Search Techniques for Rational Polynomial Orders[*]

Carsten Fuhs[1], Rafael Navarro-Marset[2], Carsten Otto[1], Jürgen Giesl[1], Salvador Lucas[2], and Peter Schneider-Kamp[1]

[1] LuFG Informatik 2, RWTH Aachen University, Germany
{fuhs,giesl,psk}@informatik.rwth-aachen.de, carsten.otto@rwth-aachen.de
[2] DSIC, Universidad Politécnica de Valencia, Spain
{slucas,rnavarro}@dsic.upv.es

Abstract. Polynomial interpretations are a standard technique used in almost all tools for proving termination of term rewrite systems (TRSs) automatically. Traditionally, one applies interpretations with polynomials over the naturals. But recently, it was shown that interpretations with polynomials over the rationals can be significantly more powerful. However, searching for such interpretations is considerably more difficult than for natural polynomials. Moreover, while there exist highly efficient SAT-based techniques for finding natural polynomials, no such techniques had been developed for rational polynomials yet. In this paper, we tackle the two main problems when applying rational polynomial interpretations in practice: (1) We develop new criteria to decide when to use rational instead of natural polynomial interpretations. (2) Afterwards, we present SAT-based methods for finding rational polynomial interpretations and evaluate them empirically.

Topics. computer algebra systems and automated theorem provers, implementation and performance issues

Keywords. termination, term rewriting, SAT solving, dependency pairs

1   Introduction

Orders based on polynomial interpretations are essential for termination proofs. Recently, [16–18] showed that polynomial interpretations *over the rationals* are strictly more powerful for proving termination than those over the naturals.[3]

One of the most popular termination techniques that is implemented in virtually all current tools for termination analysis of TRSs is the *dependency pair*

---

[3] Several such examples where this is *provably* the case are presented in Sect. 3.1.

(DP) method, cf. e.g. [1, 9, 11–13]. In principle, rational polynomial interpretations can immediately be used in this method. In other words, the polynomial constraints (over the rationals) which have to be generated are *the same* as those for polynomials with natural coefficients [16, 18]. But as discussed in [18], the main problem when attempting to use rational polynomials in practice is that one needs *efficient and suitable methods* to find polynomial interpretations over the rationals automatically. Here, there are two main challenges:

Since searching for rational polynomial interpretations is much more time-consuming than for natural interpretations, one needs criteria to decide when to use rational interpretations. After recapitulating the necessary prerequisites on termination proving in Sect. 2, the first contribution of this paper are such criteria, presented in Sect. 3. Here, we first introduce *sufficient* criteria (i.e., criteria which state that the termination proof will fail when just using natural polynomials). Afterwards, we introduce *heuristics* to characterize the remaining termination problems where rational polynomials are "likely" to be needed.

The other challenge are efficient methods to search for rational interpretations. For interpretations over the naturals, until recently the best known techniques were dedicated constraint-based algorithms like [3]. However, recently a new approach was developed in [7] which proposes the use of SAT solvers for generating natural polynomial interpretations. This approach was implemented in the termination tool AProVE [10] and it leads to speed-ups in orders of magnitude over constraint-based algorithms. While there already exists a constraint-based algorithm for finding rational polynomial interpretations [18][4] (implemented in the tool MU-TERM [15]), a SAT-based approach similar to [7] could bring similar improvements when polynomials over the rationals are considered. The second contribution of this paper (in Sect. 4) is the development of two such SAT-based approaches. Finally, Sect. 5 contains an extensive experimental evaluation.

2 Termination Proving with Rational Polynomials

Definition 1 (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 by $DP(\mathcal{R})$.*

Example 2. Consider the following TRS $\mathcal{R}$ from [20], where random(x) computes a random number between 0 and x.

$$\text{nonZero}(0) \to \text{false} \quad (1) \qquad \text{random}(x) \to \text{rand}(x, 0) \quad (3)$$

$$\text{nonZero}(\text{s}(x)) \to \text{true} \quad (2) \qquad \text{rand}(x, y) \to \text{if}(\text{nonZero}(x), x, y) \quad (4)$$

---

[4] [18] also presents an algorithm for *real* polynomial interpretations. Extending the results of the current paper to real interpretations is a topic for future work.

$$\mathsf{p}(0) \to 0 \qquad (5) \qquad \mathsf{if}(\mathsf{false}, x, y) \to y \qquad\qquad\qquad (8)$$

$$\mathsf{p}(\mathsf{s}(x)) \to x \qquad (6) \qquad \mathsf{if}(\mathsf{true}, x, y) \to \mathsf{rand}(\mathsf{p}(x), \mathsf{id\_inc}(y)) \quad (9)$$

$$\mathsf{id\_inc}(x) \to x \qquad (7) \qquad \mathsf{id\_inc}(x) \to \mathsf{s}(x) \qquad\qquad (10)$$

The defined symbols are nonZero, p, id_inc, random, rand, if, *and the DPs are*

$$\mathsf{RANDOM}(x) \to \mathsf{RAND}(x, 0) \qquad (11) \qquad \mathsf{IF}(\mathsf{true}, x, y) \to \mathsf{RAND}(\mathsf{p}(x), \mathsf{id\_inc}(y)) \ (14)$$

$$\mathsf{RAND}(x, y) \to \mathsf{IF}(\mathsf{nonZero}(x), x, y) \ (12) \qquad \mathsf{IF}(\mathsf{true}, x, y) \to \mathsf{P}(x) \qquad\qquad (15)$$

$$\mathsf{RAND}(x, y) \to \mathsf{NONZERO}(x) \qquad (13) \qquad \mathsf{IF}(\mathsf{true}, x, y) \to \mathsf{ID\_INC}(y) \qquad (16)$$

The newset formulation of the DP method is the so-called *DP framework* [9, 11]. In this framework, termination techniques operate on sets of dependency pairs instead of TRSs. We refer to such techniques as *DP processors*. Formally, a DP processor is a function *Proc* which takes a set of DPs as input and returns several new sets of DPs which then have to be solved instead. These DP processors are *sound*: if d is a set of DPs, $Proc(d) = \{d_1, \ldots, d_n\}$, and all $d_1, \ldots, d_n$ represent terminating problems, then the original problem d is also terminating.[5]

Termination proofs in the DP framework start with the initial set of DPs $DP(\mathcal{R})$. Then DP processors are applied repeatedly. If the final processors return empty sets, then termination is proved. In Thm. 5 and 6 we recapitulate the two most important DP processors. The first uses an *estimated dependency graph* to estimate which DPs (i.e., which "function calls") follow each other in evaluations.

Definition 3 (Estimated Dependency Graph). *Let $\mathcal{P}$ be a set of DPs. The nodes of the* estimated $\mathcal{P}$-dependency graph *are the pairs of $\mathcal{P}$ and there is an arc from $s \to t$ to $u \to v$ iff* REN(CAP(t)) *and u unify. Here,* CAP(t) *replaces all subterms of t with defined root symbol by fresh variables and* REN(t) *linearizes t by renaming all occurrences of variables into pairwise different fresh variables.*

Example 4. For the TRS in Ex. 2, we obtain the following estimated $DP(\mathcal{R})$-dependency graph.
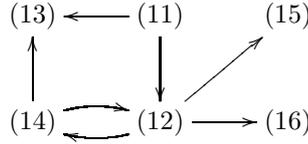
For example, the reason for the arc from (12) to (14) is that if t is the right-hand side of (12) and u is the left-hand side of (14), then REN(CAP(t)) = REN(IF(z, x, y)) = IF(z', x', y') *and $u = $ IF(true, x, y) clearly unify.*

One can prove termination separately for each strongly connected component (SCC) of the estimated dependency graph. Therefore, the following processor modularizes termination proofs by decomposing the set of DPs.

---

[5] To ease readability we consider just sets of dependency pairs instead of *DP problems* [9, 11]. This suffices for the presentation of the results of this paper. We also refer to [9, 11] for a precise definition of "terminating" problems.

Theorem 5 (Dependency Graph Processor). *Let $\mathcal{P}$ be a set of DPs whose estimated dependency graph has n SCCs. For every $i \in \{1, \ldots, n\}$, let $\mathcal{P}_i$ be the set of DPs in the i-th SCC. Then the following DP processor is sound:*

$$Proc(\mathcal{P}) = \{\mathcal{P}_1, \ldots, \mathcal{P}_n\}$$

So in our example, the original set of DPs $DP(\mathcal{R}) = \{(11), \ldots, (16)\}$ is transformed to the subset $\mathcal{P}_1 = \{(12), (14)\}$, i.e., $Proc(DP(\mathcal{R})) = \{\mathcal{P}_1\}$.

The next processor is based on *reduction pairs* $(\succsim, \succ)$. Here, $\succsim$ is reflexive, transitive, monotonic (i.e., $s \succsim t$ implies $f(\ldots s \ldots) \succsim f(\ldots t \ldots)$ for all function symbols f), and stable (i.e., $s \succsim t$ implies $s\sigma \succsim t\sigma$ for all substitutions σ) and $\succ$ is a stable well-founded order compatible with $\succsim$ (i.e., $\succsim \circ \succ \subseteq \succ$ or $\succ \circ \succsim \subseteq \succ$).

The following processor generates inequality constraints which have to be satisfied by a reduction pair $(\succsim, \succ)$. The constraints require that all DPs in $\mathcal{P}$ are strictly or weakly decreasing (i.e., w.r.t. $\succ$ or $\succsim$) and all *usable rules* $\mathcal{U}(\mathcal{P})$ are weakly decreasing. Then one can delete all strictly decreasing DPs from $\mathcal{P}$.

The *usable rules* include all rules that can reduce the terms in right-hand sides of $\mathcal{P}$ when their variables are instantiated with normal forms. To ensure that it suffices to regard only the *usable* rules instead of *all* rules in the following processor, one has to demand that $\succsim$ is $\mathcal{C}_\varepsilon$-*compatible*, i.e., that $\mathsf{c}(x, y) \succsim x$ and $\mathsf{c}(x, y) \succsim y$ hold for a fresh function symbol c [11, 13]. This requirement is satisfied by almost all quasi-orders used in practice.

Theorem 6 (Reduction Pair Processor). *Let $(\succsim, \succ)$ be a reduction pair where $\succsim$ is $\mathcal{C}_\varepsilon$-compatible. Then the following DP processor Proc is sound.*

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

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*

$$\mathcal{U}(f(t_1, \ldots, t_n)) = Rls(f) \cup \bigcup_{\ell \to r \in Rls(f)} \mathcal{U}(r) \cup \bigcup_{i=1}^{n} \mathcal{U}(t_i)$$

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

There are many techniques to search for reduction pairs automatically (recursive path orders, polynomial interpretations, etc. [4]). In this paper, we consider polynomial interpretations $\mathcal{P}ol$ which map every n-ary function symbol f to a polynomial $f_{\mathcal{P}ol} \in \mathbb{Q}_0[x_1, \ldots, x_n]$. So the coefficients of $f_{\mathcal{P}ol}$ are from $\mathbb{Q}_0 = \{\frac{p}{q} \mid p \in \mathbb{N}, q \in \mathbb{N} \setminus \{0\}\}$ and the variables $x_1, \ldots, x_n$ also range over $\mathbb{Q}_0$. This is in contrast to traditional polynomial interpretations where one uses $\mathbb{N} = \{0, 1, 2, \ldots\}$ instead of $\mathbb{Q}_0$. The mapping $\mathcal{P}ol$ is extended to terms by defining $[x]_{\mathcal{P}ol} = x$ for variables x and $[f(t_1, \ldots, t_n)]_{\mathcal{P}ol} = f_{\mathcal{P}ol}([t_1]_{\mathcal{P}ol}, \ldots, [t_n]_{\mathcal{P}ol})$. An interpretation $\mathcal{P}ol$ induces an order $\succ_{\mathcal{P}ol}$ and a quasi-order $\succsim_{\mathcal{P}ol}$ where $s \succsim_{\mathcal{P}ol} t$ iff $[s]_{\mathcal{P}ol} - [t]_{\mathcal{P}ol} \geq 0$ holds for all instantiations of the variables with numbers from $\mathbb{Q}_0$. To

define $\succ_{\mathcal{P}ol}$ one needs a number $\delta > 0$ and then $s \succ_{\mathcal{P}ol} t$ iff $[s]_{\mathcal{P}ol} - [t]_{\mathcal{P}ol} \geq \delta$ holds for all instantiations of the variables with numbers from $\mathbb{Q}_0$. Then, $\succ_{\mathcal{P}ol}$ is also well founded for rational polynomial interpretations [16, 18].

Example 7. For the TRS of Ex. 2, the dependency graph processor reduced the set of DPs to $\mathcal{P}_1 = \{(12), (14)\}$. The rules for the defined symbols nonZero, p, and id_inc in the right-hand sides of (12) and (14) are usable, i.e., $\mathcal{U}(\mathcal{P}_1) = \{(1), (2), (5), (6), (7), (10)\}$. We have to find a reduction pair which makes the rules in $\mathcal{U}(\mathcal{P}_1)$ weakly decreasing and the DPs in $\mathcal{P}_1$ weakly or strictly decreasing. Then the strictly decreasing DPs are removed. We use $(\succsim_{\mathcal{P}ol}, \succ_{\mathcal{P}ol})$ with

$$0_{\mathcal{P}ol} = 0 \qquad\qquad \mathsf{p}_{\mathcal{P}ol} = \tfrac{1}{2}\,x_1$$
$$\mathsf{s}_{\mathcal{P}ol} = 2\,x_1 + 1 \qquad\qquad \mathsf{id\_inc}_{\mathcal{P}ol} = 2\,x_1 + 1$$
$$\mathsf{true}_{\mathcal{P}ol} = 1 \qquad\qquad \mathsf{RAND}_{\mathcal{P}ol} = 2\,x_1$$
$$\mathsf{false}_{\mathcal{P}ol} = 0 \qquad\qquad \mathsf{IF}_{\mathcal{P}ol} = x_1 + x_2$$
$$\mathsf{nonZero}_{\mathcal{P}ol} = x_1 \qquad\qquad \delta = 1$$

Now all usable rules from $\mathcal{U}(\mathcal{P}_1)$ and all DPs from $\mathcal{P}_1$ are weakly decreasing. Moreover, the DP (14) is strictly decreasing since $[\mathsf{IF}(\mathsf{true}, x, y)]_{\mathcal{P}ol} - [\mathsf{RAND}(\mathsf{p}(x), \mathsf{id\_inc}(y))]_{\mathcal{P}ol} = 1 + x - 2 * \tfrac{1}{2}\,x \geq 1$. Thus, it is removed by Thm. 6 and the resulting set of DPs is $\{(12)\}$. Afterwards, another application of the dependency graph processor results in the empty set of DPs, since now the graph has no arcs anymore. Hence, termination of this example is proved.

To measure the performance of termination tools, there is an annual *International Termination Competition* [19] where the tools are applied to a large collection of TRSs (the so-called *Termination Problem Data Base* (TPDB)). The TRS of Ex. 2 comes from the TPDB (`SchneiderKamp-trs-thiemann40`), but none of the tools in the Termination Competition 2007 could show its termination.[6] Indeed, almost all termination tools use polynomial interpretations, but most of them are restricted to interpretations with natural or integer coefficients. If they were extended to rational coefficients, TRSs like Ex. 2 could easily be handled by virtually all existing tools. Thus, this TRS shows that rational polynomial interpretations indeed increase the power of termination proving substantially.

3 Criteria for Rational Polynomial Interpretations

In this section, we introduce criteria to decide when to use rational polynomial interpretations. In Sect. 3.1 we present sufficient criteria[7] which state that the

---

[6] [20] presents a (manual) termination proof for this TRS using an improved variant of predictive labeling, but their technique has not been implemented yet. In contrast, our proof is much easier and (apart from *rational* interpretations) it only uses standard methods that are already implemented in most termination provers.

[7] The criteria in Sect. 3.1 are restricted to *linear* polynomial interpretations which are used in the vast majority of automated termination proofs for TRSs, cf. [19]. All other results of the paper (i.e., the heuristics of Sect. 3.2 as well as the automation of Sect. 4) can be used for interpretations with polynomials of arbitrary degree.

termination proof will fail if one uses natural instead of rational interpretations. In particular, this proves that rational polynomials really increase power, i.e., that there are examples where termination can be proved with rational, but not with natural interpretations. Afterwards, Sect. 3.2 introduces heuristics to detect remaining cases where rational interpretations are also likely to be needed.

3.1 Sufficient Criteria for Rational Polynomial Interpretations

Our sufficient criteria are based on the following notions of monotonicity.

Definition 8 (Monotonicity). *Let $\mathcal{P}ol$ be a linear polynomial interpretation, let f be a function symbol with arity n, let $1 \leq i \leq n$, and let $f_{\mathcal{P}ol} = f_0 + f_1\,x_1 + \ldots + f_n\,x_n$ with $f_0, \ldots, f_n \in \mathbb{Q}_0$. Then[8] f is* monotonically increasing *(MI) on i iff $f_i > 0$ and f is* strongly monotonically increasing *(SMI) on i iff $f_i \geq 1$. So if f is MI, but not SMI on i, then we have $0 < f_i < 1$, i.e., $f_i \notin \mathbb{N}$.*

Now we present sufficient criteria to detect when a function symbol must be MI but not SMI. This indicates that one has to use rational interpretations for the termination proof. We start with a criterion to detect that certain argument positions cannot be SMI. To this end, we check whether there are terms s and t where $s \succ_{\mathcal{P}ol} t$ must hold although s is embedded in t. To formalize the notion of embedding, we use the TRS $\mathcal{E}mb$ which consists of the rules $f(x_1, \ldots, x_n) \to x_i$ for all function symbols f and all $1 \leq i \leq n$ where n is the arity of f.

Theorem 9 (Sufficient Criterion for Non-SMI). *Let $\mathcal{P}ol$ be a linear polynomial interpretation. If $s \succ_{\mathcal{P}ol} t$ and $t \to_{\mathcal{E}}^* s$ for a set[9] of embedding rules $\mathcal{E} \subseteq \mathcal{E}mb$, then there is a rule $f(x_1, \ldots, x_n) \to x_i$ in $\mathcal{E}$ such that f is not SMI on i.*

Proof. Assume that for all $f(x_1, \ldots, x_n) \to x_i$ in $\mathcal{E}$, f is SMI on i. We show that $t \to_{\mathcal{E}}^m s$ implies $t \succsim_{\mathcal{P}ol} s$ by induction on m. This is a contradiction to $s \succ_{\mathcal{P}ol} t$.

Clearly, $t \to_{\mathcal{E}}^m s$ implies $t \succsim_{\mathcal{P}ol} s$ for $m = 0$. Now let $m > 0$, i.e., $t \to_{\mathcal{E}} t' \to_{\mathcal{E}}^* s$. So $t' \succsim_{\mathcal{P}ol} s$ by the induction hypothesis. Thus, it suffices to show $t \succsim_{\mathcal{P}ol} t'$.

As $t \to_{\mathcal{E}} t'$, we obtain $t = t[f(t_1, \ldots, t_i, \ldots, t_n)]_\pi$ and $t' = t[t_i]_\pi$ for some position π and some rule $f(x_1, \ldots, x_n) \to x_i$ in $\mathcal{E}$. Since $\mathcal{P}ol$ is linear, we have $f_{\mathcal{P}ol} = f_0 + f_1\,x_1 + \ldots + f_n\,x_n$ for $f_0, \ldots, f_n \in \mathbb{Q}_0$ and as f is SMI on i, we have $f_i \geq 1$. Thus, $f(x_1, \ldots, x_n) \succsim_{\mathcal{P}ol} x_i$. As $\succsim_{\mathcal{P}ol}$ is monotonic and stable, this implies $t[f(t_1, \ldots, t_i, \ldots, t_n)]_\pi \succsim_{\mathcal{P}ol} t[t_i]_\pi$ and hence, $t \succsim_{\mathcal{P}ol} t'$ as desired. $\square$

---

[8] In general, a function $f_{\mathcal{P}ol}$ is *monotonically increasing* if $x_i - y_i > 0$ implies $f_{\mathcal{P}ol}(x_1, ..., x_i, ..., x_n) - f_{\mathcal{P}ol}(x_1, ..., y_i, ..., x_n) > 0$ for all numbers $x_1, ..., x_n, y_i$ and $f_{\mathcal{P}ol}$ is *strongly monotonically increasing* if $x_i - y_i \geq \delta$ implies $f_{\mathcal{P}ol}(x_1, ..., x_i, ..., x_n) - f_{\mathcal{P}ol}(x_1, ..., y_i, ..., x_n) \geq \delta$ for all numbers $x_1, ..., x_n, y_i$ and all $\delta > 0$. So obviously, $\frac{\partial f_{\mathcal{P}ol}}{\partial x_i} > 0$ implies that $f_{\mathcal{P}ol}$ is monotonically increasing and $\frac{\partial f_{\mathcal{P}ol}}{\partial x_i} \geq 1$ implies that $f_{\mathcal{P}ol}$ is strongly monotonically increasing.

[9] Explicitly considering the rules $\mathcal{E}$ which are needed to come from t to s (instead of considering $\mathcal{E}mb$) gives a better approximation of the "non-SMI" arguments.

Example 10. To illustrate the criterion of Thm. 9, we consider the following TRS from the TPDB (`secret05-tpa2`).

$$\mathsf{minus}(x, 0) \to x \qquad (17) \quad \mathsf{f}(\mathsf{s}(x), y) \to \mathsf{f}(\mathsf{p}(\mathsf{minus}(\mathsf{s}(x), y)), \mathsf{p}(\mathsf{minus}(y, \mathsf{s}(x)))) \quad (20)$$

$$\mathsf{minus}(\mathsf{s}(x), \mathsf{s}(y)) \to \mathsf{minus}(x, y) \quad (18) \quad \mathsf{f}(x, \mathsf{s}(y)) \to \mathsf{f}(\mathsf{p}(\mathsf{minus}(x, \mathsf{s}(y))), \mathsf{p}(\mathsf{minus}(\mathsf{s}(y), x))) \quad (21)$$

$$\mathsf{p}(\mathsf{s}(x)) \to x \qquad (19)$$

This TRS has 11 DPs, but an application of the dependency graph processor yields the two subsets $\{(22)\}$ and $\{(23), (24)\}$, where

$$\mathsf{MINUS}(\mathsf{s}(x), \mathsf{s}(y)) \to \mathsf{MINUS}(x, y) \tag{22}$$

$$\mathsf{F}(\mathsf{s}(x), y) \to \mathsf{F}(\mathsf{p}(\mathsf{minus}(\mathsf{s}(x), y)), \mathsf{p}(\mathsf{minus}(y, \mathsf{s}(x)))) \tag{23}$$

$$\mathsf{F}(x, \mathsf{s}(y)) \to \mathsf{F}(\mathsf{p}(\mathsf{minus}(x, \mathsf{s}(y))), \mathsf{p}(\mathsf{minus}(\mathsf{s}(y), x))) \tag{24}$$

The DP (22) can immediately be removed by the reduction pair processor. It remains to find a polynomial interpretation such that one of the DPs (23) and (24) is strictly decreasing and the other DP and the usable rules $\{(17), (18), (19)\}$ are weakly decreasing. For both DPs (23) and (24), the left-hand side is embedded in the right-hand side. For instance for (23), we have $\mathsf{F}(\mathsf{p}(\mathsf{minus}(\mathsf{s}(x), y)), \mathsf{p}(\mathsf{minus}(y, \mathsf{s}(x)))) \to_{\mathcal{E}}^* \mathsf{F}(\mathsf{s}(x), y)$ with $\mathcal{E} = \{\mathsf{p}(x_1) \to x_1,\ \mathsf{minus}(x_1, x_2) \to x_1\}$. So by Thm. 9, p or minus cannot be SMI on 1.

Now we present criteria for MI. Clearly, if one has to satisfy a *collapsing* inequality $s \succsim_{\mathcal{P}ol} x$ for a variable $x \in \mathcal{V}$, then the polynomial $[s]_{\mathcal{P}ol}$ must contain x. Hence, x is at a monotonically increasing position in s. For any position π in a term s, let $\mathrm{trace}(s, \pi)$ contain all pairs (f, i) such that π is below the i-th argument of the function symbol f. So $\mathrm{trace}(s, \varepsilon) = \varnothing$ and $\mathrm{trace}(f(s_1, \dots, s_n), i\,\pi') = \{(f, i)\} \cup \mathrm{trace}(s_i, \pi')$. We omit the proof of Thm. 11, since it is obvious.

Theorem 11 (First Sufficient Criterion for MI). *Let $\mathcal{P}ol$ be a linear polynomial interpretation. If $s \succsim_{\mathcal{P}ol} x$ for $x \in \mathcal{V}$, then there exists a position π in s with $s|_\pi = x$ where f is MI on i for all $(f, i) \in \mathrm{trace}(s, \pi)$.*

Example 12. To illustrate the criterion from Thm. 11, we continue the example from Ex. 10. Since the rule (19) is usable, our polynomial interpretation has to satisfy $\mathsf{p}(\mathsf{s}(x)) \succsim_{\mathcal{P}ol} x$. We have $\mathsf{p}(\mathsf{s}(x))|_{1\,1} = x$ and $\mathrm{trace}(\mathsf{p}(\mathsf{s}(x)), 1\,1) = \{(\mathsf{p}, 1), (\mathsf{s}, 1)\}$. Hence, both p and s have to be MI on 1. Similarly, the rule (17) is also usable and therefore, we have to satisfy $\mathsf{minus}(x, 0) \succsim_{\mathcal{P}ol} x$. By Thm. 11 this implies that minus also has to be MI on 1.

As both p and minus are MI on 1 but at least one of them is not SMI on 1 (cf. Ex. 10), the constraints of the reduction pair processor are not satisfied by a linear polynomial interpretation over the naturals. More precisely, if $\mathsf{p}_{\mathcal{P}ol} = p_0 + p_1\,x_1$ and $\mathsf{minus}_{\mathcal{P}ol} = m_0 + m_1\,x_1 + m_2\,x_2$ then $0 < p_1 < 1$ or $0 < m_1 < 1$.

Indeed, the following rational polynomial interpretation makes all usable rules weakly decreasing and both DPs (23) and (24) strictly decreasing. Hence, they can both be removed, which proves termination of this example.

$$0_{\mathcal{P}ol} = 0 \qquad\qquad\qquad \mathsf{minus}_{\mathcal{P}ol} = x_1$$
$$\mathsf{s}_{\mathcal{P}ol} = 2\,x_1 + 1 \qquad\qquad\qquad \mathsf{F}_{\mathcal{P}ol} = x_1 + x_2$$
$$\mathsf{p}_{\mathcal{P}ol} = \tfrac{1}{2}\,x_1 \qquad\qquad\qquad \delta = \tfrac{1}{2}$$

Example 13. The criteria presented so far can also detect the need for rational coefficients in the TRS of Ex. 2. As explained in Ex. 7, one has to find an interpretation such that one of the DPs (12) and (14) is strictly decreasing and the other DP and the usable rules $\{(1), (2), (5), (6), (7), (10)\}$ are weakly decreasing. So

$$
\begin{aligned}
\mathsf{RAND}(\mathsf{s}(x), y) &\succsim_{\mathcal{P}ol} \mathsf{IF}(\mathsf{nonZero}(\mathsf{s}(x)), \mathsf{s}(x), y) && \textit{by weak decrease of (12)}\\
&\succsim_{\mathcal{P}ol} \mathsf{IF}(\mathsf{true}, \mathsf{s}(x), y) && \textit{by weak decrease of (2)}\\
&\succsim_{\mathcal{P}ol} \mathsf{RAND}(\mathsf{p}(\mathsf{s}(x)), \mathsf{id\_inc}(y)) && \textit{by weak decrease of (14)}
\end{aligned}
$$

and as at least one of the DPs is strictly decreasing, we also have[10]

$$\mathsf{RAND}(\mathsf{s}(x), y) \succ_{\mathcal{P}ol} \mathsf{RAND}(\mathsf{p}(\mathsf{s}(x)), \mathsf{id\_inc}(y)).$$

Note that the term in the left-hand side is embedded in the right-hand side, i.e., $\mathsf{RAND}(\mathsf{p}(\mathsf{s}(x)), \mathsf{id\_inc}(y)) \to_{\mathcal{E}}^{*} \mathsf{RAND}(\mathsf{s}(x), y)$ with $\mathcal{E} = \{\mathsf{p}(x_1) \to x_1, \mathsf{id\_inc}(x_1) \to x_1\}$. So by Thm. 9, one of the symbols p and $\mathsf{id\_inc}$ is not SMI on 1. But due to the usable rules (6) and (7), by Thm. 11 both p and $\mathsf{id\_inc}$ have to be MI on 1. Thus here we again need a rational polynomial interpretation. More precisely, if $\mathsf{p}_{\mathcal{P}ol} = p_0 + p_1\,x_1$ and $\mathsf{id\_inc}_{\mathcal{P}ol} = i_0 + i_1\,x_1 + i_2\,x_2$, then $0 < p_1 < 1$ or $0 < i_1 < 1$.

Thm. 14 is a second criterion for MI which can be used instead of Thm. 11.

Theorem 14 (Second Sufficient Criterion for MI). *Let $\mathcal{P}ol$ be a linear polynomial interpretation. Let $C[f(s_1, \ldots, s_n)] \succ_{\mathcal{P}ol} C[f(t_1, \ldots, t_n)]$ and let there be an $1 \le i \le n$ such that $s_j \in \mathcal{V}$ for all $j \ne i$. Then f is MI on i. If moreover t_i is a variable that does not occur in s_i, then there must be an $i' \ne i$ with $s_{i'} = t_i$ and f is also MI on i'.*

Proof. Clearly, $C[f(s_1, \ldots, s_n)] \succ_{\mathcal{P}ol} C[f(t_1, \ldots, t_n)]$ for a context C implies $f(s_1, \ldots, s_n) \succ_{\mathcal{P}ol} f(t_1, \ldots, t_n)$. If $f_{\mathcal{P}ol} = f_0 + f_1\,x_1 + \ldots + f_n\,x_n$, then $pl = [f(s_1, \ldots, s_n)]_{\mathcal{P}ol} - [f(t_1, \ldots, t_n)]_{\mathcal{P}ol} = f_1\,([s_1]_{\mathcal{P}ol} - [t_1]_{\mathcal{P}ol}) + \ldots + f_n\,([s_n]_{\mathcal{P}ol} - [t_n]_{\mathcal{P}ol}) \ge \delta$. Thus we must have $f_i > 0$ (i.e., f is MI on i), because otherwise the polynomial pl is 0 or negative when instantiating all variables with 0.

Now let t_i be a variable that does not occur in s_i. If the variable t_i did not occur in s, then the coefficient for the variable t_i in the polynomial pl would be $-f_i$, i.e., pl would be negative if one instantiates t_i by a large enough number. Hence, there must be an $i' \ne i$ with $s_{i'} = t_i$ and $f_{i'} > 0$. $\qquad\square$

Example 15. To illustrate the criterion of Thm. 14, we consider the following TRS from the TPDB (Zantema-jw05).

$$\mathsf{f}(\mathsf{f}(\mathsf{a}, x), \mathsf{a}) \to \mathsf{f}(\mathsf{f}(x, \mathsf{f}(\mathsf{a}, \mathsf{a})), \mathsf{a}) \tag{25}$$

---

[10] To automate Thm. 9, one has to search for inequalities $s \succ_{\mathcal{P}ol} t$ where s is embedded in t. To this end, one could use *narrowing* on right-hand sides of DPs.

This TRS has 3 DPs:

$$\mathsf{F}(\mathsf{f}(\mathsf{a},x),\mathsf{a}) \to \mathsf{F}(\mathsf{f}(x,\mathsf{f}(\mathsf{a},\mathsf{a})),\mathsf{a}) \quad (26) \qquad\qquad \mathsf{F}(\mathsf{f}(\mathsf{a},x),\mathsf{a}) \to \mathsf{F}(\mathsf{a},\mathsf{a}) \quad (28)$$
$$\mathsf{F}(\mathsf{f}(\mathsf{a},x),\mathsf{a}) \to \mathsf{F}(x,\mathsf{f}(\mathsf{a},\mathsf{a})) \qquad (27)$$

The dependency graph processor removes the DP (28). We first try to find a polynomial interpretation where the DP (27) is strictly decreasing and where the DP (26) and the usable rule (25) are weakly decreasing. This is easy by using $\mathsf{F}_{\mathcal{P}ol} = x_2$, $\mathsf{a}_{\mathcal{P}ol} = 1$, $\mathsf{f}_{\mathcal{P}ol} = 0$, and $\delta = 1$. Hence, (27) can be removed.

Finally, we have to find a polynomial interpretation where (26) is strictly decreasing and where the usable rule (25) is weakly decreasing. Now we can apply Thm. 14 by choosing "C", "$f(s_1, s_2)$", "i", and "$f(t_1, t_2)$" as follows: C is $\mathsf{F}(\square, \mathsf{a})$, $f(s_1, s_2)$ is $\mathsf{f}(\mathsf{a}, x)$, i is 1, and $f(t_1, t_2)$ is $\mathsf{f}(x, \mathsf{f}(\mathsf{a}, \mathsf{a}))$. So by Thm. 14, f is MI on 1 and as the variable t_1 does not occur in s_1, f is also MI on 2.

Moreover, strict decrease of (26) implies $\mathsf{F}(\mathsf{f}(\mathsf{a},\mathsf{a}),\mathsf{a}) \succ_{\mathcal{P}ol} \mathsf{F}(\mathsf{f}(\mathsf{a},\mathsf{f}(\mathsf{a},\mathsf{a})),\mathsf{a})$ where the left-hand side is embedded in the right-hand side, i.e., $\mathsf{F}(\mathsf{f}(\mathsf{a},\mathsf{f}(\mathsf{a},\mathsf{a})),\mathsf{a})$ $\to_{\mathcal{E}}^{*} \mathsf{F}(\mathsf{f}(\mathsf{a},\mathsf{a}),\mathsf{a})$ with $\mathcal{E} = \{\mathsf{f}(x_1, x_2) \to x_1\}$ or $\mathcal{E} = \{\mathsf{f}(x_1, x_2) \to x_2\}$. So by Thm. 9, f is neither SMI on 1 nor on 2. Hence if $\mathsf{f}_{\mathcal{P}ol} = f_0 + f_1\, x_1 + f_2\, x_2$, then both $0 < f_1 < 1$ and $0 < f_2 < 1$. Indeed, (26) is strictly decreasing and (25) is weakly decreasing if we use the following interpretation:

$$\mathsf{f}_{\mathcal{P}ol} = \tfrac{1}{4}\, x_1 + \tfrac{1}{4}\, x_2 \qquad\qquad \mathsf{F}_{\mathcal{P}ol} = 4\, x_1 \qquad\qquad \mathsf{a}_{\mathcal{P}ol} = 4 \qquad\qquad \delta = 2$$

3.2 Heuristics for Rational Polynomial Interpretations

The criteria from Sect. 3.1 are only sufficient, i.e., there are TRSs where rational interpretations are needed although the criteria are not fulfilled. Therefore, we now develop heuristics which indicate that rational polynomials are *likely* to be useful. So one should apply rational interpretations whenever one of the sufficient criteria of Sect. 3.1 or one of the following heuristical criteria is fulfilled.

The first heuristic suggests to apply rational interpretations whenever a destructor symbol occurs in the right-hand side of a DP. A *destructor* is a symbol which is the inverse function to a constructor. So if s is a constructor and we have a rule $\mathsf{p}(\mathsf{s}(x)) \to x$, then the symbol p is a destructor.

Heuristic 16 (Destructor Heuristic). *Let $\mathcal{P}$ be a set of DPs. If the TRS $\mathcal{R}$ contains $\mathsf{f}(\mathsf{c}(x_1, \ldots, x_n)) \to x_i$, c is a constructor, and f occurs in the right-hand side of a DP from $\mathcal{P}$, then apply rational polynomials in the processor of Thm. 6.*

For instance, in the TRS of Ex. 2, we indeed have the rule (6) for the destructor p and p occurs in the right-hand side of the DP (14). Hence, the above heuristic suggests to apply rational polynomial interpretations.

However, one can of course also formulate destructor rules in a different way. The next heuristic serves to detect such alternative formulations.

Heuristic 17 (Permutation Heuristic). *Let $\mathcal{R}$ be a TRS and $\mathcal{P}$ be a set of DPs. If $\mathcal{R} \cup \mathcal{P}$ contains a rule $C_1[t_1] \to C_2[t_2]$ where $t_1 = f(\ldots, D_1[g(\ldots)], \ldots)$*

and $t_2 = \mathsf{g}(\ldots, D_2[\mathsf{f}(\ldots)], \ldots)$ *and where at least one of the terms t_1 or t_2 contains two nested* f*-symbols or two nested* g*-symbols, then apply rational polynomials in the processor of Thm. 6. Here, C_1, C_2, D_1, D_2 are contexts and f and g may also be the same function symbol.*

As an example, we replace the rules $\mathsf{p}(0) \to 0$ and $\mathsf{p}(\mathsf{s}(x)) \to x$ in the TRS of Ex. 2 by $\mathsf{p}(\mathsf{s}(0)) \to 0$ and $\mathsf{p}(\mathsf{s}(\mathsf{s}(x))) \to \mathsf{s}(\mathsf{p}(\mathsf{s}(x)))$. Now p still acts as a destructor and termination of the TRS can be proved almost[11] as before, but the destructor heuristic (Heuristic 16) fails. Instead, the permutation heuristic is applicable now.

Example 18. Another class of examples recognized by this heuristic are permutative TRSs like the following example `Endrullis-pair3swap` *from the TPDB.*

$$\mathsf{p}(\mathsf{a}(\mathsf{a}(x_0)), \mathsf{p}(x_1, \mathsf{p}(\mathsf{a}(x_2), x_3))) \to \mathsf{p}(x_2, \mathsf{p}(\mathsf{a}(\mathsf{a}(\mathsf{b}(x_1))), \mathsf{p}(\mathsf{a}(\mathsf{a}(x_0)), x_3)))$$

By two repeated applications of the dependency graph and the reduction pair processor, this example can easily be solved. However, in the reduction pair processor, one should use rational polynomial interpretations. This would be detected by the permutation heuristic above.[12]

Finally, the last heuristic detects rules where the same variable occurs twice in different arguments of a constructor on the right-hand side.

Heuristic 19 (Non-Linearity Heuristic). *Let $\mathcal{R}$ be a TRS and $\mathcal{P}$ be a set of DPs. If $\mathcal{R} \cup \mathcal{P}$ contains a rule $\ell \to C[\mathsf{c}(\ldots, t_1, \ldots, t_2, \ldots)]$ where $\mathcal{V}(t_1) \cap \mathcal{V}(t_2) \neq \varnothing$, then apply rational polynomials in the processor of Thm. 6.*

Example 20. To illustrate this heuristic, consider the following example. Its behavior is similar to Ex. 2, i.e., $\mathsf{f}(\mathsf{s}^n(0))$ rewrites to $\mathsf{f}(\mathsf{s}^m(0))$ for any $0 \leq m < n$.

$$\begin{array}{ll} \mathsf{f}(\mathsf{s}(x)) \to \mathsf{f}(\mathsf{id\_inc}(\mathsf{c}(x, x))) & \qquad \mathsf{id\_inc}(\mathsf{s}(x)) \to \mathsf{s}(\mathsf{id\_inc}(x)) \\ \mathsf{f}(\mathsf{c}(\mathsf{s}(x), y)) \to \mathsf{g}(\mathsf{c}(x, y)) & \qquad \mathsf{id\_inc}(\mathsf{c}(x, y)) \to \mathsf{c}(\mathsf{id\_inc}(x), \mathsf{id\_inc}(y)) \\ \mathsf{g}(\mathsf{c}(\mathsf{s}(x), y)) \to \mathsf{g}(\mathsf{c}(y, x)) & \qquad \mathsf{id\_inc}(0) \to 0 \\ \mathsf{g}(\mathsf{c}(x, \mathsf{s}(y))) \to \mathsf{g}(\mathsf{c}(y, x)) & \qquad \mathsf{id\_inc}(0) \to \mathsf{s}(0) \\ \mathsf{g}(\mathsf{c}(x, x)) \to \mathsf{f}(x) & \end{array}$$

When applying the dependency graph processor, the set of DPs can be split into the set of $\mathsf{ID\_INC}$*-DPs (here the termination proof is trivial) and into the set with the* F*- and* G*-DPs. Due to the DP*

$$\mathsf{F}(\mathsf{s}(x)) \;\to\; \mathsf{F}(\mathsf{id\_inc}(\mathsf{c}(x, x))), \tag{29}$$

the non-linearity heuristic applies. One can use the rational polynomial interpretation with $\mathsf{F}_{\mathcal{P}ol} = \mathsf{G}_{\mathcal{P}ol} = x_1$, $0_{\mathcal{P}ol} = 0$, $\mathsf{s}_{\mathcal{P}ol} = \mathsf{id\_inc}_{\mathcal{P}ol} = x_1 + 1$,

---

[11] The only difference is that the polynomial interpretation of s must be modified. Instead of $\mathsf{s}_{\mathcal{P}ol} = 2\,x_1 + 1$ we now use $\mathsf{s}_{\mathcal{P}ol} = 2\,x_1^2 + 1$.

[12] For this example, a termination proof is also possible with matrix orders [6], but no tool found a proof with natural polynomial interpretations in the competitions.

$\mathsf{c}_{\mathcal{P}ol} = \frac{1}{2}x_1 + \frac{1}{2}x_2$, and $\delta = \frac{1}{2}$ to remove all DPs with G on the right-hand side. *Another application of the dependency graph processor removes the remaining DP with G on the left-hand side. To handle the last DP (29), we can use the interpretation* $\mathsf{F}_{\mathcal{P}ol} = \mathsf{id\_inc}_{\mathcal{P}ol} = x_1$, $0_{\mathcal{P}ol} = \mathsf{s}_{\mathcal{P}ol} = 1$, $\mathsf{c}_{\mathcal{P}ol} = 0$, $\delta = 1$. *In contrast, it is not clear how to prove termination of this system with natural polynomial interpretations.*[13] *For example, the tool AProVE [10] was the winner of the Termination Competition 2007 for TRSs, but the version of AProVE used at the competition fails on this example.*

4 Generating Rational Interpretations by SAT Solving

In this section, we present two approaches to extend the SAT-based method of [7] in order to search for polynomial interpretations *over the rationals*. The approach of Sect. 4.1 transforms constraints over the rationals into constraints over the naturals which are then solved with the SAT-based technique of [7]. In contrast to that, Sect. 4.2 introduces a novel direct reduction of the search problem for rational polynomial interpretations into a SAT problem.

4.1 Transformation from Rationals to Naturals

To solve constraints over rational unknowns, one can reduce the problem to so-called *Diophantine* constraints where the unknowns are natural numbers. Subsequently, one can apply a Diophantine solver to solve the resulting constraints, cf. [16]. Such an approach was already implemented in the tool MU-TERM [15], but there the resulting Diophantine constraints were solved with the constraint-based solver CiME [2] instead of a more efficient approach using SAT solving. As shown in [18], this transformational approach in MU-TERM [15] is not competitive.[14]

We now illustrate our transformation in more detail. One starts with an *abstract* polynomial interpretation. It maps each function symbol to a polynomial with *abstract* coefficients. Thus, one has to determine the degree and the shape of the polynomial, but the actual coefficients are left open. For instance, for the TRS of Ex. 2 we could use an abstract polynomial interpretation $\mathcal{P}ol$ where $\mathsf{p}_{\mathcal{P}ol} = p_0 + p_1 x_1$, $\mathsf{s}_{\mathcal{P}ol} = s_0 + s_1 x_1$, etc. Here, p_0, p_1, s_0, s_1 are abstract coefficients.

To apply the reduction pair processor of Thm. 6, we obtain inequalities of the form $s \succ_{\mathcal{P}ol} t$ or $s \succsim_{\mathcal{P}ol} t$ that we would like to hold. These inequalities then lead to constraints on the abstract coefficients. To ensure $s \succsim_{\mathcal{P}ol} t$, it suffices to require that $[s]_{\mathcal{P}ol} - [t]_{\mathcal{P}ol}$ has only non-negative coefficients, cf. [14]. For $s \succ_{\mathcal{P}ol} t$, in addition we require that the constant coefficient of $[s]_{\mathcal{P}ol} - [t]_{\mathcal{P}ol}$ is > 0.[15] So

---

[13] However, one can prove termination using other techniques. For example, the tool Jambox [5] finds a proof using dependency pairs and matrix interpretations [6].

[14] It is much slower than MU-TERM's direct constraint-based approach [18] for finding rational polynomials. However, in Sect. 5 we show that our new SAT-based technique even significantly outperforms MU-TERM's direct constraint-based approach.

[15] This is sufficient, since we only regard finitely many inequalities of the form $s \succ_{\mathcal{P}ol} t$. Hence, δ can be defined to be the smallest constant coefficient of all these polynomials $[s]_{\mathcal{P}ol} - [t]_{\mathcal{P}ol}$, cf. [16, 18].

to ensure $\mathsf{p}(\mathsf{s}(x)) \succ_{\mathcal{P}ol} x$ with the abstract interpretation $\mathcal{P}ol$ above, we have to regard $[\mathsf{p}(\mathsf{s}(x))]_{\mathcal{P}ol} - [x]_{\mathcal{P}ol} = (p_0 + p_1 s_0) + (p_1 s_1 - 1)\, x$. Hence, we require

$$p_0 + p_1 s_0 > 0 \qquad (30) \qquad\qquad\qquad p_1 s_1 - 1 \geq 0 \qquad (31)$$

In this way, the search for a polynomial interpretation is transformed to the search for values of abstract coefficients satisfying certain inequalities.

In our setting, the values for the abstract coefficients may be numbers from $\mathbb{Q}_0$. To make this problem decidable, we restrict the possible values to numbers from a finite set $\mathcal{D}om = \{ \frac{p}{q} \mid 0 \leq p \leq m \wedge 1 \leq q \leq n \}$. To transform this problem into a problem with abstract coefficients over the naturals instead of the rationals, we now apply the following transformation:

1. Replace all abstract variables a by fractions $\frac{a_N}{a_D}$ where a_N and a_D are new abstract variables. Here "N" stands for "numerator" and "D" stands for "denominator". The values for the abstract variables a_N and a_D are chosen from the domains $\mathcal{D}om_N = \{0, \ldots, m\}$ and $\mathcal{D}om_D = \{1, \ldots, n\}$, respectively. So in our example, the constraints (30) and (31) would be replaced by

$$\frac{p_{0_N}}{p_{0_D}} + \frac{p_{1_N}}{p_{1_D}} \frac{s_{0_N}}{s_{0_D}} > 0 \quad (32) \qquad\qquad \frac{p_{1_N}}{p_{1_D}} \frac{s_{1_N}}{s_{1_D}} - 1 \geq 0 \quad (33)$$

2. Multiply each constraint with the product of all its denominators. So (32) is multiplied by $p_{0_D}\, p_{1_D}\, s_{0_D}$ and (33) is multiplied by $p_{1_D}\, s_{1_D}$. This yields

$$p_{0_N}\, p_{1_D}\, s_{0_D} + p_{1_N}\, s_{0_N}\, p_{0_D} > 0 \quad (34) \qquad p_{1_N}\, s_{1_N} - p_{1_D}\, s_{1_D} \geq 0 \qquad (35)$$

Now we obtained Diophantine constraints of the form $pl > 0$ or $pl \geq 0$ where pl is a (possibly non-linear) polynomial over abstract coefficients and where the values for the abstract coefficients are natural numbers.

3. Apply a Diophantine solver to search for suitable values for the abstract coefficients. In [7], it was shown how to translate Diophantine constraints into a satisfiability problem for propositional logic which can be handled by SAT solvers efficiently. In our example, the constraints (34) and (35) are for instance satisfied by $p_{0_N} = 0$, $p_{0_D} = 1$, $p_{1_N} = 1$, $p_{1_D} = 2$, $s_{0_N} = s_{0_D} = 1$, $s_{1_N} = 2$, $s_{1_D} = 1$. This corresponds to the values $p_0 = 0$, $p_1 = \frac{1}{2}$, $s_0 = 1$, $s_1 = 2$ for the original abstract coefficients. So with these values, the abstract interpretation with $\mathsf{p}_{\mathcal{P}ol} = p_0 + p_1 x_1$ and $\mathsf{s}_{\mathcal{P}ol} = s_0 + s_1 x_1$ is turned into the concrete interpretation with $\mathsf{p}_{\mathcal{P}ol} = \frac{1}{2}\, x_1$ and $\mathsf{s}_{\mathcal{P}ol} = 1 + 2\, x_1$.

4.2 SAT Encoding for Searching Rational Interpretations

Next we present an alternative approach which encodes the search for rational polynomial interpretations *directly* into a SAT problem. One again starts with an abstract polynomial interpretation and thus, one obtains constraints like (30) and (31). In this approach, we follow a heuristic suggested in [18] and let the domains for the abstract variables have the form $\mathcal{D}om = \{2^{-k}, 2^{-k+1}, \ldots, 2^{\ell-1}, 2^{\ell}\} \cup \{0\}$ for $k, \ell \in \mathbb{N}$. The advantage of such domains is

that they are particularly suitable for a SAT encoding. To encode constraints like (30) and (31) into a SAT problem, we now proceed as follows:

1. Up to now, the abstract coefficients like p_0, p_1, s_0, s_1 may take rational values from $\mathcal{D}om$. We now transform the constraints so that the abstract coefficients only take natural values from $\mathcal{D}om' = \{2^0, \ldots, 2^{k+\ell}\} \cup \{0\}$. To this end, every abstract coefficient a in the constraints is replaced by $\frac{1}{2^k} a'$ where a' is a fresh abstract coefficient. In our example, let $k = 1$ and $\ell = 2$, i.e., the values for the original abstract coefficients are from $\mathcal{D}om = \{2^{-1}, 2^0, 2^1, 2^2, 0\} = \{0, \frac{1}{2}, 1, 2, 4\}$. Then (30) and (31) are transformed into

$$\frac{1}{2} p_0' + \frac{1}{4} p_1' s_0' > 0 \quad (36) \qquad\qquad \frac{1}{4} p_1' s_1' - 1 \geq 0 \quad (37)$$

   The values for p_0', p_1', s_0', s_1' are from $\mathcal{D}om' = \{2^0, 2^1, 2^2, 2^3, 0\}$.

2. To remove the rational numbers from the constraints, one now multiplies them with the least common multiple of all denominators occurring in the respective constraint. So (36) and (37) are both multiplied by 4 which yields

$$2 p_0' + p_1' s_0' > 0 \quad (38) \qquad\qquad p_1' s_1' - 4 \geq 0 \quad (39)$$

3. Now we have again obtained *Diophantine* constraints. The only difference to the Diophantine constraints handled in existing SAT encodings like [7] is that the domains used for the values of abstract coefficients are not intervals of natural numbers, but sets of powers of 2. In [7], one used a mapping $||.||$ from Diophantine constraints to propositional formulas such that a constraint α is satisfiable with values from a domain $\{0, 1, 2, 3, \ldots, 2^n - 1\}$ iff the propositional formula $||\alpha||$ is satisfiable. We now have to modify this mapping in order to handle domains of the form $\{2^0, 2^1, \ldots, 2^n\} \cup \{0\}$.
   As usual, propositional formulas $\mathcal{F}$ are built from propositional variables $\mathcal{X}$, the constants 0 ("false") and 1 ("true"), and the usual Boolean connectives. Propositional interpretations are mappings $\mathfrak{I} : \mathcal{X} \to \{0, 1\}$ which can be extended to propositional formulas as usual (i.e., then we have $\mathfrak{I} : \mathcal{F} \to \{0, 1\}$). Moreover, one can extend $\mathfrak{I}$ further to *tuples* of formulas by defining

$$\mathfrak{I}(\langle \varphi_1, \ldots, \varphi_n \rangle) = 2^{n-1} * \mathfrak{I}(\varphi_1) + 2^{n-2} * \mathfrak{I}(\varphi_2) + \ldots + 2 * \mathfrak{I}(\varphi_{n-1}) + \mathfrak{I}(\varphi_n).$$

   Hence, then $\mathfrak{I} : \mathcal{F}^n \to \mathbb{N}$. So if $b \in \mathcal{X}$ and $\mathfrak{I}(b) = 0$, then $\mathfrak{I}(\langle 1, b \vee \neg b, b \rangle) = 4 * \mathfrak{I}(1) + 2 * \mathfrak{I}(b \vee \neg b) + \mathfrak{I}(b) = 4 * 1 + 2 * 1 + 0 = 6$.
   To determine $||.||$, one first defines the mapping of polynomials to *tuples* of propositional formulas. For numbers k, $||k||$ is the corresponding binary representation (e.g., $||6|| = \langle 1, 1, 0 \rangle$) and every abstract coefficient (i.e., Diophantine variable) a is mapped to an n-tuple of propositional variables (e.g., $||a|| = \langle a_1, a_2, a_3 \rangle$). Having defined $||pl_1||$ and $||pl_2||$ for polynomials pl_1 and pl_2, one can also define $||pl_1 + pl_2||$ and $||pl_1 * pl_2||$. Finally, one defines the mapping $||.||$ from Diophantine constraints like $pl > 0$ or $pl \geq 0$ to propositional formulas (not tuples of formulas). For details, we refer to [7].
   To handle the new domains of the form $\{2^0, \ldots, 2^n\} \cup \{0\}$ we now extend propositional interpretations also to *pairs* of tuples of formulas. If Φ and Ψ

13

are two tuples of propositional formulas, then we define

$$\mathfrak{I}(\ll \Phi, \Psi \gg) = \mathfrak{I}(\Phi) * 2^{\mathfrak{I}(\Psi)}$$

We now introduce a new mapping τ instead of $||.||$. For polynomials pl, $\tau(pl)$ is a *pair* of tuples of propositional formulas. For any number k, we define $\tau(k) = \ll ||m||, ||e|| \gg$ where $k = m * 2^e$ and m is an odd number (unless $k = m = 0$). So since $6 = 3 * 2^1$, we obtain $\tau(6) = \ll ||3||, ||1|| \gg$.

Every abstract coefficient (i.e., Diophantine variable) a is now mapped to a pair $\tau(a) = \ll a_0, \langle a_1, \ldots, a_{\lceil \log n \rceil} \rangle \gg$. Here, a_0 is just a single propositional variable (i.e., $\mathfrak{I}(a_0) \in \{0, 1\}$ for any interpretation $\mathfrak{I}$) and $\mathfrak{I}(\langle a_1, \ldots, a_{\lceil \log n \rceil} \rangle)$ can be any number between 0 and n. Hence, $\ll a_0, \langle a_1, \ldots, a_{\lceil \log n \rceil} \rangle \gg$ can indeed represent the numbers from $\{2^0, \ldots, 2^n\} \cup \{0\}$. Afterwards, one has to extend the mapping τ to more complex polynomials and to Diophantine constraints, similar to the mapping $||.||$ from [7].

In our example, we could finally obtain an interpretation with $\mathfrak{I}(\tau(p'_0)) = 0$, $\mathfrak{I}(\tau(p'_1)) = 1$, $\mathfrak{I}(\tau(s'_0)) = 2$, $\mathfrak{I}(\tau(s'_1)) = 4$. This would correspond to the solution $p_0 = \frac{1}{2} * p'_0 = 0$, $p_1 = \frac{1}{2} * p'_1 = \frac{1}{2}$, $s_0 = \frac{1}{2} * s'_0 = 1$, and $s_1 = \frac{1}{2} * s'_1 = 2$. With these values, the abstract interpretation with $\mathsf{p}_{\mathcal{P}ol} = p_0 + p_1 x_1$ and $\mathsf{s}_{\mathcal{P}ol} = s_0 + s_1 x_1$ is again turned into the concrete interpretation with $\mathsf{p}_{\mathcal{P}ol} = \frac{1}{2} x_1$ and $\mathsf{s}_{\mathcal{P}ol} = 1 + 2 x_1$.

5 Experiments and Conclusion

In Sect. 3, we developed new criteria to determine when to use rational interpretations in termination proofs. Moreover, in Sect. 4.1 and 4.2 we proposed two SAT-based approaches to automate the search for rational polynomials.

We implemented our contributions in the termination prover AProVE [10] and evaluated the performance of different variants of AProVE on all 2061 term and string rewrite systems from the TPDB. As in the Termination Competition 2007, we used a time limit of 120 seconds for each example.

In the following table, we only used the dependency graph and reduction pair processor, but no other termination techniques. In the first technique "Nat", we only searched for natural polynomials where the coefficients take values from $\{0, 1, 2, 3, 4\}$. In the technique "Rat + Sect. 4.1", we used rational coefficients from $\{\frac{p}{4} \mid 0 \le p \le 16\}$ instead[16] and applied the transformational technique of Sect. 4.1 to convert constraints over the rationals to constraints over the naturals. Here, we *always* search for rational polynomials, whereas in the technique "Rat + Sect. 4.1 + Sect. 3" we only search for rationals if this is suggested by the criteria from Sect. 3. Otherwise, we use natural polynomials with coefficients from $\{0, 1, 2, 3, 4\}$. Finally, in the technique "Rat + Sect. 4.2" we (always) use rational coefficients from $\{2^{-2}, 2^{-1}, 2^0, 2^1, 2^2, 0\}$ and apply the direct SAT-encoding from

---

[16] The idea of fixing the value of the denominator (e.g. to 4) and only to search for suitable values of the numerator was already proposed by [8].

Sect. 4.2.[17] The column "Yes" shows the number of TRSs where the termination proof succeeds. "SucTime" gives the average runtime for successful examples and "FulTime" gives the average runtime for all examples.

Nat			Rat + Sect. 4.1			Rat + Sect. 4.1 + Sect. 3			Rat + Sect. 4.2		
Yes	SucTime	FulTime	Yes	SucTime	FulTime	Yes	SucTime	FulTime	Yes	SucTime	FulTime
606	1.9 s	2.9 s	742	3.1 s	15.4 s	685	2.6 s	11.0 s	696	6.1 s	29.2 s

Comparing "Nat" with the other setting shows that rational polynomials can significantly increase power, but they also increase runtimes. The comparison of "Rat + Sect. 4.1" with "Rat + Sect. 4.1 + Sect. 3" shows the usefulness of the criteria from Sect. 3: if one applies these criteria, then runtimes are not increased that much anymore, but (as long as one does not use any other termination techniques) one also loses several examples where rational interpretations were needed. Finally, the comparison with the last setting in the table shows that the method of Sect. 4.1 which transforms constraints over the rationals to constraints over the naturals is preferable to the direct SAT encoding from Sect. 4.2.

The next experiment compares "Rat + Sect. 4.1" with the existing constraint-based method [18] for generating rational interpretations, implemented in MU-TERM [15]. More precisely, we compare this version ("MU-TERM + [18]") with a version of MU-TERM where instead of [18] one calls AProVE (with the technique of "Rat + Sect. 4.1") externally. Since MU-TERM generates the polynomial constraints and it only calls AProVE with this set of constraints, the implementation of the criteria from Sect. 3 cannot be used here. In this table, we only ran MU-TERM on a collection of 79 TRSs from the TPDB. These are TRSs where MU-TERM needs rational polynomials in order to succeed with the proof. It turns out that in spite of the external calls, the new SAT-based implementation is indeed significantly faster than the previous non-SAT-based method of [18].

MU-TERM + [18]		MU-TERM + Rat + Sect. 4.1	
Yes	FulTime	Yes	FulTime
62	10.1 s	65	4.1 s

Finally, to measure the usefulness of our contributions in full termination provers, the next table compares the performance of *full* versions of AProVE on all 2061 examples. Here, many termination techniques are used in addition to the dependency graph and reduction pair processor. Moreover, there are also techniques to disprove termination (cf. column "No"). The next table shows that the results of the current paper are also useful when integrating them into such a powerful prover. AProVE-07 is the version which participated in the Termination Competition 2007 (and which won this competition in the category of TRSs). "AProVE-07 + Sect. 4.1" differs from AProVE-07 by using rational polynomials with the setting "Rat + Sect. 4.1" and "AProVE-07 + Sect. 4.1 + Sect. 3" uses "Rat + Sect. 4.1 + Sect. 3" instead. It is interesting to note that when integrating rational polynomials into this full version of AProVE, the criteria of Sect. 3 have quite positive effects. In other words, they reduce the runtimes and hardly affect the power. For details on our experiments (including details on runtimes and

---

[17] We also experimented with different ranges for the coefficients, but the above ranges gave the best results as far as power and runtimes are concerned.

timeouts) and to run "AProVE-07 + Sect. 4.1 + Sect. 3" via a web-interface, we refer to `http://aprove.informatik.rwth-aachen.de/eval/RATPOLO/`.

AProVE-07				AProVE-07 + Sect. 4.1				AProVE-07 + Sect. 4.1 + Sect. 3			
Yes	No	SucTime	FulTime	Yes	No	SucTime	FulTime	Yes	No	SucTime	FulTime
1089	238	3.8 s	29.6 s	1119	238	5.2 s	30.4 s	1118	238	4.9 s	30.1 s

References

1. T. Arts and J. Giesl. Termination of term rewriting using dependency pairs. *Theoretical Computer Science*, 236:133-178, 2000.
2. E. Contejean, C. Marché, B. Monate, and X. Urbain. CiME. `http://cime.lri.fr`.
3. E. Contejean, C. Marché, A. P. Tomás, and X. Urbain. Mechanically proving termination using polynomial interpretations. *Journal of Automated Reasoning*, 34(4):325-363, 2005.
4. N. Dershowitz. Termination of rewriting. *Journal of Symbolic Computation*, 3:69-116, 1987.
5. J. Endrullis. Jambox. Available from `http://joerg.endrullis.de`.
6. J. Endrullis, J. Waldmann, and H. Zantema. Matrix interpretations for proving termination of term rewriting. In *Proc. IJCAR '06*, LNAI 4130, pages 574-588, 2006.
7. C. Fuhs, J. Giesl, A. Middeldorp, R. Thiemann, P. Schneider-Kamp, and H. Zankl. SAT solving for termination analysis with polynomial interpretations. In *Proc. SAT '07*, LNCS 4501, pages 340-354, 2007.
8. A. Gebhardt, D. Hofbauer, and J. Waldmann. Matrix Evolutions. *Proc. WST'07*, 2007.
9. J. Giesl, R. Thiemann, and P. Schneider-Kamp. The dependency pair framework: Combining techniques for automated termination proofs. In *Proc. LPAR '04*, LNAI 3542, pages 301-331, 2005.
10. J. Giesl, P. Schneider-Kamp, R. Thiemann. AProVE 1.2: Automatic termination proofs in the DP framework. In *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. H. Hong and D. Jakuš. Testing positiveness of polynomials. *Journal of Automated Reasoning*, 21(1):23-38, 1998.
15. S. Lucas. MU-TERM: a tool for proving termination of context-sensitive rewriting. In *Proc. RTA '04*, LNCS 3091, pages 200-209, 2004.
16. S. Lucas. Polynomials over the reals in proofs of termination: From theory to practice. *RAIRO Theoretical Informatics and Applications*, 39(3):547-586, 2005.
17. S. Lucas. On the relative power of polynomials with real, rational, and integer coefficients in proofs of termination of rewriting. *Applicable Algebra in Engineering, Communication and Computing*, 17(1):49-73, 2006.
18. S. Lucas. Practical use of polynomials over the reals in proofs of termination. In *Proc. PPDP '07*, ACM Press, pages 39-50, 2007.
19. C. Marché and H. Zantema. The termination competition. In *Proc. RTA '07*, LNCS 4533, pages 303-313, 2007.
20. R. Thiemann and A. Middeldorp. Innermost termination of rewrite systems by labeling. In *Proc. WRS '07*, ENTCS 204, pages 3-19, 2008.