Complexity invariance of real interpretations

Guillaume Bonfante, Florian Deloup

Complexity invariance of real interpretations

Guillaume Bonfante[1], Florian Deloup[2]

1- Université de Nancy - LORIA, Nancy, France,
2- Université Paul Sabatier, Toulouse - IMT, France

Abstract. In the field of implicit computational complexity, we are considering in this paper the fruitful branch of interpretation methods. In this area, the synthesis problem is solved by Tarski's decision procedure, and consequently interpretations are usually chosen over the reals rather than over the integers. Doing so, one cannot use anymore the (good) properties of the natural (well-) ordering of **N** employed to bound the complexity of programs. We show that, actually, polynomials over the reals benefit from some properties that allow their safe use for complexity. We illustrate this by two characterizations, one of PTIME and one of PSPACE.

To prove the termination of a rewrite system, it is natural to interpret terms into a well-founded ordering. For instance, Lankford describes interpretations as monotone Σ-algebras with domain of interpretation being the natural numbers with their usual ordering (c.f. [16, 15]).

However, in the late seventies, Dershowitz showed in a seminal paper [8] that the well-foundedness of the domain of interpretation is not necessary whenever the interpretations are chosen to be monotonic and to have the sub-term property. Thus, the domain of the Σ-algebra mentioned above can be the set of real numbers.

One of the main interesting points about choosing of real numbers rather than natural numbers is that we get (at least from a theoretical point of view) a procedure to verify the validity of an interpretation of a program by Tarski's decomposition procedure [25] and an algorithm to compute interpretations up to some fixed degree. Following Roy et al. [3], the complexity of these algorithms is exponential with respect to the size of the program.

A second good point is that the use of reals (as opposed to integers) enlarges the set of rewriting systems that are compatible with an interpretation, as shown recently by Lucas [17].

In the last years, the study of termination methods has been one of the major tools in implicit computational complexity. For instance, Moser et al. have characterized Ptime by means of pop* in [2], and context dependent interpretations in [22] after their introduction by Hofbauer [12]. One of our two characterizations, Theorem 8, use dependency pairs (c.f. [1]). In this vein, we mention here the work of Hirokawa and Moser [10], and, in the same spirit, Lucas and Peña

[0] Work partially supported by project ANR-08-BLANC-0211-01 (COMPLICE)

in [19] made some investigation on the tools of rewriting to tackle the complexity of a first order functional programs.

But, the main concern of the present paper is to show that the structure of polynomials over the reals has an important role from the point of view of complexity. Our thesis is that, in the field of complexity, due to Stengle's Positivstellensatz [24], polynomials over the reals can safely replace polynomials over the integers. It is illustrated by two theorems, Theorem 6 and Theorem 8. We show that one may recover both derivational complexity (up to a polynomial) and size bounds (also, up to a polynomial) on terms as applications of Positivstellensatz. Moreover, this can be done in a constructive way. Our thesis But, let us draw briefly the roadmap of the key technical features of this work.

Given a strict interpretation for a term rewriting system, it follows immediately that for all rewriting steps $s \to t$, we have $(\!|s|\!) > (\!|t|\!)$. If one takes the interpretation on natural numbers (as they were introduced by Lankford [16]), this can be used to give a bound on the derivation height. Thus, Hofbauer and Lautemann have shown in [11] that the derivation height is bounded by a double exponential. However, their argument uses deeply the fact that the interpretation of a term is itself a bound on the derivation height:

$$\mathtt{dh}(t) \leq (\!|t|\!). \tag{1}$$

Suppose $t_0 \to t_1 \to \cdots \to t_n$, since $(\!|t_0|\!) > (\!|t_1|\!) > \cdots > (\!|t_n|\!)$, on natural numbers, this means that $n \leq (\!|t_0|\!)$. Such a proof does not hold with real numbers. The inequalities $(\!|t_i|\!) > (\!|t_{i+1}|\!)$ are due to a) for all rewrite rule $\ell \to r$, the inequality $(\!|\ell|\!) > (\!|r|\!)$ implies that

$$(\!|\ell|\!) \geq (\!|r|\!) + 1 \tag{2}$$

and b) that for all $x_i > y_i$:

$$(\!|f|\!)(x_1, \ldots, x_i, \ldots, x_n) - (\!|f|\!)(x_1, \ldots, y_i, \ldots, x_n) \geq x_i - y_i. \tag{3}$$

The two inequalities 2, 3 do not hold in general for real interpretations. To recover the good properties holding with natural numbers, people have enforced the inequalities on terms. For instance [17, 21] suppose the existence of some real $\delta > 0$ such that for any rule $\ell \to r : (\!|\ell|\!) \geq (\!|r|\!) + \delta$. We prove that even without the existence of such a δ, the derivation height of a term t is bounded by $(\!|t|\!)$ up to a polynomial.

To save some space, we have omitted some proofs. The reader will find them in the extended version of the paper, see [5].

1 Preliminaries

We suppose that the reader has familiarity with first order rewriting. We briefly recall some of the main notions of the theory, essentially to fix the notations. Dershowitz and Jouannaud's survey [9] is a good entry point.

Let $\mathcal{X}$ denote a (countable) set of *variables*. Given a *signature* Σ, the set of *terms* over Σ and $\mathcal{X}$ id denoted by $\mathcal{T}(\Sigma, \mathcal{X})$ and the set of *ground terms* as $\mathcal{T}(\Sigma)$. The size $|t|$ of a term t is defined as the number of symbols in t.

Given a signature Σ, a rule is an oriented equation $\ell \to r$ with $\ell, r \in \mathcal{T}(\Sigma, \mathcal{X})$ such that variables occurring in r also occur in ℓ. A Term Rewrite System (TRS) is a finite set of such rules. A TRS induces a rewriting relation denoted by $\to$. The relation $\xrightarrow{+}$ is the transitive closure of $\to$ and $\xrightarrow{*}$ is the reflexive and transitive closure of $\to$. Finally, we say that a term t is a *normal form* if there is no term u such that $t \to u$. Given two terms t and u, $t \xrightarrow{!} u$ denotes the fact that $t \xrightarrow{*} u$ and u is a normal form. We write $t_0 \to^n t_n$ the fact that $t_0 \to t_1 \cdots \to t_n$. One defines the derivation height for a term t as the maximal length of a derivation: $\mathtt{dh}(t) = \max\{n \in \mathbf{N} \mid \exists v : t \to^n v\}$.

A *context* is a term C with a particular variable $\Diamond$. If t is a term, $C[t]$ denotes the term C where the variable $\Diamond$ has been replaced by t. A substitution is a mapping from variables to terms. A substitution σ can be extended canonically to terms and we note $t\sigma$ the application of the substitution σ to the term t.

1.1 Syntax of programs

Definition 1. *A program is a* 5*-tuple* $\mathtt{f} = \langle \mathcal{X}, \mathcal{C}, \mathcal{F}, \mathtt{main}, \mathcal{E} \rangle$ *with:*

- $\mathcal{C}$ *is a (finite) signature of* constructor *symbols and* $\mathcal{F}$ *a (finite) signature of* function symbols*.* $\mathtt{main} \in \mathcal{F}$ *is the "main" function symbol*
- $\mathcal{E}$ *is a finite set of rules of the shape* $\mathtt{f}(p_1, \cdots, p_n) \to r$ *where* $\mathtt{f} \in \mathcal{F}$ *and* $p_i \in \mathcal{T}(\mathcal{C}, \mathcal{X})$*.*

Moreover, we suppose programs to be confluent. This is achieved by the following syntactic restriction due to Huet [14] (see also [23]): (i) Each rule $\mathtt{f}(p_1, \ldots, p_n) \to t$ is left-linear, that is a variable appears only once in $\mathtt{f}(p_1, \cdots, p_n)$, and (ii) there are no two left hand-sides which are overlapping.

The program $\langle \mathcal{X}, \mathcal{C}, \mathcal{F}, \mathtt{f}, \mathcal{E} \rangle$ computes the partial function $[\![\mathtt{f}]\!] : \mathcal{T}(\mathcal{C})^n \to \mathcal{T}(\mathcal{C})$ defined as follows. For every $u_1, \cdots, u_n \in \mathcal{T}(\mathcal{C}), [\![\mathtt{f}]\!](u_1, \cdots, u_n) = v$ iff $\mathtt{f}(u_1, \cdots, u_n) \xrightarrow{*} v$ and $v \in \mathcal{T}(\mathcal{C})$. Otherwise, it is undefined.

Example 1. The following program computes the membership in a list. The constructors of lists are **cons**, **nil**. Elements in the list are the tally natural numbers build from **0** and **s**.

$$\begin{array}{lll} \mathtt{not}(\mathbf{tt}) \to \mathbf{ff} & \mathtt{or}(\mathbf{tt}, y) \to \mathbf{tt} & \mathbf{0} = \mathbf{0} \to \mathbf{tt} \\ \mathtt{not}(\mathbf{ff}) \to \mathbf{tt} & \mathtt{or}(x, \mathbf{tt}) \to \mathbf{tt} & \mathbf{0} = \mathbf{s}(y) \to \mathbf{ff} \\ & \mathtt{or}(\mathbf{ff}, \mathbf{ff}) \to \mathbf{ff} & \mathbf{s}(x) = \mathbf{0} \to \mathbf{ff} \\ & & \mathbf{s}(x) = \mathbf{s}(y) \to x = y \end{array}$$

$$\begin{array}{r} \mathtt{in}(x, \mathbf{nil}) \to \mathbf{ff} \\ \mathtt{in}(x, \mathbf{cons}(a, l)) \to \mathtt{or}(x = a, \mathtt{in}(x, l)) \end{array}$$

Definition 2 (Call-tree). *Suppose we are given a program* $\langle \mathcal{X}, \mathcal{C}, \mathcal{F}, \mathcal{E} \rangle$*. Let* $\rightsquigarrow$ *be the relation*

$$(f, t_1, \ldots, t_n) \rightsquigarrow (g, u_1, \ldots, u_m) \Leftrightarrow f(t_1, \ldots, t_n) \to C[g(v_1, \ldots, v_m)] \xrightarrow{*} C[g(u_1, \ldots, u_m)]$$

where f and g are defined symbols, $t_1, \ldots, t_n, u_1, \ldots, u_m$ are ground constructor terms and $v_1, \ldots, v_m$ are arbitrary (ground) terms. Given a term $f(t_1, \ldots, t_n)$, the relation $\rightsquigarrow$ defines a tree whose root is $(f, t_1, \ldots, t_n)$ and η' is a daughter of η iff $\eta \rightsquigarrow \eta'$.

1.2 Interpretations of programs

Given a signature Σ, a Σ-algebra on the domain $\mathbf{R}^+$ is a mapping $(\!|-|\!)$ which associates to every n-ary symbol $f \in \Sigma$ an n-ary function $(\!|f|\!) : \mathbf{R}^{+n} \to \mathbf{R}^+$. Such a Σ-algebra can be extended to terms by:

- $(\!|x|\!) = 1_{\mathbf{R}^+}$, that is the identity on $\mathbf{R}^+$, for $x \in \mathcal{X}$,
- $(\!|f(t_1, \ldots, t_m)|\!) = \mathrm{comp}((\!|f|\!), (\!|t_1|\!), \ldots, (\!|t_m|\!))$ where comp is the composition of functions.

Given a term t with n variables, $(\!|t|\!)$ is a function $\mathbf{R}^{+n} \to \mathbf{R}^+$.

Definition 3. *Given a program $\langle \mathcal{X}, \mathcal{C}, \mathcal{F}, \mathsf{f}, \mathcal{E} \rangle$, let us consider a $(\mathcal{C} \cup \mathcal{F})$-algebra $(\!|-|\!)$ on $\mathbf{R}^+$. It is said to:*

1. *be strictly monotonic if for any symbol f, the function $(\!|f|\!)$ is a strictly monotonic function, that is if $x_i > x'_i$, then*
$$(\!|f|\!)(x_1, \ldots, x_n) > (\!|f|\!)(x_1, \ldots, x'_i, \ldots, x_n),$$
2. *be weakly monotonic if for any symbol f, the function $(\!|f|\!)$ is a weakly monotonic function, that is if $x_i \geq x'_i$, then*
$$(\!|f|\!)(x_1, \ldots, x_n) \geq (\!|f|\!)(x_1, \ldots, x'_i, \ldots, x_n),$$
3. *have the strict sub-term property if for any symbol f, the function $(\!|f|\!)$ verifies $(\!|f|\!)(x_1, \ldots, x_n) > x_i$ with $i \in \{1, \ldots, n\}$,*
4. *to be strictly compatible (with the rewriting relation) if for all rules $\ell \to r$, $(\!|\ell|\!) > (\!|r|\!)$,*
5. *to be a sup-approximation if for all constructor terms $t_1, \ldots, t_n$, we have the inequality $(\!|f(t_1, \ldots, t_n)|\!) \geq (\!|[\![f]\!](t_1, \ldots, t_n)|\!)$.*

Definition 4. *Given a program $\langle \mathcal{X}, \mathcal{C}, \mathcal{F}, \mathsf{f}, \mathcal{E} \rangle$, a $(\mathcal{C} \cup \mathcal{F})$-algebra on $\mathbf{R}^+$ is said to be a strict interpretation whenever it verifies (1), (3), (4). It is a sup-interpretation whenever it verifies (2) and (5).*

Sup-interpretation have been introduced by Marion and Pechoux in [20]. We gave here a slight variant of their definition. In [20], the last inequality refers to the size of normal forms. We preferred to have a more uniform definition. Clearly, a strict interpretation is a sup-interpretation.When we want to speak arbitrarily of one of those concepts, we use the generic word "interpretation". We also use this terminology to speak about the function $(\!|f|\!)$ given a symbol f.

Finally, by default, we restrict the interpretations of symbols to be *Max-Poly functions*, that is functions obtained by finite compositions of the constant functions, maximum, addition and multiplication.

Definition 5. *The interpretation of a symbol f is said to be additive if it has the shape $\sum_i x_i + c$ with $c > 0$. A program with an interpretation is said to be additive when its* constructors *are additive.*

Example 2. The program given in Example 1 has both an additive strict interpretation (left side, black) and an additive sup-interpretation (right side, blue):

$$\begin{aligned} (\!|\mathbf{tt}|\!) = (\!|\mathbf{ff}|\!) = (\!|\mathbf{0}|\!) = (\!|\mathbf{nil}|\!) &= 1 \\ (\!|\mathbf{s}|\!)(x) &= x+1 \\ (\!|\mathbf{cons}|\!)(x,y) &= x+y+3 \\ (\!|\mathtt{not}|\!)(x) &= x+1 \\ (\!|\mathtt{or}|\!)(x,y) = (\!|=|\!)(x,y) &= x+y+1 \\ (\!|\mathtt{in}|\!)(x,y) &= (x+1)(y+1) \end{aligned} \qquad \begin{aligned} (\!|\mathbf{tt}|\!) = (\!|\mathbf{ff}|\!) = (\!|\mathbf{0}|\!) = (\!|\mathbf{nil}|\!) &= 1 \\ (\!|\mathbf{s}|\!)(x) &= x+1 \\ (\!|\mathbf{cons}|\!)(x,y) &= x+y+1 \\ (\!|\mathtt{not}|\!)(x) &= 1 \\ (\!|\mathtt{or}|\!)(x,y) = (\!|=|\!)(x,y) &= 1 \\ (\!|\mathtt{in}|\!)(x,y) &= 1 \end{aligned}$$

Example 3. The Quantified Boolean Formula (QBF) problem is well known to be Pspace complete. It consists in determining the validity of a boolean formula with quantifiers over propositional variables. Without loss of generality, we restrict formulae to $\neg, \vee, \exists$. Variables are represented by tally numbers. The QBF problem is solved extending the preceding program with:

$$\begin{aligned} \mathtt{verify}(\mathbf{Var}(x), t) &\to \mathtt{in}(x,t) \\ \mathtt{verify}(\mathbf{Not}(\varphi), t) &\to \mathtt{not}(\mathtt{verify}(\varphi, t)) \\ \mathtt{verify}(\mathbf{Or}(\varphi_1, \varphi_2), t) &\to \mathtt{or}(\mathtt{verify}(\varphi_1, t), \mathtt{verify}(\varphi_2, t)) \\ \mathtt{verify}(\mathbf{Exists}(n, \varphi), t) &\to \mathtt{or}(\mathtt{verify}(\varphi, \mathbf{cons}(n, t)), \mathtt{verify}(\varphi, t)) \\ \mathtt{qbf}(\varphi) &\to \mathtt{verify}(\varphi, \varepsilon) \end{aligned}$$

It has a sup-interpretation but not a strict interpretation:

$$\begin{aligned} (\!|\mathbf{Not}|\!)(x) = (\!|\mathbf{Var}|\!)(x) &= x+1 \\ (\!|\mathbf{Or}|\!)(x,y) = (\!|\mathbf{Exists}|\!)(x,y) &= x+y+1 \\ (\!|\mathtt{verify}|\!)(x,y) = (\!|\mathtt{qbf}|\!)(x) &= 1 \end{aligned}$$

Actually, as Theorem 6 will prove it, unless Ptime = Pspace, there is no program computing QBF with an additive strict interpretation.

2 Positivstellensatz and applications

In this section, we introduce a deep mathematical result, the Positivstellensatz. Then we give some applications to polynomial interpretations. They will be key points of the Theorems 6 and 8 in our analysis of the role of reals in complexity (§3).

Let $n > 0$. Denote by $\mathbf{R}[x_1, \ldots, x_n]$ the $\mathbf{R}$-algebra of polynomials with real coefficients. Denote by $(\mathbf{R}^+)^n = \{x = (x_1, \ldots, x_n) \in \mathbf{R}^n \mid x_1, \ldots, x_n > 0\}$ the first quadrant. Since we need to consider only the $\mathbf{R}$-algebra of polynomial functions $(\mathbf{R}^+)^n \to \mathbf{R}$, it will be convenient to identify the two spaces. In particular throughout this section, all polynomial functions are defined on $(\mathbf{R}^+)^n$.

Theorem 3 (Positivstellensatz, Stengle [24]). *Suppose that we are given polynomials $P_1, \ldots, P_m \in \mathbf{R}[x_1, \ldots, x_k]$, the following two assertions are equivalent:*

1. $\{x_1, \ldots, x_k : P_1(x_1, \ldots, x_k) \geq 0 \wedge \cdots \wedge P_m(x_1, \ldots, x_k) \geq 0\} = \emptyset$
2. $\exists Q_1, \ldots, Q_m : -1 = \sum_{i \leq m} Q_i P_i$ *where each Q_i is a sum of squares of polynomials (and so is positive and monotonic).*

Moreover, these polynomials $Q_1, \ldots, Q_m$ can effectively computed. We refer the reader to the work of Lombardi, Coste and Roy [6]. As a consequence, all the constructions given below can be actually (at least theoretically) computed.

It will be convenient to derive from the Positivstellensatz some propositions useful for our applications.

Proposition 2. *Suppose that a TRS (Σ, R) admits an interpretation $(\!|-|\!)$ over* Max-Poly *such that for all rules $\ell \to r$, we have $(\!|\ell|\!) > (\!|r|\!)$. There is a positive, monotonic polynomial P such that for any rule $\ell \to r$, we have $(\!|\ell|\!)(x_1, \ldots, x_k) - (\!|r|\!)(x_1, \ldots, x_k) \geq \dfrac{1}{P(x_1, \ldots, x_k)}$.*

The proof is direct consequence of Theorem 3 when ℓ and r are polynomials. By a finite case analysis, one may cope with the max function. One may notice that one cannot a priori find some constant $\delta > 0$ such that: $(\!|\ell|\!)(x_1, \ldots, x_k) \geq (\!|r|\!)(x_1, \ldots, x_k) + \delta$. Indeed, suppose that $(\!|\ell|\!)(x_1, \ldots, x_k) - (\!|r|\!)(x_1, \ldots, x_k) > 0$. Observe that $\lim_{x \to 0} P(x, 1/x) = \lim_{x \to 0} x^2 = 0$. However, taking $Q(x, y) = (1 + x + y)^2$, one has $P(x, y)Q(x, y) \geq 1$ for all $x, y \geq 0$.

Proposition 2 has an important consequence. Since, in a derivation all terms have an interpretation bounded by the interpretation of the first term, there is a minimal decay for each rule of the derivation. Then, due to the next Theorem, the result can be extended to contexts.

Theorem 4. *Given a polynomial $P \in \mathbf{R}[x_1, \ldots, x_n]$ such that*

(i) $\forall x_1 \geq 0, \ldots, x_n \geq 0 : P(x_1, \ldots, x_n) > \max(x_1, \ldots, x_n)$,

(ii) $\forall x_1 \geq 0, \ldots, x_n \geq 0 : \dfrac{\partial P}{\partial x_i}(x_1, \ldots, x_n) > 0$ *for all $i \leq n$,*

then, there exist $A > 0$ such that for any $\Delta > 0$, we have $P(x_1, \ldots, x_i + \Delta, \ldots, x_n) > P(x_1, \ldots, x_n) + \Delta$ whenever $||x|| > A$.

In other words, we recovered some equivalent Equations to the Equations 2, 3 for sufficiently large terms.

Proposition 3. *Suppose that a TRS (Σ, R) admits a strict interpretation $(\!|-|\!)$ over* Max-Poly. *For all $A > 0$, the set of terms $\{t \in \mathcal{T}(\Sigma) \mid (\!|t|\!) < A\}$ is finite.*

Proposition 4. *Suppose that a TRS (Σ, R) admits a strict interpretation $(\!|-|\!)$ over Poly. There are a real $A > 0$ and a positive, monotonic polynomial P such that for all $x_1, \ldots, x_n \geq 0$, if $x_{i_1}, \ldots, x_{i_k} > A$, then for all symbols f, we have $(\!|f|\!)(x_1, \ldots, x_n) \geq x_{i_1} + \cdots + x_{i_k} + \dfrac{1}{P((\!|f|\!)(x_1, \ldots, x_n))}$.*

This latter result gives (more or less) directly a bound on the size of terms. It is a consequence of Theorem 3 and the following Theorem.

Theorem 5. *Given a polynomial* $P \in \mathbf{R}[x_1, \ldots, x_n]$ *such that*

(i) $\forall x_1, \ldots, x_n \geq 0 : P(x_1, \ldots, x_n) > \max(x_1, \ldots, x_n)$,
(ii) $\forall x_i' > x_i, x_1, \ldots, x_n \geq 0 : P(x_1, \ldots, x_i', x_{i+1}, \ldots, x_n) > P(x_1, \ldots, x_n)$,

then, there exist $A \geq 0$ *such that* $P(x_1, \ldots, x_n) > x_1 + \cdots + x_n$ *whenever* $||x|| > A$.

All the hypotheses are necessary. If $P(x, y)$ is not supposed to be greater than $\max(x, y)$, you can simply take $P(x, y) = (x + y)/2$. It is strictly monotonic, but clearly, $P(x, y) < x + y$ for all $x, y > 0$.

The second hypothesis is also necessary. A counter example is given by $P(x, y) = 16(x - y)^2 + (3/2)x + 1$.

3 The role of reals in complexity

We have now all the tools to prove that reals can safely replace integers from a complexity point of view. This is illustrated by Theorem 6 and Theorem 8.

Theorem 6. *Functions computed by programs with an additive strict interpretation (over the reals) are exactly* Ptime *functions.*

The rest of the section is devoted to the proof of the Theorem. The main difficulty of the proof is that inequalities as given by the preceding section only hold for sufficiently large values. So, the main issue is to split "small" terms (and "small rewriting steps") from "large" ones. Positivstellensatz gives us the arguments for the large terms (Lemma 7), Lemmas 8,9 show that there are not too many small steps between two large steps. Lemma 11 describe how small steps and big steps alternate.

From now on, we suppose we are given a program with an additive strict interpretation over polynomials. The following Lemmas are direct applications of Proposition 2,4, they are the main steps to prove both Theorem 6 and Theorem 8. A full proof of the lemmas can be found in the technical report.

Lemma 7. *There is a polynomial* P *and a real* $A > 0$ *such that for all steps* $\ell\sigma \to r\sigma$ *with* $(\!|r\sigma|\!) > A$, *then, for all contexts* C, *we have* $(\!|C[\ell\sigma]|\!) \geq (\!|C[r\sigma]|\!) + \frac{1}{P((\!|\ell\sigma|\!))}$.

Proof. This is a consequence of Proposition 2.

Definition 7. *Given a real* $A > 0$, *we say that the* A*-size of a closed term* t *is the number of subterms* u *of* t *(including itself) such that* $(\!|u|\!) > A$. *We note* $|t|_A$ *the* A*-size of* t.

Lemma 8. *There is a constant* A *such that for all* $C > A$, *there is a polynomial* Q *for which* $|t|_C \leq Q((\!|t|\!))$ *for all closed terms* t.

Proof. This is consequence of Proposition 4.

For $A > 0$, we say that $t = C[\ell\sigma] \to C[r\sigma] = u$ is an A-step whenever $(\!|r\sigma|\!) > A$. We note such a rewriting step $t \to_{>A} u$. Otherwise, it is an $\leq A$-step, and we note it $t \to_{\leq A} u$. We use the usual $*$ notation for transitive closure. In case we restrict the relation to the call by value strategy[1], we add "cbv" as a subscript. Take care that an $\to_{\leq A}$-normal form is not necessarily a normal form for $\to$.

Lemma 9. *There is a constant A such that for all $C > A$ there is a (monotonic) polynomial P such that for all terms t, any call by value derivation $t \to^*_{\leq C,cbv} u$ has length less than $P((\!|t|\!))$.*

Proof. This is a consequence of Proposition 2.

Lemma 10. *For constructor terms, we have $(\!|t|\!) \leq \Gamma \times |t|$ for some constant Γ.*

Proof. Take $\Gamma = \max\{\frac{1}{\gamma_c} \mid (\!|c|\!)(x_1, \ldots, x_n) = \sum_{i=1}^{n} x_i + \gamma_c\}$. By induction on terms.

Lemma 11. *Let us suppose we are given a program with an additive strict interpretation. There is a strategy such that for all function symbol f, for all constructor terms $t_1, \ldots, t_n$, any derivation following the strategy starting from $f(t_1, \ldots, t_n)$ has length bounded by $Q(\max(|t_1|, \ldots, |t_n|))$ where Q is a polynomial.*

Proof. Let us consider A as defined in Lemma 9, B and P_1 as defined in Lemma 7. We define $C = \max(A, B)$. Let P_0 be the polynomial thus induced from Lemma 9. Finally, let us consider the strategy as introduced above: rewrite as long as possible the according to $\to_{\leq C,\mathrm{cbv}}$, and then, apply an C-step. That is, we have $t_1 \to^*_{\leq C,\mathrm{cbv}} t'_1 \to_{>C,\mathrm{cbv}} t_2 \to^*_{\leq C,\mathrm{cbv}} t'_2 \to^*$. In Lemma 9, we have seen that there are at most $P_0((\!|t_i|\!))$ steps in the derivation $t_i \to^*_{\leq C,\mathrm{cbv}} t'_i$. From Lemma 7, we can state that there are at most $(\!|t_1|\!) \times P_1((\!|t_1|\!))$ such C-steps. Consequently, the derivation length is bounded by $(\!|t_1|\!) \times P_1((\!|t_1|\!)) \times P_0((\!|t_1|\!))$ since for all $i \geq 1$, $(\!|t_i|\!) \leq (\!|t_1|\!)$.

Consider now a function symbol $f \in \mathcal{F}$, from Lemma 10, $(\!|f(t_1, \ldots, t_n)|\!) = (\!|f|\!)((\!|t_1|\!), \ldots, (\!|t_n|\!)) \leq (\!|f|\!)(\Gamma \max(|t_1|, \ldots, |t_n|), \ldots, \Gamma \max(|t_1|, \ldots, |t_n|))$. Then, the conclusion is immediate.

Proof. Of Theorem 6 With the strategy defined above, we have seen that the derivation length of a term $f(t_1, \ldots, t_n)$ is polynomial wrt to $\max(|t_1|, \ldots, |t_n|)$. The computation can be done in polynomial time due to dal Lago and Martini, see [7], together with the fact that the normal form has polynomial size (Lemma 10). For the converse part, we refer the reader to [4] where a proof that PTIME programs can be computed by functional programs with strict interpretations over the integers. This proof can be safely used in the present context.

[1] Innermost in the present context.

3.1 Dependency Pairs with polynomial interpretation over the reals

Termination by Dependency Pairs is a general method introduced by Arts and Giesl [1]. It puts into light recursive calls. Suppose $f(t_1, \ldots, t_n) \to C[g(u_1, \ldots, u_m)]$ is a rule of the program. Then, $(F(t_1, \ldots, t_n), G(u_1, \ldots, u_m))$ is a dependency pair where F and G are new symbols associated to f and g respectively. $S(\mathcal{C}, \mathcal{F}, R)$ denotes the program thus obtained by adding these rules. The dependency graph links dependency pairs $(u, v) \to (u', v')$ if there is a substitution σ such that $\sigma(v) \xrightarrow{*} \sigma(u)$ and termination is obtained when there is no cycles in the graph. Since the definition of the graph involves the rewriting relation, its computation is undecidable. In practice, one gives an approximation of the graph which is bigger. Since this is not the issue here, we suppose that we have a procedure to compute this supergraph which we call the dependency graph.

Theorem 7. *[Arts,Giesl [1]] A TRS $(\mathcal{C}, \mathcal{F}, R)$ is terminating iff there exists a well-founded weakly monotonic quasi-ordering $\geq$, where both $\geq$ and $>$ are closed under substitution, such that*

- *$\ell \geq r$ for all rules $\ell \to r$,*
- *$s \geq t$ for all dependency pairs (s, t) on a cycle of the dependency graph and*
- *$s > t$ for at least one dependency pair on each cycle of the graph.*

It is natural to use sup interpretations for the quasi-ordering and the ordering of terms. However, the ordering $>$ is not well-founded on $\mathbf{R}$, so that system may not terminate. Here is such an example.

Example 4. Consider the non terminating system:

$$\begin{pmatrix} \mathtt{f}(\mathbf{0}) \to \mathbf{0} \\ \mathtt{f}(x) \to \mathtt{f}(\mathbf{s}(x)) \end{pmatrix}$$

Take $(\!|0|\!) = 1$, $(\!|\mathbf{s}|\!)(x) = x/2$. There is a unique dependency pair $F(x) \to F(\mathbf{s}(x))$. We define $(\!|F|\!)(x) = (\!|f|\!)(x) = x + 1$.

One way to avoid these infinite descent is to force the inequalities over reals to be of the form $P(x_1, \ldots, x_n) \geq Q(x_1, \ldots, x_n) + \delta$ for some $\delta > 0$ (see for instance Lucas's work [18]). Doing so, one gets a well-founded ordering on reals. We propose an alternative approach to that problem, keeping the original ordering of $\mathbf{R}$.

Definition 8. *A $\mathbf{R}$-DP-interpretation for a program associates to each symbol f a monotonic function $(\!|f|\!)$such that*

1. *constructors have additive interpretations,*
2. *$(\!|\ell|\!) \geq (\!|r|\!)$ for $\ell \to r \in R$,*
3. *$(\!|s|\!) \geq (\!|r|\!)$ for $(s, r) \in DP(R)$,*
4. *for each dependency pair (s, t) in a cycle, $(\!|s|\!) > (\!|r|\!)$ holds.*

Example 5. Let us come back to Example 3. The QBF problem can be given a **R**-DP interpretation. Let us add the interpretations:

$$\begin{aligned} (\!|\mathrm{NOT}|\!)(x) &= x \\ (\!|\mathrm{OR}|\!)(x,y) = (\!|\mathrm{EQ}|\!)(x,y) &= \max(x,y) \\ (\!|\mathrm{IN}|\!)(x,y) &= x+y \\ (\!|\mathrm{VERIFY}|\!)(x,y) &= 2x+y+1 \\ (\!|\mathrm{QBF}|\!)(x) &= 2x+1 \end{aligned}$$

Theorem 8. *Functions computed by programs*

- *with additive* **R***-DP-interpretations*
- *the interpretation of any capital symbol F has the sub-term property*

are exactly PSPACE *computable functions.*

Proof. The completeness comes from the example of the QBF, plus the compositionality of such interpretation.

In the other direction, the key argument is to prove that the call tree has a polynomial depth w.r.t. the size of arguments. The proof relies again on Lemmas 7, 8, 9, 11 adapted to dependency pairs (in cycles). Indeed, since capital symbol have the sub-term property, the lemmas are actually valid in the present context.[2] The rewriting steps of dependency pairs can be reinterpreted as depth-first traversal in the call-tree. Thus, we can state that the depth of the call tree is polynomial, as we stated in an analogous way that the derivation length was polynomial.

Conclusion

If one goes back to the two characterization of complexity classes presented in this paper, one sees that we essentially use two arguments: a) interpretations with the subset properties provide a polynomial bound wrt the interpretation of the initial interpretation, and b) the size of terms is polynomial w.r.t. their interpretations.

As a consequence, our result can be used in other context such as proofs of termination by matrix interpretations [13] or context dependent interpretations [12]. Potentially, any system dealing with decreasing chain of (interpreted) terms could be treated.

Thanks to: Antoine Henrot for his helpful comments on an earlier version of the contribution.

[2] Lemma 8 is immediate here since we focus on terms of the shape $F(t_1, \ldots, t_n)$ where $t_1, \ldots, t_n$ are constructor terms.

References

1. Thomas Arts and Jürgen Giesl. Termination of term rewriting using dependency pairs. *Theoretical Computer Science*, 236(1-2):133–178, 2000.
2. Martin Avanzini and Georg Moser. Complexity analysis by rewriting. In Jacques Garrigue and Manuel V. Hermenegildo, editors, *FLOPS*, volume 4989 of *Lecture Notes in Computer Science*, pages 130–146. Springer, 2008.
3. Saugata Basu, Richard Pollack, and Marie.-Françoise Roy. *Algorithms in real algebraic geometry*. Springer, Berlin Heidelberg New York, 2003.
4. Guillaume Bonfante, Adam Cichon, Jean-Yves Marion, and Hélène Touzet. Algorithms with polynomial interpretation termination proof. *Journal of Functional Programming*, 11(1):33–53, 2001.
5. Guillaume Bonfante and Florian Deloup. Complexity invariance of real interpretations. Technical report, LORIA, 2010. `http://www.loria.fr/~bonfante/ciri.pdf`.
6. Michel Coste, Henri Lombardi, and Marie-Françoise Roy. Dynamical method in algebra: Effective nullstellensätze. *Annals of Pure and Applied Logic*, 111:203–256, 2001.
7. Ugo dal Lago and Simone Martini. Derivational complexity is an invariant cost model. In *Foundational and Practical Aspects of Resource Analysis (FOPARA '09)*, 2009.
8. Nachum Dershowitz. A note on simplification orderings. *Information Processing Letters*, pages 212–215, 1979.
9. Nachum Dershowitz and Jean-Pierre Jouannaud. *Handbook of Theoretical Computer Science vol.B*, chapter Rewrite systems, pages 243–320. 1990.
10. Nao Hirokawa and Georg Moser. Automated complexity analysis based on the dependency pair method. In Alessandro Armando, Peter Baumgartner, and Gilles Dowek, editors, *IJCAR*, volume 5195 of *Lecture Notes in Computer Science*, pages 364–379. Springer, 2008.
11. D. Hofbauer and C. Lautemann. Termination proofs and the length of derivations. In *Proceedings of the 3rd international conference on Rewriting Techniques and Applications, RTA 89*, pages 167–177, New York, NY, USA, 1989. Springer-Verlag.
12. Dieter Hofbauer. Termination proofs by context-dependent interpretations. In Aart Middeldorp, editor, *Proceedings of the 12th International Conference on Rewriting Techniques and Applications, RTA-01*, volume 2051 of *Lecture Notes in Computer Science*, pages 108–121, Utrecht, The Netherlands, 2001. Springer-Verlag.
13. Dieter Hofbauer. Proving termination with matrix interpretations. In *Proceedings of the 17th International Conference on Rewriting Techniques and Applications, RTA-06*, volume 4098 of *Lecture Notes in Computer Science*, pages 328–342, Seattle, USA, 2006. Springer-Verlag.
14. Gérard Huet. Confluent reductions: Abstract properties and applications to term rewriting systems. *Journal of the ACM*, 27(4):797–821, 1980.
15. Gérard Huet and Derek Oppen. Equations and rewrite rules: A survey. In R. V. Book, editor, *Formal Language Theory: Perspectives and Open Problems*, pages 349–405. AP, 1980.
16. D.S. Lankford. On proving term rewriting systems are noetherien. Technical report, 1979.
17. Salvador Lucas. On the relative power of polynomials with real, rational, and integer coefficients in proofs of termination of rewriting. *Appl. Algebra Eng., Commun. Comput.*, 17(1):49–73, 2006.

18. Salvador Lucas. Practical use of polynomials over the reals in proofs of termination. In *PPDP '07: Proceedings of the 9th ACM SIGPLAN international conference on Principles and practice of declarative programming*, pages 39–50, New York, NY, USA, 2007. ACM.
19. Salvador Lucas and Ricardo Peña. Rewriting techniques for analysing termination and complexity bounds of safe programs. In *LOPSTR 08*, pages 43–57, 2008.
20. Jean-Yves Marion and Romain Péchoux. Resource analysis by sup-interpretation. In *FLOPS*, volume 3945 of *Lecture Notes in Computer Science*, pages 163–176. Springer-Verlag, 2006.
21. Jean-Yves Marion and Romain Péchoux. Characterizations of polynomial complexity classes with a better intensionality. In Sergio Antoy and Elvira Albert, editors, *PPDP*, pages 79–88. ACM, 2008.
22. Georg Moser and Andreas Schnabl. Proving quadratic derivational complexities using context dependent interpretations. In *Rewriting Techniques and Applications*, volume 5117 of *Lecture Notes in Computer Science*, pages 276–290. Springer-Verlag, 2008.
23. Barry Rosen. Tree-manipulating systems and church-rosser theorems. *Journal of the ACM*, 20(1):160–187, 1973.
24. Gilbert Stengle. A nullstellensatz and a positivstellensatz in semialgebraic geometry. *Mathematische Annalen*, 207(2):87–97, 1973.
25. Alfred Tarski. *A Decision Method for Elementary Algebra and Geometry*. University of California Press, 1951. 2nd edition.