Available online at www.sciencedirect.com
ScienceDirect
Electronic Notes in Theoretical Computer Science
ELSEVIER
Electronic Notes in Theoretical Computer Science 258 (2009) 41–61
www.elsevier.com/locate/entcs

Automatic Proofs of Termination With Elementary Interpretations

Salvador Lucas[1,2]

Departamento de Sistemas Informáticos y Computación
Universidad Politécnica de Valencia
Valencia, Spain

Abstract

Symbolic constraints arising in proofs of termination of programs are often translated into *numeric constraints* before checking them for satisfiability. In this setting, polynomial interpretations are a simple and popular choice. In the nineties, Lescanne introduced the *elementary algebraic interpretations* as a suitable alternative to polynomial interpretations in proofs of termination of term rewriting. Here, not only addition and product but also exponential expressions are allowed. Lescanne investigated the use of elementary interpretations for witnessing satisfiability of a given set of symbolic constraints. He also motivated the usefulness of elementary interpretations in proofs of termination by means of several examples. Unfortunately, he did not consider the *automatic generation* of such interpretations for a given termination problem. This is an important drawback for using these interpretations in practice. In this paper we show how to solve this problem by using a combination of rewriting, CLP, and CSP techniques for handling the elementary constraints which are obtained when giving the symbols *parametric elementary interpretations*.

Keywords: Constraint solving, elementary interpretations, program analysis, termination.

1 Introduction

In this paper, we are interested in termination analysis of term rewriting systems (TRSs [4]) as a suitable basis for approaching termination of more sophisticated programming languages and computational systems. Proofs of termination in term rewriting involve solving *weak* or *strict* symbolic constraints $s \succeq t$ or $s \sqsupset t$ between terms s and t coming from (parts of) the

[1] This work has been partially supported by the EU (FEDER) and the Spanish MEC/MICINN, under grant TIN 2007-68093-C02-02.
[2] Email: {slucas}@dsic.upv.es

doi:10.1016/j.entcs.2009.12.004

rules of the TRS. Here, $\succeq$ and $\sqsupset$ are (quasi-)orderings on terms satisfying appropriate conditions [3,6,9]. These constraints are often treated as *numeric* constraints $[s] \geq [t]$ and $[s] > [t]$ where $[s]$ and $[t]$ are *numeric functions* obtained by *interpreting* the symbols f occurring in s and t as numeric functions $[f]$. Termination tools that aim at achieving *automatic* termination proofs have to *compute* such interpretations out from the *symbolic* constraints associated to the termination problem. In this setting, polynomial interpretations are a widely used choice. In this approach, k-ary symbols $f \in \mathcal{F}$ are given *parametric* polynomials $[f]$. For instance, consider the usual rules for the addition:

$$add(X, 0) \to X \tag{1}$$
$$add(X, s(Y)) \to s(add(X, Y)) \tag{2}$$

The following (parametric) polynomials are given to the symbols in the system:

$$[0] = z_0 \qquad [s](x) = s_1 x + s_0 \qquad [add](x, y) = a_1 x + a_2 y + a_0$$

Variables in terms s and t (e.g., X and Y in our example) become *universally quantified* numeric variables in polynomial constraints $[s] \geq [t]$ or $[s] > [t]$. In contrast, the parametric coefficients $a_0, a_1, \ldots, z_0$ become *existentially quantified variables*. For instance, following [9], in order to prove termination of $\mathcal{R}$, we have to ensure that $[s] > [t]$ for all rewrite rules $s \to t$ in $\mathcal{R}$. For (2) we have $[add(X, s(Y))] = a_1 X + a_2 s_1 Y + a_2 s_0 + a_0$ and $[s(add(X, Y))] = s_1 a_1 X + s_1 a_2 Y + s_1 a_0 + s_0$. Consider the constraint

$$\exists a_1, a_2, s_0, s_1, z_0 \in D\, \forall X, Y \in A\; a_1 X + a_2 s_1 Y + a_2 s_0 + a_0 > s_1 a_1 X + s_1 a_2 Y + s_1 a_0 + s_0$$

where D is a (usually small) *domain of coefficients* and A is the *semantic domain* of the interpretation (e.g., $\mathbb{N}$, $[0, +\infty)$, etc.). In order to *solve* this constraint, most termination tools work on semantic domains A of nonnegative numbers and implement Hong and Jakuš' criterion [11,8] to *remove* the universally quantified variables from polynomial constraints to obtain an *existential constraint* like

$$\exists a_1, a_2, s_0, s_1, z_0 \in D\; a_1 \geq s_1 a_1 \wedge a_2 s_1 \geq s_1 a_2 \wedge a_2 s_0 + a_0 > s_1 a_0 + s_0$$

The idea is having an independent comparison of the different *monomials* (w.r.t. the semantic variables only). Then, suitable *constraint solving systems* are used to give specific values to the parametric coefficients [5,8,19].

1.1 Using elementary interpretations

Lescanne introduced and motivated the use of *elementary algebraic interpretations* as an alternative to polynomial interpretations [14,8,7,18] in proofs of

termination of term rewriting [15].

Example 1.1 The following specification $\mathcal{R}$ of the factorial function is a variant of Lescanne's leading example [16, Introduction] (add rules (1) and (2) above).

$$mul(0, X) \to 0 \tag{3}$$
$$mul(s(X), Y) \to add(mul(X, Y), Y) \tag{4}$$
$$fact(0) \to s(0) \tag{5}$$
$$fact(s(X)) \to mul(s(X), fact(X)) \tag{6}$$

Lescanne showed that termination of $\mathcal{R}$ *cannot* be proved by using polynomial interpretations as sketched above. In contrast, $\mathcal{R}$ can be proved terminating by using the following *elementary interpretation*:

$$[0] = 1 \qquad [s](x) = x + 1 \qquad [fact](x) = (2x + 2)^{2x+1}$$
$$[add](x, y) = x + 2y \qquad [mul](x, y) = 2xy + 2x + y$$

This is *compatible* with the rewrite rules $l \to r$ in $\mathcal{R}$, i.e., $[l] > [r]$ for all rules $l \to r$ in $\mathcal{R}$. For instance, $[fact(s(X))] = (2X + 4)^{2X+3}$ and $[mul(s(X), fact(X))] = (2X + 3)(2X + 2)^{2X+1} + 2X + 2$. In Example 5.2 we prove that $[fact(s(X))] > [mul(s(X), fact(X))]$ for all $X \in \mathbb{N}$.

Following the usual practice in the nineties, Lescanne focused on the use of *reduction orderings* $>$ which can be used to prove termination of a TRS by just comparing the left- and right-hand sides of the rules [9]. Lescanne considered elementary interpretations *over the naturals* and his work addresses the problem of *checking* the inequalities $[l] > [r]$ which are obtained for a *given* interpretation which should be provided by the user. In contrast, current state-of-the-art termination tools which use polynomial interpretations: (1) use the dependency pairs (DP) method [3] to generate the constraints to be solved, (2) use polynomial interpretations *over the reals* [18], and (3) such interpretations are *not* given by the user but rather *generated* by the tools.

This paper aims at enabling the use of elementary interpretations in automatic proofs of termination. The contribution of this paper is twofold. In Sections 3 and 4, we investigate elementary interpretations *over the reals* and show how to translate the standard requirements of the DP-method into symbolic *elementary* constraints over the reals. In particular, the generation of *reduction pairs* $(\gtrsim, \sqsupset)$ (where $\gtrsim$ is a monotonic quasi-ordering[3] and $\sqsupset$ is a well-founded ordering on terms) is considered. These are the basic components of the DP-method for building proofs of termination. We also consider

[3] A quasi-ordering is a reflexive and transitive relation.

the generation of reduction orderings. Our development admits the specification of *monotonicity* conditions which must be satisfied by the orderings. This provides a more flexible framework enabling more applications. For instance, we could want to impose such restrictions to prove termination of variants of term rewriting like context-sensitive rewriting, infinitary rewriting, innermost rewriting, outermost rewriting, etc. (see [18] for further motivation).

Our second contribution is the definition of a rule-based transformation system for solving (parametric) elementary constraints (Sections 5, 6 and 7). This is an essential part of the work which does not depend on our main application focus (termination of programs), thus being useful for dealing with elementary constraints arising from other problems. Section 8 concludes.

2 Preliminaries

A binary relation R on a set A is *terminating* (or well-founded) if there is no infinite sequence $a_1\ R\ a_2\ R\ a_3 \cdots$. Given $f : A^k \to A$ and $i \in \{1, \ldots, k\}$, we say that R is monotonic on the i-th argument of f (or that f is i-monotone regarding R) if $f(x_1, \ldots, x_{i-1}, x, \ldots, x_k)\ R\ f(x_1, \ldots, x_{i-1}, y, \ldots, x_k)$ whenever $x\,R\,y$, for all $x, y, x_1, \ldots, x_k \in A$. We say that R is *monotonic* regarding f (or that f is R-monotone) if R is i-monotonic on the i-th argument of f for all i, $1 \leq i \leq k$. A transitive and reflexive relation $\succeq$ on A is a quasi-ordering. A transitive and irreflexive relation $>$ on A is an ordering.

In this paper, $\mathcal{X}$ denotes a countable set of variables and $\mathcal{F}$ denotes a signature, i.e., a set of function symbols $\{\mathtt{f}, \mathtt{g}, \ldots\}$, each having a fixed arity given by a mapping $ar : \mathcal{F} \to \mathbb{N}$. The set of terms built from $\mathcal{F}$ and $\mathcal{X}$ is $\mathcal{T}(\mathcal{F}, \mathcal{X})$. A *context* is a term $C[\,]$ with a 'hole' (formally, a fresh constant symbol). A binary relation R on terms is *stable* if, for all terms s, t and substitutions σ, $\sigma(s)\ R\ \sigma(t)$ whenever $s\ R\ t$.

A rewrite rule is an ordered pair (l, r), written $l \to r$, with $l, r \in \mathcal{T}(\mathcal{F}, \mathcal{X})$, $l \notin \mathcal{X}$ and $\mathcal{V}ar(r) \subseteq \mathcal{V}ar(l)$. A TRS is a pair $\mathcal{R} = (\mathcal{F}, R)$ where R is a set of rewrite rules. The problem of proving termination of a TRS is equivalent to finding a well-founded, stable, and monotonic (strict) ordering $>$ on terms (i.e., a *reduction ordering*) which is *compatible* with the rules of the TRS, i.e., $l > r$ for all $l \to r \in R$ [9]. Termination of rewriting can also be proved by using the dependency pairs approach [3]. Reduction pairs are used in this case. A reduction pair $(\gtrsim, \sqsupset)$ consists of a stable and weakly monotonic quasi-ordering $\gtrsim$, and a stable and well-founded ordering $\sqsupset$ satisfying either $\gtrsim \circ \sqsupset \subseteq \sqsupset$ or $\sqsupset \circ \gtrsim \subseteq \sqsupset$. No *monotonicity is required* for $\sqsupset$. The quasi-ordering $\gtrsim$ is used to compare the rules of the TRS and the strict ordering $\sqsupset$ is used to compare the *dependency pairs*, see [3] for further details.

Term (quasi-)orderings can be obtained by giving appropriate *interpreta-*

tions to the function symbols of a signature. Given a signature $\mathcal{F}$, an $\mathcal{F}$-algebra is a pair $\mathcal{A} = (\mathsf{A}, \mathcal{F}_{\mathsf{A}})$, where A is a set and $\mathcal{F}_{\mathsf{A}}$ is a set of mappings $f_{\mathcal{A}} : \mathsf{A}^k \to \mathsf{A}$ for each $f \in \mathcal{F}$ where $k = ar(f)$. For a given valuation mapping $\alpha : \mathcal{X} \to \mathsf{A}$, the evaluation mapping $[\alpha]_{\mathcal{A}} : \mathcal{T}(\mathcal{F}, \mathcal{X}) \to \mathsf{A}$ is inductively defined by $[\alpha]_{\mathcal{A}}(x) = \alpha(x)$ if $x \in \mathcal{X}$ and $[\alpha]_{\mathcal{A}}(f(t_1, \ldots, t_k)) = f_{\mathcal{A}}([\alpha]_{\mathcal{A}}(t_1), \ldots, [\alpha]_{\mathcal{A}}(t_k))$ for $x \in \mathcal{X}$, $f \in \mathcal{F}$, and $t_1, \ldots, t_k \in \mathcal{T}(\mathcal{F}, \mathcal{X})$. Given a term t with $\mathcal{V}ar(t) = \{x_1, \ldots, x_n\}$, we write $[t]_{\mathcal{A}}$ (or just $[t]$ if $\mathcal{A}$ is clear from the context) to denote the *function* $F_t : \mathsf{A}^n \to \mathsf{A}$ given by $F_t(a_1, \ldots, a_n) = [\alpha_{(a_1,\ldots,a_n)}]_{\mathcal{A}}(t)$ for each tuple $(a_1, \ldots, a_n) \in \mathsf{A}^n$, where $\alpha_{(a_1,\ldots,a_n)}(x_i) = a_i$ for $1 \leq i \leq n$.

We want to use real functions to define term (quasi-) orderings [18]. Given a signature $\mathcal{F}$, an interval $\mathsf{A} \subseteq [0, +\infty)$ (usually $\mathsf{A} = [0, +\infty)$), and an $\mathcal{F}$-algebra over the reals $\mathcal{A} = (\mathsf{A}, \mathcal{F}_{\mathsf{A}})$, $\gtrsim$ given by $t \gtrsim s \Leftrightarrow \forall \alpha : \mathcal{X} \to \mathsf{A}, [\alpha](t) - [\alpha](s) \geq_{\mathbb{R}} 0$ for all $t, s \in \mathcal{T}(\mathcal{F}, \mathcal{X})$ is a stable quasi-ordering on terms. Given $\delta > 0$, the relation $>_\delta$ on terms given by $t >_\delta s \Leftrightarrow \forall \alpha : \mathcal{X} \to \mathsf{A}, [\alpha]_{\mathcal{A}}(t) - [\alpha]_{\mathcal{A}}(s) \geq_{\mathbb{R}} \delta$ is a *well-founded* strict ordering on terms. As discussed in [18], rather than imposing the monotonicity requirements to all arguments of all function symbols, we use sufficient conditions ensuring the monotonicity of either $\gtrsim$ or $>_\delta$ for a given argument $i \in \{1, \ldots, k\}$ of a given k-ary symbol f. Then, we speak of i-monotonicity of $\gtrsim$ (or *weak* i-monotonicity) or i-monotonicity of $>_\delta$ (*strong* i-monotonicity) for a given symbol f: $\frac{\partial f_{\mathcal{A}}}{\partial x_i} \geq 0$ ensures weak i-monotonicity of f [18, Proposition 2] and $\frac{\partial f_{\mathcal{A}}}{\partial x_i} \geq 1$ ensures strong i-monotonicity of f [18, Theorem 2]. If $\gtrsim$ is guaranteed to be weakly monotonic (for all arguments $i \in \{1, \ldots, k\}$ of all k-ary symbols $f \in \mathcal{F}$), then $(\gtrsim, >_\delta)$ is a reduction pair for all $\delta > 0$ [18, Proposition 4]. If $>_\delta$ is strongly monotonic (again for all arguments and symbols), then $>_\delta$ is a reduction ordering.

3 Elementary interpretations

Given a set of numbers N, the following grammar describes Lescanne's EP-terms (or $EP(N)$-terms if we want to make N explicit) [4] in [16, Section 5]:

$$E := x \mid n \mid E^E \mid E + E \mid E \cdot E$$

where $n \in N$ and x are numeric variables.

Remark 3.1 Lescanne's description of EP-terms makes use of the numeric constants 0 and 1 only. This is due to his particular representation of EP-terms, where an 'EP-monomial' like $3x$ is written $x + x + x$, thus avoiding the explicit use of the constant 3. We do not follow this approach because:

[4] Our presentation of EP-terms is slightly different from Lescanne's, but equivalent.

(i) We do not assume that the coefficients are given; we rather want to *compute* them through some constraint solving procedure. Hence, they are treated as *unknowns* and represented by means of parameters.

(ii) The values in N can be *real* (possibly negative) numbers. Hence, representing a monomial coefficient as a repetition of the monomial is not feasible anymore.

Remark 3.2 [Use of real numbers] Although natural numbers are closed under exponentiation, addition and product, this is *not* the case for other prominent subsets of (nonnegative) real numbers. For instance, Hilbert conjectured that a^b is transcendental whenever $a \notin \{0, 1\}$ is algebraic and b is an irrational algebraic number (this is part of his seventh problem, see [10] for instance). Gelfond-Schneider's Theorem confirms that this is the case. In particular $\mathbb{Q}$ is *not* closed under exponentiation. For instance, $2^{(2^{\frac{1}{2}})} = 2^{\sqrt{2}}$ is transcendental. Hence, in sharp contrast with polynomial interpretations (see [19]), *transcendental* numbers are essential to deal with $\mathcal{F}$-algebras based on EP-functions over domains of real numbers.

4 Linear elementary interpretations

The size and structural complexity of the parametric constraints which are obtained during the automatic treatment of termination problems highly depend on the shape of the parametric functions which are given to function symbols in the interpretation. The usual choice in termination provers that rely on polynomial interpretations is using *linear polynomials*. In this section, we introduce and investigate a subclass of elementary functions which is based on using linear polynomials in the additive and exponential components of the elementary functions: the *Linear Elementary Interpretations*. Each k-ary symbol $f \in \mathcal{F}$ is given a function

$$f(x_1, \ldots, x_k) = A(x_1, \ldots, x_k) + B(x_1, \ldots, x_k)^{C(x_1,\ldots,x_k)} \tag{7}$$

where $A = a_1x_1 + \cdots + a_kx_k + a_0$, $B = b_1x_1 + \cdots + b_kx_k + b_0$, and $C = c_1x_1 + \cdots + c_kx_k + c_0$ are linear polynomials over the reals.

Remark 4.1 Special cases for f, depending on the shape of A, B and C are:

(i) If A is zero ($A \equiv 0$), then $f = B^C$ is a 'pure exponential' interpretation.

(ii) If B is a constant b, then $f = A + b^C$ and, whenever $b > 0$ we can use negative coefficients in C without any problem.

(iii) If C is a constant c, then f is either a possibly non-linear polynomial $f = A + B^c$ (if $c > 0$ is a positive integer) or a polynomial *fraction* $f = A + \frac{1}{B^{|c|}}$ (if $c < 0$ is a negative integer). B could contain negative coefficients also.

In this section, we assume that $f(x_1,\ldots,x_k)$ is a linear elementary function (7) and formulate sufficient conditions to guarantee algebraicity and monotonicity of linear elementary interpretations on the semantic domain $\mathsf{A} = [0,+\infty)$. Conveniently, our results are formulated as *constraints* involving the parametric coefficients $a_0,\ldots,a_k,b_0,\ldots,b_k,c_0,\ldots,c_k$ only. The following proposition about constraints of linear polynomials is used below (see also [11]).

Proposition 4.2 *Let $P = a_1x_1 + \cdots + a_nx_n + a_0$ be a linear polynomial and $\alpha \in [0,\infty)$. Then, $\forall x_1,\ldots,x_n \geq 0, P(x_1,\ldots,x_n) \geq \alpha$ holds if and only if $\forall i, 1 \leq i \leq n$, $a_i \geq 0$ and $a_0 \geq \alpha$.*

4.1 Algebraicity of linear elementary interpretations

If the exponent polynomial $C(x_1,\ldots,x_k)$ takes a noninteger value $c \in \mathbb{R} - \mathbb{Z}$ for some $(x_1,\ldots,x_k) \in \mathsf{A}^k$, then we have to ensure that $B(x_1,\ldots,x_k) \geq 0$ for the base polynomial B. otherwise, B^C would be undefined for some points in A^k. Thus, we require that $B(x_1,\ldots,x_k) \geq 0$ for all $x_1,\ldots,x_k \in \mathsf{A}$. By Proposition 4.2, this is equivalent to impose $b_i \geq 0$ for all i, $0 \leq i \leq k$. We also need to ensure that $f(x_1,\ldots,x_k) \geq 0$ for all $x_1,\ldots,x_k \geq 0$ (algebraicity). In general, this amounts at requiring that A is non-negative: for all $x_1,\ldots,x_k \geq 0$, $A(x_1,\ldots,x_k) \geq 0$. Again, this is equivalent to impose $a_i \geq 0$ for all i, $0 \leq i \leq k$.

In the following section, we investigate monotonicity of linear elementary functions. First of all, we have, for each i, $1 \leq i \leq k$,

$$\frac{\partial f}{\partial x_i} = \frac{\partial A}{\partial x_i} + B^C \cdot \left(\ln(B) \cdot \frac{\partial C}{\partial x_i} + \frac{C}{B} \cdot \frac{\partial B}{\partial x_i}\right) = a_i + B^C \cdot \left(c_i \ln(B) + b_i \frac{C}{B}\right)$$

which is well-defined only if $B > 0$, i.e., if $b_0 > 0$. We need to consider two relevant monotonicity conditions for each argument i of f: $\frac{\partial f}{\partial x_i} \geq 0$ (weak monotonicity) and $\frac{\partial f}{\partial x_i} \geq 1$ (strong monotonicity).

4.2 Weak monotonicity of linear elementary interpretations

The following proposition provides a sufficient condition for weak i-monotonicity of linear elementary functions.

Proposition 4.3 *Let f be a linear elementary k-ary function and $i \in \{1,\ldots,k\}$. If $a_i \geq 0$, $b_0 > 0$, $c_ib_0 + b_ic_0 - c_i \geq 0$ and $\bigwedge_{j=1}^{k} c_ib_j + b_ic_j \geq 0$, then f is weakly i-monotone over $\mathsf{A} = [0,+\infty)$.*

Corollary 4.4 *Let f be a linear elementary k-ary function and $i \in \{1,\ldots,k\}$. Assume that $a_i \geq 0$, $b_0 \geq 1$ and $c_i \geq 0$. Then, f is weakly i-monotone over $\mathsf{A} = [0,+\infty)$ if $C \geq 0$ or $b_i = 0$.*

Example 4.5 Consider the following linear elementary functions $f(x,y) = (2y+4)^{4x-y+1}$ and $g(x,y) = x + (\frac{x}{2} + y + 1)^{x-2y}$. Both of them are weakly 1-monotonic:

(i) We can apply Corollary 4.4 to f: $a_1 = 0$, $b_0 = 4 \geq 1$, $c_1 = 4 \geq 0$, and $b_1 = 0$.

(ii) Corollary 4.4 does not apply to g, but Proposition 4.3 does: $a_1 = 1 \geq 0$, $b_0 = 1 > 0$, $c_1 b_0 + b_1 c_0 - c_1 = 0$, $c_1 b_1 + b_1 c_1 = 1 \geq 0$, $c_1 b_2 + b_1 c_2 = 0$.

4.3 Strong monotonicity of linear elementary interpretations

Regarding strong monotonicity, we have the following:

Proposition 4.6 *Let f be a linear elementary k-ary function and $i \in \{1,\ldots,k\}$. If*

(i) *$a_i \geq 1$, $b_0 > 0$, $c_i b_0 + b_i c_0 - c_i \geq 0$ and $\bigwedge_{j=1}^{k} c_i b_j + b_i c_j \geq 0$, or*

(ii) *$a_i \geq 0$, $b_0 \geq 1$, $b_i = 0$, $C \geq 0$, $c_i \ln(b_0) \geq 1$, or*

(iii) *$a_i \geq 0$, $b_0 \geq 1$, $c_i \geq 0$, and $\bigwedge_{j=0}^{k} b_i c_j \geq b_j$,*

then f is strongly i-monotone over $\mathsf{A} = [0,+\infty)$.

Example 4.7 Consider the linear elementary function $fact(x) = (2x+2)^{2x+1}$ in Example 1.1. We can prove strong 1-monotonicity of $fact$ by using Proposition 4.6(iii): $a_1 = 0$, $b_0 = 2 \geq 1$, $c_1 = 2 \geq 0$, $b_1 c_0 = 2 \cdot 1 \geq 2 = b_0$ and $b_1 c_1 = 4 \geq 2 = b_1$.

Remark 4.8 [Use of negative coefficients] Regarding the possibility of using negative coefficients in (arbitrary) linear elementary interpretations, we know that this is possible in the exponent C only. If we use Proposition 4.6 to guarantee some non-trivial degree of strong monotonicity for a k-ary function f (i.e., at least one of the arguments $i \in \{1,\ldots,k\}$ is intended to be strongly monotonic), only the first condition is compatible with such negative coefficients.

The following result avoids the logarithmic constraint in Proposition 4.6(ii).

Corollary 4.9 *Let f be a linear elementary k-ary function and $i \in \{1,\ldots,k\}$. If $b_0 \geq 1$, $b_i = 0$ and*

(i) *$a_i \geq 1$ and $c_i \geq 0$, or*

(ii) *$a_i \geq 0$, $\bigwedge_{j=0}^{k} c_j \geq 0$, and either $b_0 \geq e$ and $c_i \geq 1$ or $c_i b_0 \geq b_0 + c_i$,*

then, f is strongly i-monotone over $\mathsf{A} = [0,+\infty)$.

When using Corollary 4.9 in practice, instead of imposing $b_0 \geq e$ we rather use a suitable upper approximation to $e = 2.7182\cdots$ as in $b_0 \geq 3$.

5 Solving elementary constraints

Lescanne compares elementary expressions e and e' by using rewrite systems which either *preserve* its value or *decrease* it. His first group of rules ($\mathcal{R}$ in [16]) encodes well-known arithmetic properties of addition, product and exponentials:

$$\begin{array}{lll} 0 + x \to x & x \cdot (y + z) \to (x \cdot y) + (x \cdot z) & x^{y+z} \to x^y \cdot x^z \\ 0 \cdot x \to 0 & (x^y)^z \to x^{(y \cdot z)} & x^1 \to x \\ 1 \cdot x \to x & (x \cdot y)^z \to x^z \cdot y^z & x^0 \to 1 \end{array}$$

Actually, they can be used in both directions (as *equations*): every arithmetic expression e (in particular any EP-expression) which is rewritten using these rules yields an equivalent expression e': $e \to e'$ means that $[\![e]\!] = [\![e']\!]$, where $[\![\ ]\!]$ is the (intended) interpretation of elementary expressions which is obtained when variables x, y, z, constants $0, 1$, and operations '$+$', '$\cdot$', and exponential are interpreted as arithmethic variables, constants, and operations in the usual way.

The following rewrite rule ($\mathcal{H}$ in [16]) encodes a semantic transformation which yields an expression e' which is *smaller* than the original one:

$$(x + y)^z \hookrightarrow x^z + y^z$$

That is: $e \hookrightarrow e'$ implies that $\forall x_1, \ldots, x_n \in \mathsf{A}, [\![e]\!] \geq [\![e']\!]$, where $x_1, \ldots, x_n$ are the variables occurring in e and $\mathsf{A} = \{2, 3, \ldots\}$ in [16]. Roughly speaking, in order to check that $e > e'$ holds, Lescanne performs arbitrary rewrite steps on e and e' using $\mathcal{R}$. He only uses $\mathcal{H}$ to rewrite the left-hand side e of the inequality: if $e \hookrightarrow e''$, then $[\![e]\!] \geq [\![e'']\!]$; so, if we are able to prove $[\![e'']\!] > [\![e']\!]$ later, then $[\![e]\!] > [\![e']\!]$ as desired. The idea is reaching in this way a final constraint $\overline{e} > \overline{e'}$ whose satisfaction is easily established (see [16] for details).

5.1 Auxiliary results

The following result generalizes to *real* numbers the result encoded by rule $\mathcal{H}$ for natural numbers.

Proposition 5.1 *Let $x, y \geq 0$ and $z \geq 1$. Then, $(x+y)^z \geq x^z + y^z$. Furthermore, if $x, y > 0$ and $z > 1$, then $(x + y)^z > x^z + y^z$.*

Example 5.2 (Continuing Example 1.1) By Proposition 5.1, for all $X \geq 0$, $(2X+2+2)^{2X+2} > (2X+2)^{2X+2} + 2^{2X+2}$ holds. Thus, we have:

$$\begin{aligned}[fact(s(X))] &= (2X+4)^{2X+3} = (2X+4)(2X+2+2)^{2X+2} \\ &> (2X+4)\left((2X+2)^{2X+2} + 2^{2X+2}\right) \\ &= (2X+4)(2X+2)^{2X+2} + (2X+4)2^{2X+2}\end{aligned}$$

Since $2X+4 > 2X+3$, $(2X+2)^{2X+2} > (2X+2)^{2X+1}$, and $(2X+4)2^{2X+2} > 2X+2$ we have

$$\begin{aligned}(2X+4)(2X+2)^{2X+2} + (2X+4)2^{2X+2} &> (2X+3)(2X+2)^{2X+1} + 2X + 2 \\ &= [mul(s(X), fact(X))]\end{aligned}$$

Thus, we conclude that $[fact(s(X))] > [mul(s(X), fact(X))]$.

The following result complements Proposition 5.1.

Proposition 5.3 *Let $b, x, x_1, \ldots, x_n \in \mathbb{R}$ be such that $x_i \geq 1$ for all $1 \leq i \leq n$, $x \geq \sum_{i=1}^{n} x_i$, and $b \geq 2$. Then, $b^x \geq \sum_{i=1}^{n} b^{x_i}$. If $n > 1$ and $x_j > 1$ for some $1 \leq j \leq n$, then $b^x > \sum_{i=1}^{n} b^{x_i}$.*

5.2 Removing universal quantification from elementary constraints

When using a *parametric* linear elementary algebra $\mathcal{A} = (\mathsf{A}, \mathcal{F}_{\mathsf{A}})$ to solve a conjunction $\bigwedge_{i=1}^{m} s_i \succeq t_i \wedge \bigwedge_{i=j}^{n} u_j \sqsupset v_j$ of symbolic constraints, we obtain a sentence

$$\exists \mathsf{c}_1, \ldots, \mathsf{c}_\kappa \in D\ \forall x_1, \ldots, x_n \in \mathsf{A}\ \bigwedge_{i=1}^{m} [s_i] \geq [t_i] \wedge \bigwedge_{i=j}^{n} [u_j] >_\delta [v_j]$$

(for some $\delta > 0$). Alternatively, we can leave δ unspecified and include a new (existentially quantified) parameter D:

$$\exists \mathsf{D} > 0\ \exists \mathsf{c}_1, \ldots, \mathsf{c}_\kappa \in D\ \forall x_1, \ldots, x_n \in \mathsf{A}\ \bigwedge_{i=1}^{m} [s_i] \geq [t_i] \wedge \bigwedge_{i=j}^{n} [u_j] \geq [v_j] + \mathsf{D}$$

Now we have to *witness* that this sentence is satisfiable, i.e., we have to obtain a value assignment $\gamma : \mathsf{K} \cup \{\mathsf{D}\} \to D \cup (0, +\infty)$, where $\mathsf{K} = \{\mathsf{c}_1, \ldots, \mathsf{c}_\kappa\}$ is the set of parametric coefficients that we are considering (which take values on D only) and D is the new parameter which takes values in $(0, +\infty)$. We have to do this in such a way that $\forall x_1, \ldots, x_n \in \mathsf{A}\ \bigwedge_{i=1}^{m} [s_i]_\gamma \geq [t_i]_\gamma \wedge \bigwedge_{i=j}^{n} [u_j]_\gamma >_\delta [v_j]_\gamma$ holds. Here, $[\cdot]_\gamma$ is the intepretation of terms which is obtained by using the

linear elementary algebra $\mathcal{A}_\gamma = (\mathsf{A}, \mathcal{F}_{A,\gamma})$ where the mappings $f_{\mathcal{A}_\gamma}$ in $\mathcal{F}_{A,\gamma}$ are obtained from those in $\mathcal{F}_A$ by giving the value $\gamma(\mathsf{c}_i)$ to each parameter c_i occurring in $f_A \in \mathcal{F}_A$. Furthermore, $\delta = \gamma(\mathsf{D})$ (if we use the second alternative sentence above). For instance, we can start with a sentence

$$\exists a, b, c, d, e, a', b', c', d', e', f' \in D\, \forall X \in \mathsf{A}\, (aX+b)^{cX+d} \geq (e'X+f')(a'X+b')^{c'X+d'}$$

As remarked in the introduction, we proceed by transforming this problem into an *existential constraint solving problem*

$$\exists \mathsf{c}_1, \ldots, \mathsf{c}_\kappa \in D \bigwedge_{i=1}^{N} e_i \bowtie e'_i$$

where e_i and e'_i are elementary expressions built out from parametric coefficients $\mathsf{c}_1, \ldots, \mathsf{c}_\kappa$ and numeric constants only. Let $\mathsf{E}(\mathsf{c}_1, \ldots, \mathsf{c}_\kappa)$ be the set of such expressions. Furthermore, $\bowtie$ is a comparison operator ($\geq$, $>_\delta$, ...). Note that:

(i) The *two* kinds of variables (parametric coefficients and semantic variables) have different roles in the constraints (existential vs. universal quantification).

(ii) As discussed in Remark 3.1, Lescanne's technique assumes that the coefficients of the monomials are *implicit*. Of course, this is not compatible neither with obtaining values for such coefficients by solving existential constraints involving them, nor with coefficients taking values over the rationals.

(iii) Many important properties about elementary constraints over the reals are valid under some *conditions* only. For instance, $(x + y)^z \geq x^z + y^z$ (which corresponds to the rule $\mathcal{H}$ above), is guaranteed only if $x, y \geq 0$ and $z \geq 1$ (Proposition 5.1). If we want to use this property, we need to be able to *introduce* these new *proof obligations* as auxiliary constraints which have to be solved together with the 'main' ones. This does not fit standard term rewriting anymore.

(iv) The ability of handling constraints with parametric coefficients gives us more flexibility. For instance, the constraint above can be *transformed* by introducing a fresh constant $\overline{d}$ defined by $\overline{d} + 1 = d$, leading to the equivalent constraint $(aX+b)^{cX+\overline{d}+1} \geq (e'X+f')(a'X+b')^{c'X+d'} \wedge \overline{d}+1 = d$ which can be equivalently rewritten (using $\mathcal{R}$ above) into $(aX + b)(aX + b)^{cX+\overline{d}} \geq (e'X + f')(a'X + b')^{c'X+d'} \wedge \overline{d} + 1 = d$. Now, this can be decomposed as follows: $aX + b \geq e'X + f' \wedge aX + b \geq a'X + b' \wedge cX + \overline{d} \geq c'X + d' \wedge \overline{d} + 1 = d$. This *linear* constraint can be transformed now by using Hong and Jakuš' criterion into $a \geq e' \wedge b \geq f' \wedge a \geq a' \wedge b \geq b' \wedge c \geq$

$$c' \wedge \overline{d} \geq d' \wedge \overline{d} + 1 = d.$$

In Figure 1 we introduce a new rule-based transformation system for checking and solving (parametric) elementary constraints.

5.2.1 Description of the transformation system.

Letters U, V, W, X, Y, and Z in Figure 1 denote arbitrary elementary expressions whereas $\mathcal{C}[\,]$ denotes a *context*. Making contexts explicit in the definition of the rules is in sharp contrast both with pure rewriting and CLP. Alternatively, we could provide the usual structural or congruence rules to propagate reductions on the syntactic structure of the constraint. Note, however, that the definition of the three rules in the **Exponentials** section is intentionally asymmetric: only the left-hand sides e of constraints $e \geq e'$ or $e > e'$ can be transformed by using these rules; otherwise, we could obtain a wrong approximation of the original constraint.

The meaning of the rules should be clear: they mostly rely on well-known arithmetic properties of addition, product and exponential (over the reals). Rule Add basis corresponds to Proposition 5.1; rule Add exp corresponds to Proposition 5.3 and rule Negative Exp is obvious (and necessary to deal with negative exponents).

The **Introduction** rules give support to the use of the ordering $>_\delta$ in two different ways: by either providing an *explicit* (positive) value for δ or (better) by leaving it unspecified. In the second case, a reserved parameter D (which cannot be used in the parametric interpretation) is intended to represent the appropriate value for δ. This value would be obtained together with all other parameters at the end of the process. The rule K-Introd allows us to replace basic expressions by other basic expressions to our convenience. As suggested above, this can be very useful but we must be careful when using this rule because it can easily run into a nonterminating behavior. Furthermore, we need to provide the appropriate 'conjecture' K' which leads to some progress in the deduction and also select the appropriate target K for the replacement.

Decomposition rules play a prominent role in the system: they introduce a *structural simplification* of the constraints on the basis of new comparisons between the arguments of the arithmetic operators: addition, product and exponential.

Finally, the **Constraints** rules give support to the simplification of goals by moving a basic constraint from the goal to the constraint part.

Remark 5.4 [Generalizing Hong and Jakuš' criterion] Hong and Jakuš' criterion for removing semantic variables from polynomial constraints is easily implemented by using Add decomp, Prod decomp and the constraint removal rules Constants, Variables, and Reflexivity. Thus, our system generalizes Hong

and Jakuš' criterion to parametric elementary constraints: universally quantified semantic variables are removed while an existential constraint consisting of parametric coefficients (only) is built to subsequently invoke an appropriate solver.

5.2.2 Transforming constraints.

As in CLP [12,13,21], we rewrite *states* $\langle \mathsf{G} \mid \mathsf{C} \rangle$ consisting of a goal G which is a conjunction of elementary constraints involving both parametric coefficients and semantic variables, and a constraint C which is a conjunction of elementary constraints involving parametric coefficients only. We rewrite such states as follows: write $\langle \mathsf{G} \mid \mathsf{C} \rangle \Rightarrow \langle \mathsf{G}' \mid \mathsf{C}' \rangle$ if either $\mathsf{G} \rightarrow_{\mathcal{R}} \mathsf{G}'$ (modulo associativity and commutativity of addition and product) and $\mathsf{C} = \mathsf{C}'$, or else one of the rules in Figure 1 applies in the usual way (see [21]). A computation from an initial state $\langle \mathsf{G} \mid True \rangle$ ends when either a state $\langle \Box \mid \mathsf{C} \rangle$ is reached (successful computation), or a state $\langle \mathsf{G}' \mid \mathsf{C}' \rangle$, where no further rewriting step on G' is possible, is obtained (failed computation). In case of a successful computation C is an existential constraint $(\exists \mathsf{c}_1, \ldots, \mathsf{c}_\kappa \in D\ \mathsf{C})$ which we can try to solve by using an appropriate constraint solving system [22]. In general, C is an elementary constraint, but we often obtain polynomial constraints. The variable assignment $\{\mathsf{c}_i \mapsto v_i \mid 1 \leq i \leq \kappa, v_i \in D\} \cup \{\mathsf{D} \mapsto \delta\}$ (for some $\delta > 0$) which solves C and which represents an specific elementary interpretation which is compatible with all the requirements of the termination problem would be returned to the user.

Theorem 5.5 (Correctness) *Let* $\mathsf{G} = \bigwedge_{i=1}^{N} e_i \bowtie e'_i$ *be such that* e_i, e'_i *are elementary expressions with parameters* $\mathsf{c}_1, \ldots, \mathsf{c}_\kappa$ *and variables* $x_1, \ldots, x_n$ *for all* i, $1 \leq i \leq N$, *and* $\bowtie \in \{\geq, >, >_\delta\}$. *If* $\langle \mathsf{G} \mid True \rangle \Rightarrow^* \langle \Box \mid \mathsf{C} \rangle$, *then* $\exists \mathsf{c}_1, \ldots, \mathsf{c}_\kappa \in D\ \forall x_1, \ldots x_n \in \mathsf{A}\ \mathsf{G}$ *holds if* $\exists \mathsf{c}_1, \ldots, \mathsf{c}_\kappa \in D\ \mathsf{C}$ *holds.*

5.2.3 Termination.

Lescanne's system $\mathcal{R}$ is terminating. The rules in Figure 1 are terminating if we do not use K-Introd, which introduces new expressions $K \in \mathsf{E}(\mathsf{c}_1, \ldots, \mathsf{c}_\kappa)$. As discussed in Section 5.2, this rule is useful to force a given expression to adopt some particular shape which enables the application of other rules leading to further progress in the derivation (see the second example in the next section). Therefore, we should use it only under control of an appropriate heuristic.

Exponentials

(Add basis) $\langle \mathcal{C}[(U+V)^W] \bowtie X \mid \mathsf{C}\rangle \Rightarrow \langle U \geq 0 \wedge V \geq 0 \wedge W \geq 1 \wedge \mathcal{C}[U^W + V^W] \bowtie X \mid \mathsf{C}\rangle \quad \bowtie \in \{\geq, >\}$

(Add exp) $\langle \mathcal{C}[U^{V+W}] \bowtie X \mid \mathsf{C}\rangle \Rightarrow \langle U \geq 2 \wedge V \geq 1 \wedge W \geq 1 \wedge \mathcal{C}[U^V + U^W] \bowtie X \mid \mathsf{C}\rangle \quad \bowtie \in \{\geq, >\}$

(Negative Exp) $\langle U \bowtie V^{-W} \mid \mathsf{C}\rangle \Rightarrow \langle U \cdot V^W \bowtie 1 \wedge V > 0 \mid \mathsf{C}\rangle \quad \bowtie \in \{\geq, >\}$

Introduction

(Parametric delta) $\langle X >_\delta Y \mid \mathsf{C}\rangle \Rightarrow \langle X \geq Y + \mathsf{D} \mid \mathsf{C} \wedge \mathsf{D} > 0\rangle$ where D is a reserved parameter

(Explicit delta) $\langle X >_{\mathsf{d}} Y \mid \mathsf{C}\rangle \Rightarrow \langle X \geq Y + \mathsf{d} \mid \mathsf{C}\rangle$ where d is a positive number

(K-Introd) $\langle \mathcal{C}[K] \mid \mathsf{C}\rangle \Rightarrow \langle \mathcal{C}[K'] \mid \mathsf{C} \wedge K = K'\rangle$ if $K, K' \in \mathsf{E}(\mathsf{c}_1, \ldots, \mathsf{c}_\kappa)$

Decomposition

(Add decomp) $\langle U + V \bowtie X + Y \wedge \mathsf{G} \mid \mathsf{C}\rangle \Rightarrow \langle U \geq X \wedge V \bowtie Y \wedge \mathsf{G} \mid \mathsf{C}\rangle \quad \bowtie \in \{\geq, >\}$

(Prod decomp) $\langle U \cdot V \bowtie X \cdot Y \wedge \mathsf{G} \mid \mathsf{C}\rangle \Rightarrow \langle U \bowtie X \wedge V \bowtie Y \wedge \mathsf{G} \mid \mathsf{C}\rangle \quad \bowtie \in \{\geq, >\}$

(Exp decomp) $\langle U^V \bowtie X^Y \wedge \mathsf{G} \mid \mathsf{C}\rangle \Rightarrow \langle U \bowtie X \wedge V \geq Y \wedge U \geq 1 \wedge Y \bowtie 0 \wedge \mathsf{G} \mid \mathsf{C}\rangle \quad \bowtie \in \{\geq, >\}$

Constraint

(Constants) $\langle K \bowtie L \wedge \mathsf{G} \mid \mathsf{C}\rangle \Rightarrow \langle \mathsf{G} \mid \mathsf{C} \wedge K \bowtie L\rangle$ if $K, L \in \mathsf{E}(\mathsf{c}_1, \ldots, \mathsf{c}_\kappa)$, $\bowtie \in \{=, \geq, >\}$

(Variables) $\langle x \geq 0 \wedge \mathsf{G} \mid \mathsf{C}\rangle \Rightarrow \langle \mathsf{G} \mid \mathsf{C}\rangle$

(Reflexivity) $\langle x \geq x \wedge \mathsf{G} \mid \mathsf{C}\rangle \Rightarrow \langle \mathsf{G} \mid \mathsf{C}\rangle \qquad \langle x = x \wedge \mathsf{G} \mid \mathsf{C}\rangle \Rightarrow \langle \mathsf{G} \mid \mathsf{C}\rangle$

Fig. 1. Transformation of elementary constraints

6 Examples

6.1 Checking constraints.

The reduction relation which only rewrites the subterms of a term $s = f(s_1, \ldots, s_k)$ which are reachable by following the *replacing* arguments $i \in \mu(f) \subseteq \{1, \ldots, ar(f)\}$ indicated by a *replacement map* $\mu : \mathcal{F} \to \mathcal{P}(\mathbb{N})$ is called *context-sensitive rewriting* (*CSR* [17]).

Proving termination of *CSR* is an interesting problem with several applications [20]. Consider the TRS $\mathcal{R}$ [20, Example 14]:

$$h(X) \to g(X,X) \quad (8)$$
$$g(a,X) \to f(b,X) \quad (9)$$
$$f(X,X) \to h(a) \quad (10)$$
$$a \to b \quad (11)$$

together with the following *replacement map*: $\mu(f) = \mu(g) = \mu(h) = \{1\}$. Proofs of termination of *CSR* can be obtained by using the results in [1,2] which generalize to *CSR* the well-known dependency pairs method [3]. In this setting, we have to consider the following rules $\mathsf{DP}(\mathcal{R},\mu)$ (which are the *context-sensitive dependency pairs* [1,2]):

$$H(X) \to G(X,X) \quad (12)$$
$$H(a,X) \to F(b,X) \quad (13)$$
$$F(X,X) \to H(a) \quad (14)$$

where F, G, and H are fresh symbols and we assume $\mu(F) = \mu(G) = \mu(H) = \{1\}$. Now we use *$\mu$-reduction pairs* which are pairs $(\gtrsim, \sqsupset)$ such that $\gtrsim$ is μ-monotonic (i.e., $\gtrsim$ is i-monotonic for all $i \in \mu(f)$ and all symbols f). We have to solve the following constraints [1,2]: $l \gtrsim r$ for all rules $l \to r$ in $\mathcal{R}$, $u \gtrsim v$ or $u \sqsupset v$ for all rules $u \to v$ in $\mathsf{DP}(\mathcal{R},\mu)$, and $u \sqsupset v$ for at least one rule $u \to v$ in $\mathsf{DP}(\mathcal{R},\mu)$. The following linear elementary interpretation

$$[a] = 2 \qquad [b] = 0 \qquad [f](x,y) = (2y+4)^{4x-y+1}$$
$$[g](x,y) = x + (\tfrac{1}{2}x + y + 1)^{x-2y} \qquad [h](x) = x+1 \qquad [F](x,y) = (2y+4)^{4x-y+1}$$
$$[G](x,y) = x + (\tfrac{1}{2}x + y + 1)^{x-2y} \qquad [H](x) = x+1$$

is compatible with the rules of the system in the following sense:

$$[h(X)] \geq [g(X,X)] \qquad [g(a,X)] \geq [f(b,X)]$$
$$[f(X,X)] \geq [h(a)] \qquad [a] \geq [b]$$
$$[H(X)] \geq [G(X,X)] \qquad [G(a,X)] >_1 [F(b,X)]$$
$$[F(X,X)] >_1 [H(a)]$$

As shown in Example 4.5, the linear elementary interpretations for f, g, F and G are weakly 1-monotonic, as required (for h and H it is obvious). These facts can be used to prove that $\mathcal{R}$ is μ-terminating, i.e., that no infinite context-sensitive rewrite sequence is possible[5]. In contrast, the use of polynomial interpretations does not lead to a proof of termination for this example.

Let us illustrate the use of the transformation rules in Figure 1 to *check* these inequalities. With $[h(X)] = X + 1$ and $[g(X,X)] = X + (\frac{3}{2}X + 1)^{-X}$,

[5] This conclusion is not immediate but it easily follows by using the results in [2].

we have

$$
\begin{aligned}
&\langle X+1 \geq X+(\tfrac{3}{2}X+1)^{-X} \mid True\rangle\\
&\quad \Rightarrow_{\mathsf{Add\ decomp}} \langle X \geq X \wedge 1 \geq (\tfrac{3}{2}X+1)^{-X} \mid True\rangle\\
&\quad \Rightarrow_{\mathsf{Reflexivity}} \langle 1 \geq (\tfrac{3}{2}X+1)^{-X} \mid True\rangle\\
&\quad \Rightarrow_{\mathsf{Negative\ Exp}} \langle \tfrac{3}{2}X+1 > 0 \wedge (\tfrac{3}{2}X+1)^{X} \geq 1 \mid True\rangle\\
&\quad \Rightarrow_{\mathsf{K-Introd}} \langle \tfrac{3}{2}X+1 > 0+0 \wedge (\tfrac{3}{2}X+1)^{X} \geq 1 \mid 0+0=0\rangle\\
&\quad \Rightarrow_{\mathsf{Add\ decomp}} \langle \tfrac{3}{2}X \geq 0 \wedge 1 > 0 \wedge (\tfrac{3}{2}X+1)^{X} \geq 1 \mid 0+0=0\rangle\\
&\quad \Rightarrow_{\mathsf{K-Introd}} \langle \tfrac{3}{2}X \geq 0*0 \wedge 1 > 0 \wedge (\tfrac{3}{2}X+1)^{X} \geq 1 \mid 0+0=0\rangle\\
&\quad \Rightarrow_{\mathsf{Prod\ decomp}} \langle \tfrac{3}{2} \geq 0 \wedge X \geq 0 \wedge 1 > 0 \wedge (\tfrac{3}{2}X+1)^{X} \geq 1 \mid 0+0=0 \wedge 0*0=0\rangle\\
&\quad \Rightarrow_{\mathsf{Constants}} \langle X \geq 0 \wedge 1 > 0 \wedge (\tfrac{3}{2}X+1)^{X} \geq 1 \mid 0+0=0 \wedge 0*0=0 \wedge \tfrac{3}{2} \geq 0\rangle\\
&\quad \Rightarrow_{\mathsf{Variables}} \langle 1 > 0 \wedge (\tfrac{3}{2}X+1)^{X} \geq 1 \mid 0+0=0 \wedge 0*0=0 \wedge \tfrac{3}{2} \geq 0\rangle\\
&\quad \Rightarrow_{\mathsf{Constants}} \langle (\tfrac{3}{2}X+1)^{X} \geq 1 \mid 0+0=0 \wedge 0*0=0 \wedge \tfrac{3}{2} \geq 0 \wedge 1 > 0\rangle\\
&\quad \Rightarrow_{\mathsf{K-Introd}} \langle (\tfrac{3}{2}X+1)^{X} \geq 1^0 \mid 0+0=0 \wedge 0*0=0 \wedge \tfrac{3}{2} \geq 0 \wedge 1 > 0 \wedge 1^1 = 1\rangle\\
&\quad \Rightarrow_{\mathsf{Exp\ decomp}} \langle \tfrac{3}{2}X+1 \geq 1 \wedge X \geq 0 \mid 0+0=0 \wedge 0*0=0 \wedge \tfrac{3}{2} \geq 0 \wedge 1 > 0 \wedge 1^1 = 1\rangle\\
&\quad \Rightarrow_{\mathsf{Add\ decomp}} \langle \tfrac{3}{2}X \geq 0 \wedge 1 \geq 1 \wedge X \geq 0 \mid 0+0=0 \wedge 0*0=0 \wedge \tfrac{3}{2} \geq 0 \wedge 1 > 0 \wedge 1^1 = 1\rangle\\
&\quad \Rightarrow^{*} \langle \Box \mid 0+0=0 \wedge 0*0=0 \wedge \tfrac{3}{2} \geq 0 \wedge 1 > 0 \wedge 1^1 = 1 \wedge 1 \geq 1\rangle
\end{aligned}
$$

where $0+0=0 \wedge 0*0=0 \wedge \frac{3}{2} \geq 0 \wedge 1 > 0 \wedge 1^1 = 1 \wedge 1 \geq 1$ is trivially satisfied.

Remark 6.1 No proof for $X+1 \geq X+(\frac{3}{2}X+1)^{-X}$ is possible by using the methods in [16]: it is not valid in Lescanne's formalization due to the rational number $\frac{3}{2}$ and the negative exponent; and no rule in [16] plays the role of Negative Exp.

6.2 Solving constraints.

Our second example illustrates the *generation* of coefficients. Consider the TRS $\mathcal{R}$ in Example 1.1. According to the previous considerations, we would give the following parametric interpretations to the function symbols (for simplicity, only $fact$ is given a linear elementary interpretation):

$$
\begin{aligned}
[0] &= z_0 & [s](x) &= s_1x + s_0\\
[mul](x,y) &= m_{11}xy + m_{10}x + m_{01}y + m_{00} & [add](x,y) &= a_1x + a_2y + a_0\\
[fact](x) &= f_1x + f_0 + (f'_1x + f'_0)^{f''_1x+f''_0}
\end{aligned}
$$

(i) *Algebraicity*: As discussed in Section 4.1, we just need to ensure that all coefficients for $[0]$, $[add]$, $[mul]$ and $[s]$ are nonnegative, and that $f_0, f_1, f'_0, f'_1 \geq 0$.

(ii) *Monotonicity*: For symbols add, mul and s, which are interpreted as polynomials, we require $a_1, a_2, m_{10}, m_{01}, s_1 \geq 1$. Regarding $fact$, by Proposition 4.6, we require $f'_0 \geq 1$, $f_1, f''_0, f''_1 \geq 0$, $f'_1 f''_0 \geq f'_0$ and $f'_1 f''_1 \geq f'_1$.

(iii) *Constraints corresponding to the rules*: We only consider the rules yielding non-polynomial constraints.

(a) Rule $l_5 \to r_5$, i.e., $fact(0) \to s(0)$. We have:

$$[l_5] = f_1 z_0 + f_0 + (f'_1 z_0 + f'_0)^{f''_1 z_0 + f''_0}$$
$$[r_5] = s_1 z_0 + s_0$$

The application of the transformation rules in Figure 1 yields

$$\langle f_1 z_0 + f_0 + (f'_1 z_0 + f'_0)^{f''_1 z_0 + f''_0} > s_1 z_0 + s_0 \mid True \rangle$$
$$\Rightarrow_{\mathsf{Constants}} \langle \Box \mid f_1 z_0 + f_0 + (f'_1 z_0 + f'_0)^{f''_1 z_0 + f''_0} > s_1 z_0 + s_0 \rangle$$

(b) Rule $l_6 \to r_6$, i.e., $fact(s(X)) \to mul(s(X), fact(X))$. We have:

$$\begin{aligned}
[l_6] &= f_1(s_1 X + s_0) + f_0 + (f'_1(s_1 X + s_0) + f'_0)^{f''_1(s_1 X + s_0) + f''_0} \\
&= f_1 s_1 X + f_1 s_0 + f_0 + (f'_1 s_1 X + f'_1 s_0 + f'_0)^{f''_1 s_1 X + f''_1 s_0 + f''_0} \\
&= A_1 X + A_0 + (B_1 X + B_0)^{C_1 X + C_0} \\
[r_6] &= m_{11}(s_1 X + s_0)(f_1 X + f_0 + (f'_1 X + f'_0)^{f''_1 X + f''_0}) + \\
&\quad + m_{10}(s_1 X + s_0) + m_{01}(f_1 X + f_0 + (f'_1 X + f'_0)^{f''_1 X + f''_0}) + m_{00} \\
&= f_1 m_{11} s_1 X^2 + (f_1 m_{11} s_0 + f_0 m_{11} s_1 + m_{10} s_1 + m_{01} f_1) X + \\
&\quad + f_0 m_{11} s_0 + m_{10} s_0 + m_{01} f_0 + m_{00} \\
&\quad + m_{11} s_1 X (f'_1 X + f'_0)^{f''_1 X + f''_0} + (m_{11} s_0 + m_{01})(f'_1 X + f'_0)^{f''_1 X + f''_0} \\
&= A'_2 X^2 + A'_1 X + A'_0 + D'_1 X (B'_1 X + B'_0)^{C'_1 X + C'_0} + E'_1 (B'_1 X + B'_0)^{C'_1 X + C'_0} \\
&= A'_2 X^2 + A'_1 X + A'_0 + (D'_1 X + E'_1)(B'_1 X + B'_0)^{C'_1 X + C'_0}
\end{aligned}$$

The application of the transformation rules in Figure 1 succeeds and yields

$$A'_2 = 0 \wedge B_0 > 2 \wedge C'_0 > 0 \wedge \overline{C}_0 + 2 = C_0 \wedge B_1 \geq A'_1 \wedge B_0 \geq A'_0 \wedge B_1 \geq D'_1$$
$$\wedge B_0 > E'_1 \wedge B_1 \geq B'_1 \wedge B_0 > B'_0 \wedge C_1 \geq C'_1 \wedge \overline{C}_0 \geq C'_0$$

that is, we have to solve the following (polynomial) constraints:

$$A'_2 = f_1 m_{11} s_0 + f_0 m_{11} s_1 + m_{10} s_1 + m_{01} f_1 = 0 \quad (15)$$
$$B_0 = f'_1 s_0 + f'_0 > 2 \quad (16)$$
$$C'_0 = f''_0 > 0 \quad (17)$$
$$C_0 = f''_1 s_0 + f''_0 = \overline{C}_0 + 2 \quad (18)$$
$$A'_1 = f_1 m_{11} s_0 + f_0 m_{11} s_1 + m_{10} s_1 + m_{01} f_1 \leq f'_1 s_1 = B_1 \quad (19)$$
$$A'_0 = f_0 m_{11} s_0 + m_{10} s_0 + m_{01} f_0 + m_{00} \leq f'_1 s_0 + f'_0 = B_0 \quad (20)$$
$$B_1 = f'_1 s_1 \geq m_{11} s_1 = D'_1 \quad (21)$$
$$B_0 = f'_1 s_0 + f'_0 > m_{11} s_0 + m_{01} = E'_1 \quad (22)$$
$$B_1 = f'_1 s_1 \geq f'_1 = B'_1 \quad (23)$$
$$B_0 = f'_1 s_0 + f'_0 > f'_0 = B'_0 \quad (24)$$
$$C_1 = f''_1 s_1 \geq f''_1 = C'_1 \quad (25)$$
$$\overline{C}_0 \geq f''_0 = C'_0 \quad (26)$$

Note that all these constraints hold when we let the parametric coefficients take the values which are used in the linear elementary interpretation of Example 1.1 (together with $\overline{C}_0 = 1$).

7 Implementation issues

The implementation of the techniques described above is conceptually simple, although its efficiency could highly depend on appropriate choices of the data structures and implementation languages. Comparing two polynomial expressions *à la Hong and Jakuš'* is pretty simple: we just perform an independent comparison of (coefficients of) monomials according to its composition in terms of variables and powers. With elementary expressions, things are not so simple. For instance, think of the elementary expressions $[fact(s(X))] = (2X+4)^{2X+3}$ and $[mul(s(X), fact(X))] = (2X+3)(2X+2)^{2X+1} + 2X + 2$ in Example 1.1. At first sight, the second expression is 'more complicated' and the first impression is that it cannot be smaller than the first one. However, as shown in Example 5.2, indeed this is the case. The key point is that we can use the algebraic properties of the exponential to *unfold* the first expression $(2X+4)^{2X+3}$ and show an equivalent (or smaller) expression whose shape is similar to the second one. In practice, when comparing elementary expressions e and e', this amounts to finding some *clusters* $\mathcal{E} = \{E_1, \ldots, E_m\}$ and $\mathcal{E}' = \{E'_1, \ldots, E'_n\}$ of subexpressions in e and e' in such a way that we can define a surjective *mapping* $\gamma : \mathcal{E}' \to \mathcal{E}$ such that the subexpression of e' represented by E'_j is *smaller* or *covered* by the subexpression in e represented by $E_i = \gamma(E'_j)$.

When comparing *parametric* expressions (which is the focus of this paper) the problem is similar but now we do not have (many) specific numbers which can be used to do some algebraic manipulation. For this reason, the rule

K-Introd is so important: it allows us to 'create' new expressions including new variables and constants which are semantically related with the old ones by means of equality constraints. On the other hand, we would have to do clustering as well (but we have even more flexibility due to the possibility of introducing arbitrary constants).

Since it is well-known that exponential expressions are (ultimately) bigger than polinomial ones (disregarding the degree), a possible strategy is decomposing e as $e = p + e_1 + \cdots + e_m$ where p is a purely polynomial expression and the expressions e_i contain some exponential subexpressions. Then, we use such e_i (first) *individually* to establish appropriate *corresponding* clusters of subexpressions (additive components) in e' which satisfy the conditions above.

8 Conclusions and future work

In the nineties, Lescanne introduced and motivated the use of elementary algebraic interpretations as an alternative to polynomial interpretations in proofs of termination of term rewriting. Lescanne considered elementary interpretations over the naturals (actually over the subset of natural numbers starting from 2) and investigated how to *check* the inequalities $[l] > [r]$ which are obtained for a *given* interpretation which should be provided by the user.

In this paper we have investigated elementary interpretations *over the reals*. We have introduced the *linear elementary functions* as a suitable choice for the automation of termination proofs using elementary interpretations. We have shown how the requirements of modern termination methods (e.g., the dependency pairs method) for a given termination problem are translated into *parametric* elementary constraints over the reals. We have defined a rule-based transformation system for checking and solving arbitrary (parametric) elementary constraints over the reals. Using this system, universally quantified semantic variables are removed from the parametric constraints corresponding to a given termination problem while an existential constraint consisting of parametric coefficients (only) is built to subsequently invoke an appropriate solver. The obtained solution witnesses the satisfaction of the original constraint.

The most urgent future work is having an implementation of the proposed system. We have argued that this is not conceptually difficult but it could require more research before obtaining heuristics leading to an efficient and competitive system. On the other hand, the theory developed in this paper provides an appropriate guide for extending our initial proposal of linear elementary interpretations to more refined ones (for instance, one could allow that A, B, and C in (7) are arbitrary polynomials). The special cases enumer-

ated in Remark 4.1 could also be investigated since they have specific features which could enable a simpler implementation of constraint solving.

Acknowledgement

I thank Vicent del Olmo for the proof of Proposition 5.1. I also thank Albert Rubio, Vesna Pavlovic, and the referees for many useful comments.

References

[1] B. Alarcón, R. Gutiérrez, and S. Lucas. Context-Sensitive Dependency Pairs. In S. Arun-Kumar and N. Garg, editors, *Proc. of XXVI Conference on Foundations of Software Technology and Theoretical Computer Science, FST&TCS'06*, LNCS 4337:297–308, 2006. Springer-Verlag.

[2] B. Alarcón, R. Gutiérrez and S. Lucas, *Context-sensitive dependency pairs*, Technical report, Universidad Politécnica de Valencia (2008), available as Technical Report DSIC-II/10/08.

[3] T. Arts and J. Giesl. Termination of Term Rewriting Using Dependency Pairs. *Theoretical Computer Science*, 236:133-178, 2000.

[4] F. Baader and T. Nipkow. Term Rewriting and All That. Cambridge University Press, 1998.

[5] C. Borralleras, S. Lucas, R. Navarro-Marset, E. Rodríguez-Carbonell, and A. Rubio. Solving Non-linear Polynomial Arithmetic via SAT Modulo Linear Arithmetic. In *Proc. of CADE'09*, LNAI 5663:294-305, 2009.

[6] C. Borralleras and A. Rubio. Orderings and Constraints: Theory and Practice of Proving Termination. In H. Comon-Lundh, C. Kirchner, and H. Kirchner, editors, *Rewriting, Computation and Proof, Essays Dedicated to Jean-Pierre Jouannaud on the Occasion of His 60th Birthday*, LNCS 4600, 2007.

[7] A. ben Cherifa and P. Lescanne. Termination of rewriting systems by polynomial interpretations and its implementation. *Science of Computer Programming*, 9(2):137-160, 1987.

[8] E. Contejean, C. Marché, A.-P. Tomás, and X. Urbain. Mechanically proving termination using polynomial interpretations. Journal of Automated Reasoning, 32(4):315-355, 2006.

[9] N. Dershowitz. Termination of rewriting. *Journal of Symbolic Computation*, 3:69-115, 1987.

[10] J.J. Gray. The Hilbert Challenge. Oxford University Press, 2000.

[11] H. Hong and D. Jakuš. Testing Positiveness of Polynomials. *Journal of Automated Reasoning* 21:23-38, 1998.

[12] J. Jaffar and J.-L. Lassez. Constraint Logic Programming. In Proc. of the 14th ACM Symposium on Principles of Programming Languages, POPL'87, pag. 111-119, ACM Press, 1987.

[13] J. Jaffar and M.J. Maher. Constraint Logic Programming: A Survey. *Journal of Logic Programming* 19&20:503-581, 1994.

[14] D.S. Lankford. On proving term rewriting systems are noetherian. Technical Report, Louisiana Technological University, Ruston, LA, 1979.

[15] P. Lescanne. Termination of Rewrite Systems by Elementary Interpretations. In H. Kirchner and G. Levi, editors, Proc. of 3^{rd} International Conference on Algebraic and Logic Programming, ALP'92, LNCS 632:21-36, 1992.

[16] P. Lescanne. Termination of Rewrite Systems by Elementary Interpretations. *Formal Aspects of Computing* 7:77-90, 1995.

[17] S. Lucas. Context-sensitive computations in functional and functional logic programs. *Journal of Functional and Logic Programming*, 1998(1):1-61, January 1998.

[18] S. Lucas. Polynomials over the reals in proofs of termination: from theory to practice. *RAIRO Theoretical Informatics and Applications*, 39(3):547-586, 2005.

[19] S. Lucas. Practical use of polynomials over the reals in proofs of termination. In *Proc. of 9th International Symposium on Principles and Practice of Declarative Programming, PPDP'07*, pages 39-50, ACM Press, 2007.

[20] S. Lucas. Proving Termination of Context-Sensitive Rewriting by Transformation. *Information and Computation*, 204(12):1782–1846, 2006.

[21] K. Marriot and P. Stuckey. Programming with Constraints. An Introducction. The MIT Press, 1998.

[22] F. Rossi, P. van Beek, and T. Walsh. Handbook of Constraint Programming. Elsevier, 2006.