha]^r_{\mathbb{Z}}(t_n)) \geqslant f_{\mathbb{Z}}([\alpha]_{\mathbb{N}}(t_1), \dots, [\alpha]_{\mathbb{N}}(t_n)) \geqslant f_{\mathbb{Z}}([\alpha]^l_{\mathbb{Z}}(t_1), \dots, [\alpha]^l_{\mathbb{Z}}(t_n))$$

By applying the weakly monotone function $\max\{0, \cdot\}$ we obtain $\max\{0, \alpha(P_2)\} \geqslant [\alpha]_{\mathbb{N}}(t) \geqslant \max\{0, \alpha(P_1)\}$ where $P_1 = f_{\mathbb{Z}}(P_{\mathit{left}}(t_1), \dots, P_{\mathit{left}}(t_n))$ and $P_2 = f_{\mathbb{Z}}(P_{\mathit{right}}(t_1), \dots, P_{\mathit{right}}(t_n))$. We have

$$[\alpha]^l_{\mathbb{Z}}(t) = \begin{cases} 0 & \text{if } n(P_1) = 0 \text{ and } c(P_1) < 0 \\ \alpha(P_1) & \text{otherwise} \end{cases}$$

and thus $[\alpha]^l_{\mathbb{Z}}(t) \leqslant \max\{0, \alpha(P_1)\}$. Likewise,

$$[\alpha]^r_{\mathbb{Z}}(t) = \begin{cases} \alpha(n(P_2)) & \text{if } c(P_2) < 0 \\ \alpha(P_2) & \text{otherwise} \end{cases}$$

In the former case, $\alpha(n(P_2)) = \alpha(P_2) - c(P_2) > \alpha(P_2)$ and $\alpha(n(P_2)) \geqslant 0$. In the latter case $\alpha(P_2) \geqslant 0$. So in both cases we have $[\alpha]^r_{\mathbb{Z}}(t) \geqslant \max\{0, \alpha(P_2)\}$. Hence we obtain the desired inequalities. $\square$

Corollary 8. *Let $(\mathbb{Z}, \{f_{\mathbb{Z}}\}_{f \in \mathcal{F}})$ be an $\mathcal{F}$-algebra such that every $f_{\mathbb{Z}}$ is a weakly monotone polynomial. Let s and t be terms. If $P_{\mathit{left}}(s) - P_{\mathit{right}}(t) > 0$ then $s >_{\mathbb{N}} t$. If $P_{\mathit{left}}(s) - P_{\mathit{right}}(t) \geqslant 0$ then $s \geqslant_{\mathbb{N}} t$.* $\square$

Example 9. Consider again the TRS of Example 5. By applying P_{left} to the left-hand sides and P_{right} to the right-hand sides of the rewrite rules and the dependency pairs, the following ordering constraints are obtained:

$1\colon\ 0 \geqslant 0$	$3\colon\ x+1 \geqslant x+1$	$5\colon\ x+1 \geqslant x+1$	$7\colon\ x+2 > x+1$
$2\colon\ 0 \geqslant 0$	$4\colon\ 0 \geqslant 0$	$6\colon\ x+2 > x$	$8\colon\ x+2 > x+1$

The only difference with the constraints in Example 5 is the interpretation of the term $\mathsf{s}(\mathsf{half}(x))$ on the right-hand side of rule 3. We have $P_{\mathit{right}}(\mathsf{half}(x)) = n(x-1) = x$ and thus $P_{\mathit{right}}(\mathsf{s}(\mathsf{half}(x))) = x+1$. Although $x+1$ is less precise than $\max\{0, x-1\} + 1$, it is accurate enough to solve the ordering constraint resulting from rule 3.

So once the interpretations $f_{\mathbb{Z}}$ are determined, we transform a rule $l \to r$ into the polynomial $P_{\mathit{left}}(l) - P_{\mathit{right}}(r)$. Standard techniques can then be used to test whether this polynomial is positive (or non-negative) for all values in $\mathbb{N}$ for the variables. The remaining question is how to find suitable interpretations for the function symbols. This problem will be discussed in Section 5.

4 Negative Coefficients

Let us start with an example which shows that negative coefficients in polynomial interpretations can be useful.

Example 10. Consider the following variation of a TRS in [2]:

$$
\begin{array}{rrcrl}
1: & 0 \leqslant x \to \mathsf{true} & & 7: & x - 0 \to x \\
2: & \mathsf{s}(x) \leqslant 0 \to \mathsf{false} & & 8: & \mathsf{s}(x) - \mathsf{s}(y) \to x - y \\
3: & \mathsf{s}(x) \leqslant \mathsf{s}(y) \to x \leqslant y & & 9: & \mathsf{if}(\mathsf{true}, x, y) \to x \\
4: & \mathsf{mod}(0, \mathsf{s}(y)) \to 0 & & 10: & \mathsf{if}(\mathsf{false}, x, y) \to y \\
5: & \mathsf{mod}(\mathsf{s}(x), 0) \to 0 & & & \\
6: & \multicolumn{4}{l}{\mathsf{mod}(\mathsf{s}(x), \mathsf{s}(y)) \to \mathsf{if}(y \leqslant x, \mathsf{mod}(\mathsf{s}(x) - \mathsf{s}(y), \mathsf{s}(y)), \mathsf{s}(x))}
\end{array}
$$

There are 6 dependency pairs:

$$
\begin{array}{rl}
11: & \mathsf{s}(x) \leqslant^\sharp \mathsf{s}(y) \to x \leqslant^\sharp y \\
12: & \mathsf{s}(x) -^\sharp \mathsf{s}(y) \to x -^\sharp y \\
13: & \mathsf{mod}^\sharp(\mathsf{s}(x), \mathsf{s}(y)) \to \mathsf{if}^\sharp(y \leqslant x, \mathsf{mod}(\mathsf{s}(x) - \mathsf{s}(y), \mathsf{s}(y)), \mathsf{s}(x)) \\
14: & \mathsf{mod}^\sharp(\mathsf{s}(x), \mathsf{s}(y)) \to y \leqslant^\sharp x \\
15: & \mathsf{mod}^\sharp(\mathsf{s}(x), \mathsf{s}(y)) \to \mathsf{mod}^\sharp(\mathsf{s}(x) - \mathsf{s}(y), \mathsf{s}(y)) \\
16: & \mathsf{mod}^\sharp(\mathsf{s}(x), \mathsf{s}(y)) \to \mathsf{s}(x) -^\sharp \mathsf{s}(y)
\end{array}
$$

Since the TRS is non-overlapping, it is sufficient to prove innermost termination. The problematic cycle in the (innermost) dependency graph is $\mathcal{C} = \{15\}$. The usable rewrite rules for this cycle are $\mathcal{U}(\mathcal{C}) = \{7, 8\}$. We need to find a reduction pair $(\gtrsim, >)$ such that rules 4 and 5 are weakly decreasing (i.e., compatible with $\gtrsim$) and dependency pair 15 is strictly decreasing (with respect to $>$). The only way to achieve the latter is by using the observation that $\mathsf{s}(x)$ is semantically greater than the syntactically larger term $\mathsf{s}(x) - \mathsf{s}(y)$. If we take the natural interpretation $-_\mathbb{Z}(x, y) = x - y$, $\mathsf{s}_\mathbb{Z}(x) = x - 1$, and $0_\mathbb{Z} = 0$, together with $\mathsf{mod}^\sharp_\mathbb{Z}(x, y) = x$ then we obtain the following ordering constraints over the natural numbers:

$$7\colon\ x \geqslant x \quad 8\colon\ \max\{0, x - y\} \geqslant \max\{0, x - y\} \quad 15\colon\ x + 1 > \max\{0, x - y\}$$

4.1 Theoretical Framework

The constraints in the above example are obviously satisfied, but are we allowed to use an interpretation like $-_\mathbb{Z}(x, y) = x - y$ in (innermost) termination proofs? The answer appears to be negative because Lemma 4 no longer holds. Because the induced interpretation $-_\mathbb{N}(x, y) = \max\{0, x - y\}$ is not weakly monotone in its second argument, the order $\geqslant_\mathbb{N}$ of the induced algebra is not closed under contexts, so if $s \geqslant_\mathbb{N} t$ then it may happen that $C[s] \leqslant_\mathbb{N} C[t]$. Consequently, we do not obtain a reduction pair. However, if we have $s =_\mathbb{N} t$ rather than

$s \geqslant_{\mathbb{N}} t$, closure under contexts is obtained for free. So we could take $(=_{\mathbb{N}}, >_{\mathbb{N}})$ as reduction pair. This works fine in the above example because the induced algebra is a model of the set of usable rules $\{7, 8\}$ and $>_{\mathbb{N}}$ orients dependency pair 15. However, requiring that all dependency pairs in a cycle are compatible with $=_{\mathbb{N}} \cup >_{\mathbb{N}}$ is rather restrictive because dependency pairs that are transformed into a polynomial constraint of the form $x^2 \geqslant x$ or $x + 2y \geqslant x + y$ cannot be handled. So we will allow $\geqslant_{\mathbb{N}}$ for the orientation of dependency pairs in a cycle $\mathcal{C}$ but insist that at least one dependency pair in $\mathcal{C}$ is compatible with $>_{\mathbb{N}}$. (Note that the relation $=_{\mathbb{N}} \cup >_{\mathbb{N}}$ is properly contained in $\geqslant_{\mathbb{N}}$.) The theorems below state the soundness of this approach in a more abstract setting. The proofs are straightforward modifications from the ones in [13]. The phrase "there are no minimal $\mathcal{C}$-rewrite sequences" intuitively means that if a TRS $\mathcal{R}$ is non-terminating then this is due to a different cycle of the dependency graph.

Theorem 11. *Let $\mathcal{R}$ be a TRS and let $\mathcal{C}$ be a cycle in its dependency graph. If there exists an algebra $\mathcal{A}$ equipped with a well-founded order $>$ such that $\mathcal{R} \subseteq =_{\mathcal{A}}$, $\mathcal{C} \subseteq \geqslant_{\mathcal{A}}$, and $\mathcal{C} \cap >_{\mathcal{A}} \neq \varnothing$ then there are no minimal $\mathcal{C}$-rewrite sequences.* □

In other words, when proving termination, a cycle $\mathcal{C}$ of the dependency graph can be ignored if the conditions of Theorem 11 are satisfied. A similar statement holds for innermost termination.

Theorem 12. *Let $\mathcal{R}$ be a TRS and let $\mathcal{C}$ be a cycle in its innermost dependency graph. If there exists an algebra $\mathcal{A}$ equipped with a well-founded order $>$ such that $\mathcal{U}(\mathcal{C}) \subseteq =_{\mathcal{A}}$, $\mathcal{C} \subseteq \geqslant_{\mathcal{A}}$, and $\mathcal{C} \cap >_{\mathcal{A}} \neq \varnothing$ then there are no minimal innermost $\mathcal{C}$-rewrite sequences.* □

The difference with Theorem 11 is the use of the *innermost* dependency graph and, more importantly, the replacement of the set $\mathcal{R}$ of all rewrite rules by the set $\mathcal{U}(\mathcal{C})$ of *usable rules* for $\mathcal{C}$, which in general is a much smaller set. Very recently, it has been proved [13, 22] that the usable rules criterion can also be used for termination, provided the employed reduction pair is $\mathcal{C}_{\mathcal{E}}$-compatible. However, replacing $\mathcal{R}$ by $\mathcal{U}(\mathcal{C})$ in Theorem 11 would be unsound. The reason is that the TRS $\mathcal{C}_{\mathcal{E}}$ admits no non-trivial models.

Example 13. Consider the following non-terminating TRS $\mathcal{R}$:

$$1\colon\ \mathsf{h}(\mathsf{f}(\mathsf{a}, \mathsf{b}, x)) \to \mathsf{h}(\mathsf{f}(x, x, x)) \qquad 2\colon\ \mathsf{g}(x, y) \to x \qquad 3\colon\ \mathsf{g}(x, y) \to y$$

The only dependency pair $\mathsf{h}^\sharp(\mathsf{f}(\mathsf{a}, \mathsf{b}, x)) \to \mathsf{h}^\sharp(\mathsf{f}(x, x, x))$ forms a cycle in the dependency graph. There are no usable rules. If we take the polynomial interpretation $\mathsf{a}_{\mathbb{Z}} = 1$, $\mathsf{b}_{\mathbb{Z}} = 0$, $\mathsf{f}_{\mathbb{Z}}(x, y, z) = x - y$, and $\mathsf{h}^\sharp_{\mathbb{Z}}(x) = x$ then the dependency pair is transformed into $1 > 0$. Note that it is not possible to extend the interpretation to a model for $\mathcal{R}$. Choosing $\mathsf{h}_{\mathbb{Z}}(x) = 0$ will take care of rule 1, but there is no interpretation $\mathsf{g}_{\mathbb{Z}}$ such that $\max\{0, \mathsf{g}_{\mathbb{Z}}(x, y)\} = x$ and $\max\{0, \mathsf{g}_{\mathbb{Z}}(x, y)\} = y$ for all natural numbers x and y.

4.2 Towards Automation

How do we verify a constraint like $x + 1 > \max\{0, x - y\}$? The approach that we developed in Section 3.2 for dealing with negative constants is not applicable because Lemma 7 relies essentially on weak monotonicity of the polynomial interpretations.

Let $\mathcal{P}_{\geqslant 0}$ be a subset of the set of polynomials P with integer coefficients such that $\alpha(P) \geqslant 0$ for all $\alpha\colon \mathcal{V} \to \mathbb{N}$ for which membership is decidable. For instance, $\mathcal{P}_{\geqslant 0}$ could be the set of polynomials without negative coefficients. We define $\mathcal{P}_{<0}$ in the same way.

Definition 14. *Let $(\mathbb{Z}, \{f_{\mathbb{Z}}\}_{f \in \mathcal{F}})$ be an algebra. With every term t we associate a polynomial $Q(t)$ as follows:*

$$Q(t) = \begin{cases} t & \text{if } t \text{ is a variable} \\ P & \text{if } t = f(t_1, \dots, t_n) \text{ and } P \in \mathcal{P}_{\geqslant 0} \\ 0 & \text{if } t = f(t_1, \dots, t_n) \text{ and } P \in \mathcal{P}_{<0} \\ v(P) & \text{otherwise} \end{cases}$$

where $P = f_{\mathbb{Z}}(Q(t_1), \dots, Q(t_n))$. In the last clause $v(P)$ denotes a fresh abstract *variable that we uniquely associate with P.*

There are two kinds of indeterminates in $Q(t)$: ordinary variables occurring in t and abstract variables. The intuitive meaning of an abstract variable $v(P)$ is $\max\{0, P\}$. The latter quantity is always non-negative, like an ordinary variable ranging over the natural numbers, but from $v(P)$ we can extract the original polynomial P and this information may be crucial for a comparison between two polynomial expressions to succeed. Note that the polynomial P associated with an abstract variable $v(P)$ may contain other abstract variables. However, because $v(P)$ is different from previously selected abstract variables, there are no spurious loops like $P_1 = v(x - v(P_2))$ and $P_2 = v(x - v(P_1))$.

The reason for using $\mathcal{P}_{\geqslant 0}$ and $\mathcal{P}_{<0}$ in the above definition is to make our approach independent of the particular method that is used to test non-negativeness or negativeness of polynomials.

Definition 15. *With every assignment $\alpha\colon \mathcal{V} \to \mathbb{N}$ we associate an assignment $\alpha^*\colon \mathcal{V} \to \mathbb{N}$ defined as follows:*

$$\alpha^*(x) = \begin{cases} \max\{0, \alpha^*(P)\} & \text{if } x \text{ is an abstract variable } v(P) \\ \alpha(x) & \text{otherwise} \end{cases}$$

The above definition is recursive because P may contain abstract variables. However, since $v(P)$ is different from previously selected abstract variables, the recursion terminates and it follows that α^* is well-defined.

Theorem 16. *Let $(\mathbb{Z}, \{f_{\mathbb{Z}}\}_{f \in \mathcal{F}})$ be an algebra such that every $f_{\mathbb{Z}}$ is a polynomial. Let t be a term. For every assignment α we have $[\alpha]_{\mathbb{N}}(t) = \alpha^*(Q(t))$.*

Proof. We show that $[\alpha]_{\mathbb{N}}(t) = \alpha^*(Q(t))$ by induction on t. If t is a variable then $[\alpha]_{\mathbb{N}}(t) = \alpha(t) = \alpha^*(t) = \alpha^*(Q(t))$. Suppose $t = f(t_1, \ldots, t_n)$. Let $P = f_{\mathbb{Z}}(Q(t_1), \ldots, Q(t_n))$. The induction hypothesis yields $[\alpha]_{\mathbb{N}}(t_i) = \alpha^*(Q(t_i))$ for all i and thus

$$\begin{aligned}[\alpha]_{\mathbb{N}}(t) &= f_{\mathbb{N}}(\alpha^*(Q(t_1)), \ldots, \alpha^*(Q(t_n))) \\ &= \max\{0, f_{\mathbb{Z}}(\alpha^*(Q(t_1)), \ldots, \alpha^*(Q(t_n)))\} = \max\{0, \alpha^*(P)\}\end{aligned}$$

We distinguish three cases, corresponding to the definition of $Q(t)$.

– First suppose that $P \in \mathcal{P}_{\geqslant 0}$. This implies that $\alpha^*(P) \geqslant 0$ and thus we have $\max\{0, \alpha^*(P)\} = \alpha^*(P)$. Hence $[\alpha]_{\mathbb{N}}(t) = \alpha^*(P) = \alpha^*(Q(t))$.
– Next suppose that $P \in \mathcal{P}_{<0}$. So $\alpha^*(P) < 0$ and thus $\max\{0, \alpha^*(P)\} = 0$. Hence $[\alpha]_{\mathbb{N}}(t) = 0 = \alpha^*(Q(t))$.
– In the remaining case we do not know the status of P. We have $Q(t) = v(P)$ and thus $\alpha^*(Q(t)) = \max\{0, \alpha^*(P)\}$ which immediately yields the desired identity $[\alpha]_{\mathbb{N}}(t) = \alpha^*(Q(t))$.

$\square$

Corollary 17. *Let $(\mathbb{Z}, \{f_{\mathbb{Z}}\}_{f \in \mathcal{F}})$ be an $\mathcal{F}$-algebra such that every $f_{\mathbb{Z}}$ is a polynomial. Let s and t be terms. If $Q(s) = Q(t)$ then $s =_{\mathbb{N}} t$. If $\alpha^*(Q(s) - Q(t)) > 0$ for all assignments $\alpha\colon \mathcal{V} \to \mathbb{N}$ then $s >_{\mathbb{N}} t$. If $\alpha^*(Q(s) - Q(t)) \geqslant 0$ for all assignments $\alpha\colon \mathcal{V} \to \mathbb{N}$ then $s \geqslant_{\mathbb{N}} t$.* $\square$

Example 18. Consider again dependency pair 15 from Example 10:

$$\mathsf{mod}^\sharp(\mathsf{s}(x), \mathsf{s}(y)) \to \mathsf{mod}^\sharp(\mathsf{s}(x) - \mathsf{s}(y), \mathsf{s}(y))$$

We have $Q(\mathsf{mod}^\sharp(\mathsf{s}(x), \mathsf{s}(y))) = x + 1$ and $Q(\mathsf{mod}^\sharp(\mathsf{s}(x) - \mathsf{s}(y), \mathsf{s}(y))) = v(x - y)$. Since $x + 1 - v(x - y)$ may be negative (when interpreting $v(x - y)$ as a variable), the above corollary cannot be used to conclude that 15 is strictly decreasing. However, if we estimate $v(x - y)$ by x, the non-negative part of $x - y$, then we obtain $x + 1 - x = 1$ which is clearly positive.

Given a polynomial P with coefficients in $\mathbb{Z}$, we denote the non-negative part of P by $N(P)$.

Lemma 19. *Let Q be a polynomial with integer coefficients. Suppose $v(P)$ is an abstract variable that occurs in Q but not in $N(Q)$. If Q' is the polynomial obtained from Q by replacing $v(P)$ with $N(P)$ then $\alpha^*(Q) \geqslant \alpha^*(Q')$ for all assignments $\alpha\colon \mathcal{V} \to \mathbb{N}$.*

Proof. Let $\alpha\colon \mathcal{V} \to \mathbb{N}$ be an arbitrary assignment. In $\alpha^*(Q)$ every occurrence of $v(P)$ is assigned the value $\alpha^*(v(P)) = \max\{0, \alpha^*(P)\}$. We have $\alpha^*(N(P)) \geqslant \alpha^*(P) \geqslant \alpha^*(v(P))$. By assumption, $v(P)$ occurs only in the negative part of Q. Hence Q is (strictly) anti-monotone in $v(P)$ and therefore $\alpha^*(Q) \geqslant \alpha^*(Q')$. $\square$

In order to determine whether $s \geqslant_{\mathbb{N}} t$ (or $s >_{\mathbb{N}} t$) holds, the idea now is to first use standard techniques to test the non-negativeness of $Q = Q(s) - Q(t)$ (i.e., we determine whether $\alpha(Q) \geqslant 0$ for all assignments α by checking whether $Q \in \mathcal{P}_{\geqslant 0}$). If Q is non-negative then we certainly have $\alpha^*(Q) \geqslant 0$ for all assignments α and thus $s \geqslant_{\mathbb{N}} t$ follows from Corollary 17. If non-negativeness cannot be shown then we apply the previous lemma to replace an abstract variable that occurs only in the negative part of Q. The resulting polynomial Q' is tested for non-negativeness. If the test succeeds then for all assignments α we have $\alpha^*(Q') \geqslant 0$ and thus also $\alpha^*(Q) \geqslant 0$ by the previous lemma. According to Corollary 17 this is sufficient to conclude $s \geqslant_{\mathbb{N}} t$. Otherwise we repeat the above process with Q'. The process terminates when there are no more abstract variables left that appear only in the negative part of the current polynomial.

5 Experimental Results

We implemented the techniques described in this paper in the Tyrolean Termination Tool [14]. We tested 219 terminating TRSs and 239 innermost terminating TRSs from three different sources:

– all 89 terminating and 109 innermost terminating TRSs from Arts and Giesl [2],
– all 23 TRSs from Dershowitz [7],
– all 116 terminating TRSs from Steinbach and Kühler [20, Sections 3 and 4].

Nine of these TRSs appear in more than one collection, so the total number is 219 for termination and 239 for innermost termination. In our experiments we use the dependency pair method with the recursive SCC algorithm of [12] for analyzing the dependency graph. The recent modular refinements mentioned after Theorem 12 are also used, except when we try to prove (full) termination with the approach of Section 4 (but when the TRS is non-overlapping we do use modularity since in that case innermost termination guarantees termination). All experiments were performed on a PC equipped with a 2.20 GHz Mobile Intel Pentium 4 Processor - M and 512 MB of memory.

Tables 1 and 2 show the effect of the negative constant method developed in Section 3. In Table 1 we prove termination whereas in Table 1 we prove innermost termination. In the columns labelled N we use the *natural* interpretation for certain function symbols that appear in many example TRSs:

$$\begin{array}{llll} \mathsf{0}_{\mathbb{Z}} = 0 & \mathsf{1}_{\mathbb{Z}} = 1 & \mathsf{2}_{\mathbb{Z}} = 2 & \cdots \\ \mathsf{s}_{\mathbb{Z}}(x) = x + 1 & +_{\mathbb{Z}}(x,y) = x + y & \times_{\mathbb{Z}}(x,y) = xy & \\ \mathsf{p}_{\mathbb{Z}}(x) = x - 1^1 & -_{\mathbb{Z}}(x,y) = x - y^1 & & \end{array}$$

For other function symbols we take *linear* interpretations

$$\mathsf{f}_{\mathbb{Z}}(x_1, \ldots, x_n) = a_1 x_1 + \cdots + a_n x_n + b$$

[1] In Tables 1 and 2 we do not fix the interpretation of p when -1 is not included in the indicated constant range and the natural interpretation of $-$ is not used.

Table 1. Negative constants: termination.

constant	$0,1$			$0,1,2$			$0,1,-1$		
coefficient	$0,1$			$0,1,2$			$0,1$		
interpretation	E	N	NE	E	N	NE	E	N	NE
success	179	158	182	180	159	183	188	164	191
	0.20	0.09	0.08	0.20	0.09	0.09	0.24	0.14	0.12
failure	40	61	37	39	60	36	31	55	28
	0.03	0.02	0.03	1.11	0.15	1.43	0.93	0.43	1.29
timeout	0	0	0	0	0	0	0	0	0
total time	36.41	15.72	16.56	78.48	23.88	67.29	73.01	46.14	59.45

with $a_1, \ldots, a_n$ in the indicated coefficient range and b in the indicated constant range. In the columns labelled E we do not fix the interpretations in advance; for all function symbols we search for an appropriate linear interpretation. In the columns labelled NE we start with the default natural interpretations but allow other linear interpretations if (innermost) termination cannot be proved. Determining appropriate coefficients can be done by a straightforward but inefficient "generate and test" algorithm. We implemented a more involved algorithm in our termination tool, but we anticipate that the recent techniques described in [6] might be useful to optimize the search for coefficients.

We list the number of successful termination attempts, the number of failures (which means that no termination proof was found while fully exploring the search space implied by the options), and the number of timeouts, which we set to 30 seconds. The figures below the number of successes and failures indicate the average time in seconds.

By using coefficients from $\{0, 1, 2\}$ very few additional examples can be handled. For termination the only difference is [2, Example 3.18] which contains a unary function double that requires $2x$ as interpretation. The effect of allowing -1 as constant is more apparent. Taking default natural interpretations for certain common function symbols reduces the execution time considerably but also reduces the termination proving power. However, the NE columns clearly show that fixing natural interpretations initially but allowing different interpretations later is a very useful idea.

In Table 3 we use the negative coefficient method developed in Section 4. Not surprisingly, this method is more suited for proving innermost termination because the method is incompatible with the recent modular refinements of the dependency pair method when proving termination (for non-overlapping TRSs).

Comparing the first (last) three columns in Table 3 with the last three columns in Table 1 (Table 2) one might be tempted to conclude that the approach developed in Section 4 is useless in practice. We note however that several chal-

Table 2. Negative constants: innermost termination.

constant	$0, 1$			$0, 1, 2$			$0, 1, -1$		
coefficient	$0, 1$			$0, 1, 2$			$0, 1$		
interpretation	E	N	NE	E	N	NE	E	N	NE
success	200	177	202	202	179	204	209	183	211
	0.19	0.10	0.09	0.19	0.10	0.09	0.23	0.14	0.12
failure	39	62	37	37	60	35	30	56	28
	0.04	0.03	0.05	1.19	0.16	1.47	0.92	0.43	1.26
timeout	0	0	0	0	0	0	0	0	0
total time	39.46	18.92	19.59	81.94	26.97	70.07	75.24	49.10	61.24

Table 3. Negative coefficients.

	termination			innermost termination		
interpretation	E	N	NE	E	N	NE
success	109	102	114	181	161	185
	0.74	0.39	0.58	0.04	0.06	0.07
failure	71	96	66	30	62	28
	2.50	0.71	2.48	1.72	0.33	2.00
timeout	39	21	39	28	16	26
total time	1428.41	738.16	1400.60	899.01	511.14	848.25

lenging examples can only be handled by polynomial interpretations with negative coefficients. We mention TRSs that are (innermost) terminating because of non-linearity (e.g. [2, Examples 3.46, 4.12, 4.25]) and TRSs that are terminating because the difference of certain arguments decreases (e.g. [2, Example 4.30]).

References

1. T. Arts and J. Giesl. Termination of term rewriting using dependency pairs. *Theoretical Computer Science*, 236:133–178, 2000.
2. T. Arts and J. Giesl. A collection of examples for termination of term rewriting using dependency pairs. Technical Report AIB-2001-09, RWTH Aachen, 2001.
3. F. Baader and T. Nipkow. *Term Rewriting and All That.* Cambridge University Press, 1998.

4. A. Ben Cherifa and P. Lescanne. Termination of rewriting systems by polynomial interpretations and its implementation. *Science of Computer Programming*, 9:137–159, 1987.
5. E. Contejean, C. Marché, B. Monate, and X. Urbain. C*i*ME version 2, 2000. Available at `http://cime.lri.fr/`.
6. E. Contejean, C. Marché, A.-P. Tomás, and X. Urbain. Mechanically proving termination using polynomial interpretations. Research Report 1382, LRI, 2004.
7. N. Dershowitz. 33 Examples of termination. In *French Spring School of Theoretical Computer Science*, volume 909 of *LNCS*, pages 16–26, 1995.
8. N. Dershowitz and C. Hoot. Natural termination. *Theoretical Computer Science*, 142(2):179–207, 1995.
9. J. Giesl. Generating polynomial orderings for termination proofs. In *Proc. 6th RTA*, volume 914 of *LNCS*, pages 426–431, 1995.
10. J. Giesl, T. Arts, and E. Ohlebusch. Modular termination proofs for rewriting using dependency pairs. *Journal of Symbolic Computation*, 34(1):21–58, 2002.
11. J. Giesl, R. Thiemann, P. Schneider-Kamp, and S. Falke. Automated termination proofs with AProVE. In *Proc. 15th RTA*, volume 3091 of *LNCS*, pages 210–220, 2004.
12. N. Hirokawa and A. Middeldorp. Automating the dependency pair method. In *Proc. 19th CADE*, volume 2741 of *LNAI*, pages 32–46, 2003.
13. N. Hirokawa and A. Middeldorp. Dependency pairs revisited. In *Proc. 16th RTA*, volume 3091 of *LNCS*, pages 249–268, 2004.
14. N. Hirokawa and A. Middeldorp. Tyrolean termination tool. In *Proc. 7th International Workshop on Termination*, Technical Report AIB-2004-07, RWTH Aachen, pages 59–62, 2004.
15. H. Hong and D. Jakuš. Testing positiveness of polynomials. *Journal of Automated Reasoning*, 21:23–28, 1998.
16. D. Lankford. On proving term rewriting systems are Noetherian. Technical Report MTP-3, Louisiana Technical University, Ruston, LA, USA, 1979.
17. S. Lucas. Polynomials for proving termination of context-sensitive rewriting. In *Proc. 7th FoSSaCS*, volume 2987 of *LNCS*, pages 318–332, 2004.
18. S. Lucas. Polynomials over the reals in proof of termination. In *Proc. 7th International Workshop on Termination*, Technical Report AIB-2004-07, RWTH Aachen, pages 39–42, 2004.
19. J. Steinbach. Generating polynomial orderings. *Information Processing Letters*, 49:85–93, 1994.
20. J. Steinbach and U. Kühler. Check your ordering – termination proofs and open problems. Technical Report SR-90-25, Universität Kaiserslautern, 1990.
21. Terese. *Term Rewriting Systems*, volume 55 of *Cambridge Tracts in Theoretical Computer Science*. Cambridge University Press, 2003.
22. R. Thiemann, J. Giesl, and P. Schneider-Kamp. Improved modular termination proofs using dependency pairs. In *Proc. 2nd IJCAR*, volume 3097 of *LNAI*, pages 75–90, 2004.
23. X. Urbain. Modular & incremental automated termination proofs. *Journal of Automated Reasoning*, 2004. To appear.
24. H. Zantema. Termination of term rewriting by semantic labelling. *Fundamenta Informaticae*, 24:89–105, 1995.