Synthesis of sup-interpretations: a survey

Romain Péchoux

Synthesis of sup-interpretations: a survey

Romain Péchoux

Université de Lorraine and INRIA team Carte,
LORIA, Campus Scientifique - BP 239 - 54506 Vandoeuvre-lès-Nancy Cedex

Abstract

In this paper, we survey the complexity of distinct methods that allow the programmer to synthesize a sup-interpretation, a function providing an upper-bound on the size of the output values computed by a program. It consists in a static space analysis tool without consideration of the time consumption. Although clearly related, sup-interpretation is independent from termination since it only provides an upper bound on the terminating computations. First, we study some undecidable properties of sup-interpretations from a theoretical point of view. Next, we fix term rewriting systems as our computational model and we show that a sup-interpretation can be obtained through the use of a well-known termination technique, the polynomial interpretations. The drawback is that such a method only applies to total functions (strongly normalizing programs). To overcome this problem we also study sup-interpretations through the notion of quasi-interpretation. Quasi-interpretations also suffer from a drawback that lies in the subterm property. This property drastically restricts the shape of the considered functions. Again we overcome this problem by introducing a new notion of interpretations mainly based on the dependency pairs method. We study the decidability and complexity of the sup-interpretation synthesis problem for all these three tools over sets of polynomials. Finally, we take benefit of some previous works on termination and runtime complexity to infer sup-interpretations.

Keywords: Complexity Analysis, Static Analysis, Resource Upper Bounds, Interpretation, Quasi-interpretation, Sup-interpretation

1. Introduction

1.1. Motivations

The notion of sup-interpretation was introduced in [1] in order to study program extensional complexity. This tool is devoted to statically analyze the complexity of programs guaranteeing that a secured system resists to buffer-overflows and thus allowing the programmer to verify complexity properties of

Email address: Romain.Pechoux@loria.fr (Romain Péchoux)

programs used in safety-critical systems. Sup-interpretations focus on analyzing the complexity of programs or, more specifically, term rewrite systems by considering upper bounds on the size of values computed by a program, by static analysis.
Basically, a sup-interpretation of a program is a function that provides an upper-bound on the size of the computed output with respect to the input size. In other words, given a program $\mathbf{p}$, the sup-interpretation of $\mathbf{p}$ is a function that, given some input data x such that $\mathbf{p}$ converges on input x, provides an upper-bound on the output size in the size of x.
One of the main issues concerning static analysis tools is related to their decidability and/or complexity. In other words, one tries to find if the static analysis is decidable and, if so, one tries to study its complexity. As highlighted by Rice's theorem, most of interesting (or non-trivial) analyses are undecidable and, in most of the cases, this issue is transformed into finding the complexity of a smaller instance of the initial problem. In the particular case of sup-interpretations, the analysis consists in finding the sup-interpretation of a given program, that is in synthesizing a function providing upper bounds on the program computations. We call this analysis the *sup-interpretation synthesis problem*. This paper will be dedicated to survey the results concerning the sup-interpretation (SI) synthesis problem.

1.2. Contribution.

The reader is assumed to be familiar with basic knowledge about term rewrite systems, see chapter 2 of [2] or [3], and computability and complexity, see [4, 5].
We start to show that the general problems of the sup-interpretation synthesis are undecidable when we consider functions and Gödel numberings. Moreover we show that the *sup-interpretation verification problem*, which consists in checking that a function given as input is a sup-interpretation, is Π_1^0-complete in the arithmetical hierarchy and that the sup-interpretation synthesis problem is in Σ_3^0.
Next we specify our language by introducing Term Rewriting Systems (TRS) and we define the corresponding notion of sup-interpretation. Starting from here, we will study well-known termination and complexity tools like polynomial interpretations (PI) and quasi-interpretations (QI) and show that they allow the programmer to obtain a sup-interpretation under some slight restrictions.
We demonstrate that (polynomial) interpretations for termination are special kind of sup-interpretations. However they were designed to study strong normalization and, consequently, they do not provide enough power to study programs computing partial recursive functions.
To overcome this problem, we study the notion of quasi-interpretation. We also show that quasi-interpretations define sup-interpretations.
Finally, we study a new notion called DP-interpretation (DPI) based on the

dependency pairs framework by Arts and Giesl [6] that also defines a sup-interpretation. We show that this new notion strictly generalizes the notion of quasi-interpretation since it does not require any subterm property, a property stating that considered interpretations have to be greater than each of their arguments. In other words, every program admitting a quasi-interpretation admits a DP-interpretation but the converse does not hold.
We study the sup-interpretation synthesis problem with respect to each of these tools on particular sets of polynomials ranging over a structure $\mathbb{K} \in \{\mathbb{N}, \mathbb{Q}^+, \mathbb{R}^+\}$. The considered sets of polynomials are:

- the set $\mathbb{K}[\overline{X}]$ of usual multivariate polynomials whose coefficients are in $\mathbb{K}$ and with n variables $\overline{X} = X_1, \cdots, X_n$ ranging over the field of real numbers,
- the set of $\mathsf{MaxPoly}^{(\mathsf{k},\mathsf{d})}\{\mathbb{K}\}$ polynomials, which consist in functions obtained using constants over $\mathbb{K}$ and arbitrary compositions of the operators $+,\times$ and max of degree bounded by d and max arity bounded by k,
- and the set of $\mathsf{MaxPlus}^{(\mathsf{k},\mathsf{d})}\{\mathbb{K}\}$ functions, which consist in functions obtained using constants over $\mathbb{K}$ bounded by d and arbitrary compositions of the operators $+$ and max, with a max arity bounded by k.

The obtained results can be summarized by the following Figure:

Function space \ tool	PI	QI	DPI
$\mathbb{K}[\overline{X}], \mathbb{K} \in \{\mathbb{N}, \mathbb{Q}^+\}$	Undecidable	Undecidable	Undecidable
$\mathbb{R}^+[\overline{X}]$	Exptime	Exptime	Exptime
$\mathsf{MaxPoly}\{\mathbb{K}\}, \mathbb{K} \in \{\mathbb{N}, \mathbb{Q}^+\}$	✠	Undecidable	Undecidable
$\mathsf{MaxPoly}^{(\mathsf{k},\mathsf{d})}\{\mathbb{R}^+\}$	✠	Exptime	Exptime
$\mathsf{MaxPlus}^{(\mathsf{k},\mathsf{d})}\{\mathbb{K}\}, \mathbb{K} \in \{\mathbb{N}, \mathbb{Q}^+_{\mathsf{d}}\}$	✠	NP-complete	NP-complete
$\mathsf{MaxPlus}^{(\mathsf{k},\mathsf{d})}\{\mathbb{R}^+\}$	✠	NP-hard	NP-hard

Figure 1: Decidability and complexity of the sup-interpretation synthesis problem

where $\mathbb{Q}^+_{\mathsf{d}}$ consists in rationals of bounded representation.
The first line is direct consequences of Hilbert's tenth problem undecidability whereas the second line is a consequence of Tarski's quantifier elimination Theorem over real numbers. One important point to mention here is that the

synthesis problem is exponential and not doubly exponential because the synthesis problem is more restricted than general quantifier elimination.
In the first column, the symbol ✠ means that it does not make sense to study the synthesis problem with respect to the considered set of functions. Indeed the synthesis of polynomial interpretation has no meaning for any structure including a max operator since max is not a strictly monotonic function whereas polynomial interpretations deal with functions enjoying such a property.
The results for MaxPoly function space are identical to the results on pure polynomials since the max operator can be eliminated for both QI and DPI.
Finally, in the last two lines of Figure 1.2, we show that the synthesis problem is NP-hard for MaxPlus, independently of the structure. As a corollary, on bounded search spaces like $\mathbb{N}$ or $\mathbb{Q}_{\mathsf{d}}^{+}$, the problem is NP-complete. The meaning of such a notion is unclear over an unbounded and uncountable space like $\mathbb{R}^{+}$. Note that these results are a Corrigendum to results already presented in an unpublished workshop [7] that were wrongly stating a NP-completeness result over $\mathbb{R}^{+}$.

Finally, we take benefit of termination results on the runtime complexity of TRS to infer sup-interpretations in a last section. In analogy with complexity theory, we show that time bounded computations imply size (or space) bounded computations. However the space bound may be exponential in the time, if the derivation length is the considered measure of time. Indeed, a derivation of length n may correspond to exponential space by just using variable duplication. We discuss the complexity of the synthesis problem for all of these termination techniques.

1.3. Outline

In Section 2 we consider general undecidable problems of the sup-interpretation synthesis when considering functions. In Section 3, we introduce Term Rewriting Systems and the corresponding notion of sup-interpretation that slightly differs from the sup-interpretation on functions. In Section 4, we introduce polynomial interpretations as sup-interpretations and study the decidability and complexity of their sup-interpretation synthesis problem. Sections 5 and 6 apply the same analysis to the notions of quasi-interpretation and DP-interpretation. Section 7 discusses the relation between time and space, where time is considered to be the derivation length and space is considered to be the size of a term. This section shows how to synthesize a sup-interpretation through the use of termination techniques. Finally, Section 8 discusses the main open issues.

1.4. Related works

Sup-interpretations are inspired by two former notions on Term Rewriting Systems, the polynomial interpretations, introduced in [8, 9] to analyze program termination and runtime complexity [10, 11, 12], and the quasi-interpretations, introduced in [13] and used to characterize complexity classes such as FPtime, FPspace or Logspace (See [14, 15, 16]).

The general framework of sup-interpretation was introduced in [1] without considering the synthesis problem. [17] was the first paper to combine interpretation methods together with the dependency pairs method in order to characterize polynomial time and space complexity classes in a more intensional way, that is by capturing more natural algorithms corresponding to a given polynomial time or space function. However the results were presented independently of the notion of sup-interpretation and the present paper gives a deeper understanding on the combination of both methods in order to obtain a sup-interpretation.
One important point to stress here is that sup-interpretations are an extensional tool contrarily to quasi-interpretations and polynomial interpretations that are intensional tools. It means that sup-interpretations deal with functions as mathematical object in the sense of complexity theory, that is functions computed by some programs, whereas (general) interpretations are intensional tools and deal with program properties. As a consequence, they also allow the programmer to study finer and more technical program behaviors. For example, a quasi-interpretation also provides upper-bounds on the size of a program intermediate computations whereas this property has no meaning for a sup-interpretation. However sup-interpretations can be combined in criteria in order to get intensional properties such as upper bounds on the size of intermediate values. The aim of this paper is neither to cover the way to get such intensional properties nor to show how they can help in characterizing complexity classes. Consequently, we encourage the interested reader to study [1].
The paper [18] has already deeply studied the synthesis problem for quasi-interpretations using max-polynomials with additive coefficient in $\mathbb{N}$ or $\{0, 1\}$ and variables in $\mathbb{Q}^+$. The present work takes advantage of these results to present them from a sup-interpretation point of view. Moreover they are extended, firstly, by considering rational and real multiplicative coefficients and, secondly, by extending the NP-hardness proof of [18] over $\mathbb{N}$ to NP-completeness results over natural numbers and rational numbers of bounded representation (and not only $\{0, 1\}$). One last and important point is that the aim of the current paper is not to provide an automated way to synthesize a sup-interpretation but to find the complexity of the synthesis problem depending on the tool used (interpretation, quasi-interpretation, DP-interpretation, termination tools...), on the set of considered functions (polynomials, polynomials with max,...) and on the considered domain (positive real or rational numbers, natural numbers,...). The reader interested by automation should refer to the recent papers [19, 20, 21] that allow to build interpretations (and consequently, sup-interpretations as demonstrated in Section 4) for showing program termination and to the tools that synthesize quasi-interpretations [22, 23].

2. Undecidability results

In this section, we show undecidability results for the synthesis of sup-interpretations. All these results are machine independent and rely on simple Cantor's diagonalizations using Gödel numbering and s_n^m theorem:

Definition 1. *Suppose that we have a fixed procedure that lists all the sequences of instructions. It associates the set of instructions P_x, the $(x+1)st$ set of instructions in the list, to each integer x. x is called the Gödel number of P_x and it corresponds to the partial recursive function φ_x determined by P_x.*

Theorem 1 (Kleene s_n^m [24])**.** *$\forall m, n \geq 1$ there is a recursive function s_n^m of arity $m+1$ such that $\forall x, y_1, \ldots, y_m$:*

$$\lambda z_1 \ldots \lambda z_n . \varphi_x(y_1, \ldots, y_m, z_1, \ldots, z_n) = \varphi_{s_n^m(x, y_1, \ldots, y_m)}$$

In what follows, let PRF be the set of partial recursive functions and RF be the set of total recursive functions of domain and codomain $\mathbb{N}$. Given a function $f \in PRF$ and some number x, we write $f(x)\downarrow$ (respectively $\downarrow_t$) if f yields an output on input x (resp. in time t), and we write $f(x)\uparrow$ otherwise (resp. $\uparrow_t$ otherwise). Consequently $f(x)\downarrow$ is equivalent to $\exists t,\ f(x)\downarrow_t$. μ is the classical minimization operator. Given a property $P(x)$, $\mu x.P(x)$ is the smallest x satisfying P.

Definition 2. *Given a function $f \in PRF$, a sup-interpretation of f is a function $F \in RF$ that bounds f on its definition domain, i.e. $\forall x \in \mathbb{N},\ f(x)\downarrow \implies F(x) \geq f(x)$.*

First we can show as a direct consequence of Rice's Theorem that there exist partial recursive functions that do not have any recursive sup-interpretation:

Theorem 2.

$$\neg(\forall f \in PRF,\ \exists F \in RF, \forall x \in \mathbb{N}, f(x)\downarrow \implies F(x) \geq f(x))$$

Proof. Suppose that the implication $\forall f \in PRF,\ \exists F \in RF,\ \forall x \in \mathbb{N}, f(x)\downarrow \implies F(x) \geq f(x)$ holds. Define the function f by $f(n) = \varphi_n(n) + 1$. By definition, f is clearly in PRF. Consequently, $\exists F \in RF$ such that $\forall x \in \mathbb{N}, f(x)\downarrow \implies F(x) \geq f(x)$. Let i be the Gödel number of such a function F. We obtain that $\forall x \in \mathbb{N}, f(x)\downarrow \implies \forall x,\ \varphi_i(x) \geq f(x)$. As a consequence, $\varphi_i(i) \geq f(i) = \varphi_i(i) + 1$. It contradicts the hypothesis that $F \in RF$. □

This diagonalization result no longer holds if we allow F to be in PRF (In this case, we can trivially set $F(x) = f(x)$).
Now we try to find a recursive function that given two Gödel numbers x and y would allow us to compare the corresponding partial recursive functions φ_x and φ_y. We also obtain a negative answer to this issue.

Theorem 3. *$\nexists F \in RF$,*

$$F(x,y) = \begin{cases} 1 & \text{if } \forall z,\ \varphi_x(z)\downarrow \implies \varphi_x(z) \leq \varphi_y(z) \\ 0 & \text{otherwise} \end{cases}$$

Proof. Suppose that such a recursive function F exists and define f to be the characteristic function of $\{< x, y > | \forall z,\ \varphi_x(z)\downarrow \implies \varphi_x(z) \leq \varphi_y(z)\}$. f

is recursive. Define ϕ to be a function of two variables corresponding to the following instructions set: given the input $< x, y >$, apply P_x to x and return 0 if and when this computation converges. By Church-Turing thesis, it defines a partial recursive function:

$$\phi(x,y) = \begin{cases} 0 & \text{if } \varphi_x(x)\downarrow \\ \uparrow & \text{otherwise} \end{cases}$$

Suppose that i is the Gödel number of such a function, applying s_n^m Theorem, we obtain that there is a recursive function s_1^1 such that $\forall x,\ \lambda y.\phi(x,y) = \varphi_{s_1^1(i,x)}$. Now suppose that x_0 is a Gödel number for the constant function $\lambda x.0$. We have that $\lambda x.f(s_1^1(i,x),x_0)$ is recursive since it is obtained by composition of recursive functions. However by definition:

$$\begin{aligned} f(s_1^1(i,x),x_0) &= \begin{cases} 1 & \text{if } \forall z,\ \varphi_{s_1^1(i,x)}(z)\downarrow \Longrightarrow \varphi_{s_1^1(i,x)}(z) \leq 0 \\ 0 & \text{otherwise} \end{cases} \\ &= \begin{cases} 1 & \text{if } \varphi_{s_1^1(i,x)}(z) = 0 \\ 0 & \text{otherwise} \end{cases} \\ &= \begin{cases} 1 & \text{if } \varphi_x(x)\downarrow \\ 0 & \text{otherwise} \end{cases} \end{aligned}$$

So we have reduced our function to a variant of the halting problem (see Rogers [4]) which is known to be undecidable. Consequently, $\lambda x.f(s_1^1(i,x),x_0)$ is not recursive and we obtain a contradiction. □

Consequently, we obtain that the *sup-interpretation verification problem* defined by $SI(F) = \{x \mid \forall z\ \varphi_x(z)\downarrow \Longrightarrow\ \varphi_x(z) \leq F(z))\}$, which consists in checking that a given function F is a sup-interpretation of a function f of index x (i.e. $x \in SI(F)$), is undecidable. As a corollary, we also obtain that the *sup-interpretation synthesis problem*, which consists in finding the smallest function wrt Gödel numbering that bounds another given as input, is also undecidable:

Corollary 1. *$\nexists G \in RF$ such that:*

$$G(x) = \begin{cases} \mu y.\{\forall z,\ \varphi_x(z)\downarrow \Longrightarrow \varphi_x(z) \leq \varphi_y(z)\} \\ 0 \quad \textit{otherwise} \end{cases}$$

Proof. Assume that G is recursive and that we have a Gödel numbering starting from Gödel number 1 (i.e. not defined in 0). The reason for which we take such a numbering is just that we do not want to make a confusion between the output 0 when there is no upper-bound and the index 0 of the function φ_0 that might be an upper bound of some other function. Then $\mu y.F(x,y)$, with F defined in Theorem 3 has the same characteristic function than G. Consequently, we obtain a contradiction and G cannot be recursive. □

Now let just state that the sup-interpretation verification problem which consists in checking that a fixed function F is a sup-interpretation of a function φ_x of index x, noted $SI(F)$ is Π_1^0-complete in the arithmetical hierarchy:

Theorem 4. *The sup-interpretation verification problem $SI(F)$ is Π_1^0-complete.*

Proof. For every input z and every t, either $\varphi_x(z)$ terminates within time t and, in this case, we have to compare φ_x and F or $\varphi_x(z)$ does not terminate in time t. Consequently, we can write $SI(F) = \{x \mid \forall z,\ \forall t,\ \varphi_x(z) \uparrow_t \vee(\varphi_x(z) \downarrow_t \wedge \varphi_x(z) \leq F(z))\}$.
We briefly recall that a problem B is complete for some class C of the arithmetical hierarchy if there is a total computable function f such that:

$$x \in A \text{ iff } f(x) \in B$$

for some problem A known to be C-complete. Consider the problem $A = \{x \mid \varphi_x(0) \uparrow\}$. This problem is known to be $\Pi_1^0 - complete$ since it is co-RE.
Now define the function f such that for each x the function $\varphi_{f(x)}$ of index $f(x)$ is defined by:

$$\varphi_{f(x)} = \begin{cases} F(z) + 1 & \text{if } \varphi_x(0) \downarrow \\ \uparrow & \text{otherwise} \end{cases}$$

We clearly have:

$$x \in A \text{ iff } f(x) \in SI(F)$$

Moreover the function f is clearly total, by definition, and computable, by applying s_n^m Theorem. Consequently, $SI(F)$ is $\Pi_1^0 - complete$. □

Now we show that sup-interpretation synthesis problem, SI defined to be "the set of functions $f \in PRF$ for which there is a total recursive function F, satisfying: for all $z \in \mathbb{N}$ if $f(z) \downarrow$ then $F(z) \geq f(z)$" is Σ_3^0 in the arithmetical hierarchy:

Theorem 5. $SI \in \Sigma_3^0$.

Proof. SI can be written equivalently as:

$$SI = \{x \in \mathbb{N} \mid \exists s, \forall z, \exists t, \varphi_s(z) \downarrow_t \wedge(\varphi_x(z) \downarrow_t \implies \varphi_s(z) \geq \varphi_x(z))\}$$

In other words, SI is the the set of indexes x corresponding to functions φ_x for which there exists a total function φ_s providing an upper bound on terminating computations (Indeed $\varphi_x(z) \downarrow_t \implies \varphi_s(z) \geq \varphi_x(z)$). The formula $\varphi_s(z) \downarrow_t \wedge(\varphi_x(z) \downarrow_t \implies \varphi_s(z) \geq \varphi_x(z)) \in \Pi_0^0$ and, consequently, $SI \in \Sigma_3^0$. □

3. Sup-interpretations over Term Rewriting Systems

3.1. TRS as a computational model

The previous section only deals with machine independent results and we have hidden for a while the data representation problems arising. Consequently, we have to adapt slightly the notion of sup-interpretation to each computational model under consideration. Throughout the following Sections, we will consider term rewriting systems.

A Term Rewriting System (TRS for short) is a formal system for manipulating terms over a signature by means of rules.
Terms are strings of symbols consisting of a countably infinite set of variables *Var* and a first-order signature Σ, a non-empty set of function symbols or operator symbols of fixed arity. *Var* and Σ are supposed to be disjoint. As usual, the notation $Ter(\Sigma, Var)$ will be used to denote the set of terms $s, t, \ldots$ of signature Σ and having variables in *Var*.
A (one-hole) context $C[\diamond]$ is a term in $Ter(\Sigma \cup \{\diamond\}, Var)$ with exactly one occurrence of the hole $\diamond$, a symbol of arity 0. Given a term t and context $C[\diamond]$, let $C[t]$ denote the result of replacing the hole $\diamond$ with the term t.
A substitution σ is a mapping from *Var* to $Ter(\Sigma, Var)$.
A rewrite rule for a signature Σ is a pair $l \to r$ of terms $l, r \in Ter(\Sigma, Var)$. A Term Rewrite System is as a pair $\langle \Sigma, \mathcal{R} \rangle$ of a signature Σ and a set of rewrite rules $\mathcal{R}$. In what follows, we will suppose that all the variables of a right-hand side r are included in the variables of l as in Chapter 2 of [2].
A constructor Term Rewrite System is a TRS in which the signature Σ can be partitioned into the disjoint union of a set of function symbols $\mathcal{D}$ and a set of constructors $\mathcal{C}$, such that for every rewrite rule $l \to r$ we have $l = \mathtt{f}(t_1, \cdots, t_n)$ with $\mathtt{f} \in \mathcal{D}$ and $t_1, \cdots, t_n \in Ter(\mathcal{C}, Var)$. The constructors are introduced to represent inductive data. They basically consist of a strict subset $\mathcal{C} \subset \Sigma$ of non-defined functions (a function is defined if it is the root of a left-hand side term in a rule). In what follows, we will only consider constructor TRS and we will use the notation $\langle \mathcal{D} \uplus \mathcal{C}, \mathcal{R} \rangle$ to denote such a particular TRS, $\mathcal{C} \uplus \mathcal{D}$ being the disjoint union of the sets $\mathcal{C}$ and $\mathcal{D}$. Terms in $Ter(\mathcal{C}, Var)$ will be called patterns.
In what follows, we will consider orthogonal constructor TRS since we only want to deal with functions. The notion of orthogonality requires that reduction rules of the system are all left-linear, that is each variable occurs only once on the left hand side of each rule, and there is no overlap between patterns. It is a sufficient condition to ensure that the considered TRS is confluent. It implies that we are clearly talking of functions that maps a term to another (and not functions mapping a term to a set of terms in the case of non-confluent systems). Note that this syntactic requirement could have been withdrawn in favor of a semantic restriction that would only consider TRS that compute functions. Our choice restricts the expressivity of considered TRS but makes sense in our theoretical development since it does not restrict the computed functions set.
Given two terms s and t, we have that $s \to_{\mathcal{R}} t$ if there are a substitution σ, a context $C[\diamond]$ and a rule $l \to r \in \mathcal{R}$ such that $s = C[l\sigma]$ and $t = C[r\sigma]$. Throughout the paper, let $\to_{\mathcal{R}}^*$ (resp. $\to_{\mathcal{R}}^+$) be the reflexive and transitive (resp. transitive) closure of $\to_{\mathcal{R}}$. Moreover we write $s \to_{\mathcal{R}}^n t$ if n rewrite steps are performed to rewrite s to t. A TRS terminates if there is no infinite reduction through $\to_{\mathcal{R}}$.
A function symbol $\mathtt{f}$ of arity n will define a partial function $[\![\mathtt{f}]\!]$ from constructor

terms[1] (sometimes called values) $Ter(\mathcal{C})^n$ to $Ter(\mathcal{C})$ by:

$$\forall v_1, \cdots, v_n \in Ter(\mathcal{C}),\ [\![\mathtt{f}]\!](v_1, \cdots, v_n) = v \text{ iff } \mathtt{f}(v_1, \cdots, v_n) \to_{\mathcal{R}}^* v \ \wedge\ v \in Ter(\mathcal{C})$$

In this case, we write $[\![\mathtt{f}]\!](v_1, \cdots, v_n)\downarrow$ to mean that the computation ends in a normal form (constructor term). If there is no such a v (because of divergence or because evaluation cannot reach a constructor term), then $[\![\mathtt{f}]\!](v_1, \cdots, v_n)\uparrow$. Finally, we define the notion of size of a term $|e|$ which is equal to the number of symbols in e.

3.2. Sup-interpretation of a TRS

Since the goal of sup-interpretation is to provide a non-negative upper bound on the size of computed values, we will mainly restrict our analysis to the groups $\mathbb{N}, \mathbb{Q}^+$ and $\mathbb{R}^+$, where $\mathbb{Q}^+$ and $\mathbb{R}^+$ denote positive rational numbers and positive real numbers. In what follows, let $\mathbb{K} \in \{\mathbb{N}, \mathbb{Q}^+, \mathbb{R}^+\}$ and let $\geq$ and $>$ be the natural ordering and strict ordering on such a structure. Finally, let $>_\delta$ be the strict ordering defined by $x >_\delta y$ iff $x \geq \delta + y$, for some fixed $\delta \in \mathbb{K}$ such that $\delta > 0$.

Definition 3. *Given a TRS $\langle \mathcal{D} \uplus \mathcal{C}, \mathcal{R}\rangle$, an assignment θ over $\mathbb{K}$ is a mapping that maps every symbol $\mathtt{g} \in \mathcal{D} \uplus \mathcal{C}$ of arity m to a total function $\theta(\mathtt{g}) : \mathbb{K}^m \to \mathbb{K}$ and that maps every variable $\in Var$ to a variable in $\mathbb{K}$.*
An assignment is additive *if $\forall c \in \mathcal{C}$ of arity $n > 0$, $\theta(c) = \lambda x_1, \cdots, x_n.(x_1 + \ldots + x_n + k_c)$, for some $k_c \geq 1$, and $\forall c \in \mathcal{C}$ of arity 0, $\theta(c) = 0$. An assignment is* k-additive *if for all $c \in \mathcal{C}$, $k_c \leq k$.*

Definition 4. *An assignment θ over $\mathbb{K}$ is (strictly) monotonic if for every symbol $\mathtt{f}$ of arity m, $\theta(\mathtt{f})$ is a (strictly) monotonic function in each of its arguments. In other words, $\forall i \in [1, m],\ x \geq y \implies \theta(\mathtt{f})(\ldots, x_{i-1}, x, x_{i+1}, \ldots) \geq \theta(\mathtt{f})(\ldots, x_{i-1}, y, x_{i+1}, \ldots)$ (resp. $\forall i \in [1, m], \forall \delta > 0,\ \exists \epsilon > 0,\ \theta(\mathtt{f})(\ldots, x + \delta, \ldots) >_\epsilon \theta(\mathtt{f})(\ldots, x, \ldots)$).*

Now we are able to adapt the notion of sup-interpretation to this model:

Definition 5 (Sup-interpretation)**.** *Given a TRS $\langle \mathcal{D} \uplus \mathcal{C}, \mathcal{R}\rangle$, a monotonic and additive assignment θ over $\mathbb{K}$ is a sup-interpretation over $\mathbb{K}$ if $\forall \mathtt{f} \in \mathcal{D}$ of arity m and $\forall v_1, \cdots, v_m \in Ter(\mathcal{C})$:*

$$\mathtt{f}(v_1, \cdots, v_m)\downarrow \implies \theta(\mathtt{f}(v_1, \cdots, v_m)) \geq \theta([\![\mathtt{f}]\!](v_1, \cdots, v_m))$$

where the sup-interpretation θ is extended canonically to general terms by:

$$\theta(\mathtt{g}(\mathtt{e}_1, \ldots, \mathtt{e}_n)) = \theta(\mathtt{g})(\theta(\mathtt{e}_1), \ldots, \theta(\mathtt{e}_n)), \quad \mathtt{g} \in \mathcal{D} \uplus \mathcal{C}$$

[1] As usual $Ter(\mathcal{C}) = Ter(\mathcal{C}, \emptyset)$

We restrict the shape of constructor symbol sup-interpretations by requiring a k-additive assignment. This restriction is made to relate easily the interpretation of a constructor term and its size, i.e. $\exists k \in \mathbb{N},\ \forall v \in Ter(\mathcal{C}), k \times |v| \geq \theta(v) \geq |v|$ always hold for a TRS wrt a fixed additive sup-interpretation.
We compare this new definition wrt the one presented in previous Section: in a given TRS, the sup-interpretation of a function symbol $\mathtt{f}$ of arity m can be discretized to be viewed as a function $\theta(\mathtt{f}) : \mathbb{N}^m \to \mathbb{N}$ that bounds the size of the output wrt to the input sizes (this is direct for 1-additive sup-interpretations):

Lemma 1. *Given a TRS $\langle \mathcal{D} \uplus \mathcal{C}, \mathcal{R} \rangle$ having a sup-interpretation θ then for each function symbol $\mathtt{f} \in \mathcal{D}$ and for all values $v_1, \cdots, v_m \in Ter(\mathcal{C})$ such that $\mathtt{f}(v_1, \cdots, v_m)\downarrow$, we have:*

$$\theta(\mathtt{f})(k \times |v_1|, \ldots, k \times |v_m|) \geq |[\![\mathtt{f}]\!](v_1, \cdots, v_m)|$$

Proof.

$\theta(\mathtt{f})(k \times \|v_1\|, \ldots, k \times \|v_m\|)$	By monotonicity
$\geq \theta(\mathtt{f})(\theta(v_1), \cdots, \theta(v_m))$	and k-additivity
$= \theta(\mathtt{f}(v_1, \cdots, v_m))$	By extension
$\geq \theta([\![\mathtt{f}]\!](v_1, \cdots, v_m))$	By Definition 5
$\geq \|[\![\mathtt{f}]\!](v_1, \cdots, v_m)\|$	By k-additivity

and so the conclusion. □

4. Polynomial interpretations

4.1. Interpretations as sup-interpretations

Given a TRS, the main issue is now to synthesize a sup-interpretation, that is to compute an upper-bound on the partial function it computes. The first natural technique to do so comes from the term rewriting termination community, is called (polynomial) interpretation and was introduced in [9, 8].

Definition 6 (Interpretation)**.** *Given a TRS $\langle \mathcal{D} \uplus \mathcal{C}, \mathcal{R} \rangle$, an (additive) interpretation is a* strictly *monotonic (additive) assignment $[-]$ over $\mathbb{K}$ which satisfies:*

1. $\forall l \to r \in \mathcal{R},\ [l] > [r]$
2. *If $\mathbb{K} \in \{\mathbb{Q}^+, \mathbb{R}^+\}$ then:*
   (a) *either $\forall \mathtt{g} \in \mathcal{D} \uplus \mathcal{C}$, of arity $m > 0$,*
   $\forall i \in [1, m],\ [\mathtt{g}](X_1, \cdots, X_m) > X_i$
   (b) *or $\forall l \to r \in \mathcal{R},\ [l] >_\delta [r]$*

where the interpretation $[-]$ is extended canonically to terms as usual.

Condition 1 constitutes the basis of interpretation method as introduced in [9, 8]. Condition 2(a) was introduced by Dershowitz [25] to compensate for the loss of well-foundedness over the reals. Finally, condition 2(b) is due to Lucas [26] and captures more TRS than 2(a).
As demonstrated in [8], an interpretation defines a reduction ordering (i.e. a strict, stable, monotonic and well-founded ordering)

Theorem 6. *If a TRS $\langle \mathcal{D} \uplus \mathcal{C}, \mathcal{R} \rangle$ admits an interpretation then it terminates.*

Moreover, an additive interpretation defines a sup-interpretation:

Theorem 7. *Given a TRS $\langle \mathcal{D} \uplus \mathcal{C}, \mathcal{R} \rangle$ having an additive interpretation $[-]$ then $[-]$ is a sup-interpretation.*

Proof. First note that the assignment is additive by assumption.
Second, we show that for each values $v_1, \cdots, v_n \in Ter(\mathcal{C})$ and function symbol $\mathtt{f} \in \mathcal{D}$ such that $\mathtt{f}(v_1, \cdots, v_n) \downarrow$ we have $[\mathtt{f}(v_1, \cdots, v_n)] \geq [[\![\mathtt{f}]\!](v_1, \cdots, v_n)]$. Consider a function symbol $\mathtt{f}$ and values $v_1, \cdots, v_n$, by Theorem 6, we have $\mathtt{f}(v_1, \cdots, v_n) \downarrow$. Since interpretations define a reduction ordering, we have that each reduction corresponds to a (strictly) decreasing sequence:

$$\mathtt{f}(v_1, \cdots, v_n) \to_{\mathcal{R}} u_1 \to_{\mathcal{R}} \ldots \to_{\mathcal{R}} u_k \to_{\mathcal{R}} [[\![\mathtt{f}]\!](v_1, \cdots, v_n)$$
$$[\mathtt{f}(v_1, \cdots, v_n)] > [u_1] > \ldots > [u_k] > [[\![\mathtt{f}]\!](v_1, \cdots, v_n)]$$

and, a fortiori, $[\mathtt{f}(v_1, \cdots, v_n)] \geq [[\![\mathtt{f}]\!](v_1, \cdots, v_n)]$. □

Consequently, finding the interpretation of a given program provides a sup-interpretation of this program under additivity constraints as illustrated by the following example:

Example 1. *Consider the following simple TRS:*

$$\mathtt{d}(0) \to 0 \qquad \mathtt{exp}(0) \to 1$$
$$\mathtt{d}(x+1) \to \mathtt{d}(x) + 2 \qquad \mathtt{exp}(x+1) \to \mathtt{d}(\mathtt{exp}(x))$$

where $x + 2$ and 1 are notations for $(x + 1) + 1$ and $0 + 1$. It admits the following additive interpretation $[0] = 0$, $[+1](X) = X + 1$, $[\mathtt{d}](X) = 3 \times X + 1$, $[\exp](X) = 3^{2 \times X + 1}$. Indeed, it is a strictly monotonic additive assignment and for the last rule, we have:

$$[\mathtt{exp}](x+1) = 3^{2 \times [(x+1)]+1} = 3^{2(X+1)+1} = 3^{2X+3}$$
$$> 3 \times 3^{2X+1} + 1 = [\mathtt{d}](3^{2X+1}) = [\mathtt{d}]([\mathtt{exp}(x)])$$

We let the reader check that the strict inequalities hold for the other rules.

4.2. Restriction to polynomials

It is natural to restrict the space of considered functions (the sup-interpretation codomain) to polynomials for two reasons. First, as we have seen in the first Section, considering the whole space of functions is too general in terms of decidability. Second, polynomials are admitted to be a relevant set of functions in term of time and space complexity. Consequently, we restrict the function space in order to get effective procedures.
In what follows, let $\mathbb{K}[X_1, \ldots, X_m]$ be the set of m-ary polynomials whose coefficients are in $\mathbb{K}$.

Definition 7 (Polynomial interpretation)**.** *Given a TRS $\langle \mathcal{D} \uplus \mathcal{C}, \mathcal{R} \rangle$, a polynomial interpretation over $\mathbb{K}$ is an interpretation $[-]$ over $\mathbb{K}$ that maps every symbol $\mathtt{g} \in \mathcal{D} \uplus \mathcal{C}$ of arity m to a function $[\mathtt{g}] \in \mathbb{K}[X_1, \ldots, X_m]$.*

The synthesis problem for polynomial interpretation has been deeply studied in [27, 28] where algorithms solving the constraints are described. More recently, encoding-based algorithms via SAT or SMT solving have become the state of the art for the synthesis problem [19, 21]. One important question is what is the best structure ($\mathbb{N}, \mathbb{Q}^+$ or $\mathbb{R}^+$) to consider in order to get a polynomial interpretation. This question has no answer as surveyed by the following results:

Theorem 8 (Lucas [26])**.** *There are TRS that can be proved terminating using a polynomial interpretation over $\mathbb{R}$, whereas they cannot be proved terminating using a polynomial interpretation over $\mathbb{Q}$.*

Theorem 9 (Lucas [26])**.** *There are TRS which can be proved terminating using a polynomial interpretation over $\mathbb{Q}$, whereas they cannot be proved terminating using a polynomial interpretation over $\mathbb{N}$.*

Theorem 10 (Middeldorp-Neurauter. [29])**.** *There are TRS which can be proved terminating using a polynomial interpretation over $\mathbb{N}$, whereas they cannot be proved terminating using a polynomial interpretation over $\mathbb{Q}$ or $\mathbb{R}$.*

4.3. Decidability results over polynomials

However we can compare the structures through decidability or undecidability results for the sup-interpretation synthesis problem as illustrated below.

Definition 8 (PI synthesis problem)**.** *Given a TRS $\langle \mathcal{D} \uplus \mathcal{C}, \mathcal{R} \rangle$, is there an assignment $[-]$ such that $[-]$ is a polynomial interpretation of $\langle \mathcal{D} \uplus \mathcal{C}, \mathcal{R} \rangle$?*

Theorem 11. *The PI synthesis problem is undecidable over $\mathbb{N}[\overline{X}]$ and $\mathbb{Q}^+[\overline{X}]$.*

Proof. This is a direct consequence of Hilbert's tenth Problem undecidability since every inequality of Definition 7 of the shape $\forall[\mathtt{x}_1], \ldots, [\mathtt{x}_n], [l] > [r]$, $\mathtt{x}_1, \cdots, \mathtt{x}_n$ being the free variables of l, can be turned into the satisfaction of the formula $\neg\exists[\mathtt{x}_1], \ldots, [\mathtt{x}_n],\ [l] - [r] = 0$. The interested reader should refer to [30]. Note that we have not checked that each arbitrary polynomial can be encoded. This technical check which is needed to show a reduction from Hilbert's tenth problem to the PI synthesis will be performed in the next section for the notion of quasi-interpretation. □

This result was historically mentioned to be undecidable by Lankford [8].

Now we show that the polynomial interpretation synthesis problem is decidable over $\mathbb{R}^+$ as a corollary of Tarski's Theorem [31]. Historically, Tarski's procedure was non-elementary. It has been improved by Collins [32] in a procedure of complexity doubly exponential in the number of variables. We will use the most precise upper bound on such a procedure known by the author

and described in [33], where the procedure is shown to be doubly exponential in the number of quantifiers blocks alternations and exponential in the number of variables, in order to exhibit a precise upper bound on the complexity of the PI synthesis problem: we will obtain an *exponential* procedure because the polynomial quasi-interpretation synthesis problem is more restricted than the general quantifiers elimination over $\mathbb{R}^+$ described by Tarski.

Theorem 12 (Roy et Al. [33])**.** *Given an integral domain* k *(i.e. a commutative ring with no zero divisor) included in a real closed field* $\boldsymbol{R}$*, a formula* ϕ *of size* L *in the ordered fields language under prenex normal form with parameters in* $\mathbf{K}$*, containing* m *blocks of quantifiers and* s *polynomials of* n *variables and with coefficients in* k *whose sum of degrees is less or equal to* D*, there is an algorithm of complexity* $O(L)D^{n^{O(m)}}$ *which computes an equivalent quantifier-free formula.*

Theorem 13. *The PI synthesis problem is decidable in exponential time (in the size of the program) over* $\mathbb{R}^+[\overline{X}]$*.*

Proof. We start by encoding the strict monotonicity property: Given a TRS $\langle \mathcal{D} \uplus \mathcal{C}, \mathcal{R} \rangle$, $\mathtt{f} \in \mathcal{D}$ of arity n and an assignment $[-] \in \mathbb{R}^+[\overline{X}]$ such that $[\mathtt{f}]$ is defined, the strict monotonicity property can be encoded by the following first order formula:

$$SM[\mathtt{f}] = \forall X_1, \ldots, X_n, \forall Y_1, \ldots, Y_n,$$
$$\bigwedge_{l \in [1,n]} X_l > Y_l \implies [\mathtt{f}](X_1, \cdots, X_n) > [\mathtt{f}](X_1, \cdots, X_n)$$

In other words, $SM[\mathtt{f}]$ if and only if $[\mathtt{f}]$ is strictly monotonic.
Now we encode the inequalities for each rule of a given program $\langle \mathcal{D} \uplus \mathcal{C}, \mathcal{R} \rangle$: Given a TRS $\langle \mathcal{D} \uplus \mathcal{C}, \mathcal{R} \rangle$, of assignment $[-]$, let $\overline{a}$ be a enumeration of the multiplicative coefficients involved in the polynomials $[\mathtt{f}]$, $\forall \mathtt{f} \in \mathcal{D} \uplus \mathcal{C}$, and define $PI[\langle \mathcal{D} \uplus \mathcal{C}, \mathcal{R} \rangle] = \exists \overline{a} \in \mathbb{R}^+, (\bigwedge_{\mathtt{f} \in \mathcal{D} \uplus \mathcal{C}} SM[\mathtt{f}]) \wedge (\bigwedge_{l \to r \in \mathcal{R}} [l] > [r])$.
$PI[\langle \mathcal{D} \uplus \mathcal{C}, \mathcal{R} \rangle]$ is true if and only if there is an assignment $[-]$ that is a polynomial interpretation of $\langle \mathcal{D} \uplus \mathcal{C}, \mathcal{R} \rangle$.
Performing a careful α-conversion of all the variables occurring in the distinct inequalities of the formula $PI[\langle \mathcal{D} \uplus \mathcal{C}, \mathcal{R} \rangle]$, we can extrude all the quantifiers (existential and universal) to obtain a new formula under prenex normal form with only one alternation between a block of existential quantifiers (encoding the polynomials multiplicative coefficients) and one block of universal quantifiers (encoding program variables).
Now we apply Theorem 12 by setting $\mathbf{K} = \mathbb{R}$ and $\phi = PI[\langle \mathcal{D} \uplus \mathcal{C}, \mathcal{R} \rangle]$ and we obtain an algorithm of complexity $O(|PI[\langle \mathcal{D} \uplus \mathcal{C}, \mathcal{R} \rangle]|)D^{n^{O(m)}}$ which computes an equivalent quantifier-free formula. Note that:

- the size of the formula $PI[\langle \mathcal{D} \uplus \mathcal{C}, \mathcal{R} \rangle]$ is bounded polynomially by the size of the program and exponentially by the maximal degree of the polynomial, which is also bounded by D. Indeed the number of multiplicative coefficients within a polynomial of bounded degree D is exponential in D.

- the number n of variables is bounded polynomially by the size of the program and exponentially by the degree D
- the number m of blocks is bounded by 2

Consequently, the algorithm has a complexity exponential in the size of the program. □

4.4. Drawbacks of (polynomial) interpretations

The previous Subsection has provided a positive result, that is a mechanical way to synthesize the sup-interpretation of a given program. On the other hand, Theorem 6 can be interpreted as a negative result. Indeed, in terms of TRS, termination means that either the evaluation stops on a constructor term $v \in Ter(\mathcal{C})$ or that the evaluation stops on a (undefined) term still containing non-evaluated function symbols in $\mathcal{D}$. In particular, it means that this analysis rejects all the partial functions that diverge on some input domain but still remain bounded on its complement, as illustrated by the following example:

Example 2.

$$\mathtt{f}(x+2) \to \mathtt{f}(x)+2 \qquad \mathtt{f}(0) \to \mathtt{f}(0) \qquad \mathtt{f}(1) \to 1$$

The function $\mathtt{f}$ *computes the identity function on odd numbers whereas it infinitely diverges on even numbers. Consequently, it does not admit any polynomial interpretation whereas we would expect* $\theta(\mathtt{f})(X) = X$ *to be a suitable sup-interpretation.*

5. Quasi-interpretations

5.1. Quasi-interpretations as sup-interpretations

We introduce the notion of quasi-interpretation [34] that, in contrast with (polynomial) interpretations, allows us to study partial functions.

Definition 9 (Quasi-interpretation)**.** *Given a TRS* $\langle \mathcal{D} \uplus \mathcal{C}, \mathcal{R} \rangle$*, a (additive) quasi-interpretation (QI for short) is a monotonic (additive) assignment* $(\!|-|\!)$ *over* $\mathbb{K}$ *satisfying:*

1. $\forall l \to r \in \mathcal{R},\ (\!|l|\!) \geq (\!|r|\!)$
2. $\forall \mathsf{g} \in \mathcal{D} \uplus \mathcal{C}$*, of arity* m*,* $\forall i \in [1,m],\ (\!|\mathsf{g}|\!)(X_1, \cdots, X_m) \geq X_i$

where the quasi-interpretation $(\!|-|\!)$ *is extended canonically to terms as usual.*

Condition 2 is called the *subterm property*. Quasi-interpretations do not tell anything about program termination since the strict ordering of Definition 6 has been replaced by its reflexive closure. Well-foundedness is lost and this is the main reason why such a tool can be adapted to partial functions. With this notion, we obtain a result similar to Theorem 7.

Theorem 14. *Given a program $\langle \mathcal{D} \uplus \mathcal{C}, \mathcal{R} \rangle$ having an additive quasi-interpretation $(\!|-|\!)$ then $(\!|-|\!)$ is a sup-interpretation.*

Proof. The proof is essentially the same as the one in Theorem 7. Strict inequalities are replaced by non-strict inequalities. □

Example 3. *The program of Example 2 admits the following additive quasi-interpretation:* $(\!|0|\!) = 0$, $(\!|+1|\!)(X) = X + 1$ *and* $(\!|\mathtt{f}|\!)(X) = X$. *Indeed, for the first rule, we check:*

$$\begin{aligned}(\!|\mathtt{f}(x+2)|\!) &= (\!|\mathtt{f}|\!)((\!|(x+1)+1|\!)) = X + 2\\ &\geq (\!|\mathtt{f}(x)|\!) + 2 = (\!|(\mathtt{f}(x)+1)+1|\!)\end{aligned}$$

For the second, rule we clearly have $(\!|\mathtt{f}(0)|\!) \geq (\!|\mathtt{f}(0)|\!)$ *and, for the last rule, we have* $(\!|\mathtt{f}(1)|\!) = (\!|\mathtt{f}|\!)((\!|1|\!)) \geq (\!|1|\!)$.

5.2. Quasi-interpretation synthesis problem

The quasi-interpretation synthesis problem was introduced by Amadio in [18] and is prominent in the perspective of practical uses of quasi-interpretation since an algorithm synthesizing a quasi-interpretation of a given program would allow the programmer to automatically perform a static analysis of program resources use on terminating computations. It can be defined as follows:

Definition 10 (QI synthesis problem)**.** *Given a TRS $\langle \mathcal{D} \uplus \mathcal{C}, \mathcal{R} \rangle$, is there an assignment $(\!|-|\!)$ such that $(\!|-|\!)$ is a quasi-interpretation of $\langle \mathcal{D} \uplus \mathcal{C}, \mathcal{R} \rangle$?*

This problem is undecidable in the general case where we consider total functions as a consequence of Rice's Theorem and as illustrated by Corollary 1. Indeed there is no function (and consequently no program) that for a program index given as input provides the smallest index of a sup-interpretation. Consequently, we have to restrict again the set of considered functions. The immediate candidate is the set of polynomials presented in the previous Section. However we choose to add an extra max function. There are many reasons to do so: firstly, max is the smallest function satisfying the subterm condition. Thus it provides the tightest upper bound that we could expect on a function symbol computation. Secondly, it remains stable for the set of polynomials since the max is always bounded by the sum. Lastly, it was not considered in polynomial interpretations for the only reason that it is not strictly monotonic in each of its arguments (i.e. $x > x' \Rightarrow \max(x, y) > \max(x', y)$ does not hold in the case where $y > x$ with $x, x', y \in \mathbb{K}$). We define the set of $\mathsf{MaxPoly}$ functions as follows:

Definition 11. *Let* $\mathsf{MaxPoly}\,\{\mathbb{K}\}$ *be the set of functions obtained using constants and variables ranging over* $\mathbb{K}$ *and arbitrary compositions of the operators* $+, \times$ *and* max.

We exhibit a normalization result on such a set of functions showing that max operator can be restricted to the upper most level:

Proposition 1 (Normalisation)**.** *Each function $Q \in \mathsf{MaxPoly}\{\mathbb{K}\}, Q \neq 0$, can be written into the following normal form:*

$$Q(X_1, \ldots, X_n) = \max(P_1(X_1, \ldots, X_n), \ldots, P_k(X_1, \ldots, X_n))$$

for some $k \geq 1$ and where $P_i \neq 0$ are polynomials.

Proof. By induction on the structure of Q:

- The base case is when Q is a monomial then $Q = \max(Q)$.
- If $Q = Q_1 + Q_2$ then by induction hypothesis $Q_i = \max(P_1^i, \cdots, P_{n_i}^i)$, for $i \in \{1, 2\}$, with P_j^i polynomials. Consequently, $Q = \max(P_1^1, \cdots, P_{n_1}^1) + \max(P_1^2, \cdots, P_{n_2}^2) = \max_{j \leq n_1, k \leq n_2}(P_j^1 + P_k^2)$ since the max operator can be extruded using rules of the shape $\max(Q, R) + P = \max(Q + P, R + P)$ and $\max(\max(P, Q), \max(R, S)) = \max(P, Q, R, S)$.
- In the same way, if $Q = Q_1 \times Q_2$ then $Q = \max_{j \leq n_1, k \leq n_2}(P_j^1 \times P_k^2)$

and so the conclusion. □

Moreover, we show that the satisfaction of an inequality in $\mathsf{MaxPoly}\{\mathbb{K}\}$ can be transformed into an equivalent problem over polynomials, that is an inequality over $\mathsf{MaxPoly}\{\mathbb{K}\}$ can be turned into a conjunction of disjunctions of inequalities over polynomials:

Proposition 2. *Given an inequality $Q \geq Q'$, with $Q, Q' \in \mathsf{MaxPoly}\{\mathbb{K}\}$ there are two integers n and m and polynomials over $\mathbb{K}$, P_i, R_j for $i \leq n$, $j \leq m$, such that:*

$$Q \geq Q' \text{ iff } \bigwedge_{j \in [1,m]} \bigvee_{i \in [1,n]} P_i \geq R_j$$

Proof. By the previous Proposition, Q and Q' can be written as $\max(P_1, \ldots, P_n)$ and $\max(R_1, \ldots, R_m)$, for some n and m. Consequently:

$$\begin{aligned} &\max(P_1, \ldots, P_n) \geq \max(R_1, \ldots, R_m) \\ \Leftrightarrow &\bigwedge_{j \in [1,m]} \max(P_1, \ldots, P_n) \geq R_j \\ \Leftrightarrow &\bigwedge_{j \in [1,m]} \bigvee_{i \in [1,n]} P_i \geq R_j \end{aligned}$$

and so the result holds. □

5.3. Undecidable synthesis over Max-Poly$\{\mathbb{N}\}$

As expected, the QI synthesis problem remains undecidable over $\mathbb{N}$ and $\mathbb{Q}^+$:

Theorem 15. *The QI synthesis problem is undecidable over $\mathsf{MaxPoly}\{\mathbb{N}\}$ and $\mathsf{MaxPoly}\{\mathbb{Q}^+\}$.*

Proof. We demonstrate, using Proposition 2, that the synthesis problem over $\mathsf{MaxPoly}\{\mathbb{N}\}$ and $\mathsf{MaxPoly}\{\mathbb{Q}^+\}$ can be turned in the satisfaction of (disjunctions and conjunctions of) inequalities of the shape[2]:

$$\exists a_1, \cdots, a_n \forall x_1, \cdots, x_m,\ P(a_1, \cdots, a_n, x_1, \cdots, x_m) \geq 0$$

where the a_i represent the multiplicative coefficients of the function symbols quasi-interpretations and where the x_j represent the program variables quasi-interpretations. Fixing the a_i, this problem consists in checking that:

$$\forall x_1, \cdots, x_m,\ P'(x_1, \cdots, x_m) \geq 0$$

with $P'(x_1, \cdots, x_m) = P(a_1, \cdots, a_n, x_1, \cdots, x_m)$. Now we consider Hilbert's tenth problem that was shown to be undecidable over $\mathbb{Q}^+$ (and $\mathbb{N}$) by Matijasevich [35]. Given a polynomial P of arity n, there is no procedure that decides:

$$\exists x_1, \cdots, x_n,\ P(x_1, \cdots, x_n) = 0$$

Over $\mathbb{N}$, we have:

$$\begin{aligned}
&\exists x_1, \cdots, x_n,\ P(x_1, \cdots, x_n) = 0\\
&\iff \neg(\forall x_1, \cdots, x_n,\ P(x_1, \cdots, x_n)^2 > 0)\\
&\iff \neg(\forall x_1, \cdots, x_n,\ P(x_1, \cdots, x_n)^2 - 1 \geq 0)
\end{aligned}$$

Given a polynomial P, having a computable procedure that checks whether $\forall x_1, \cdots, x_n,\ P(x_1, \cdots, x_n)^2 - 1 \geq 0$ holds would provide a positive answer to Hilbert's problem (and conversely). As a consequence, we know that there is no such a procedure. Finally, we check (a technical but not difficult fact) that for any polynomial P of arity n we can enforce the interpretation of a n-ary symbol $\mathtt{f}$ to satisfy $(\!|\mathtt{f}|\!)(x_1, \cdots, x_n) = P(x_1, \cdots, x_n)^2$ and $\forall x_1, \cdots, x_n,\ P(x_1, \cdots, x_n)^2 \geq 1$ adding arbitrary rules to a program (provided that $(\!|\mathtt{f}|\!) \in \mathsf{MaxPoly}\{\mathbb{N}\}$). For simplicity, suppose that we have additive constructors $\mathtt{c}_n$ of arity $n \in \mathbb{N}$ and such that $(\!|\mathtt{c}_0|\!) = 0$ and $(\!|\mathtt{c}_n|\!)(X_1, \cdots, X_n) = \sum_{i=1}^{n} X_i + 1$ for $n \geq 1$, we can encode every natural number n by n compositions of the shape $\underline{n} = \mathtt{c}_1(\ldots \mathtt{c}_1(\mathtt{c}_0)\ldots)$ and we can encode the identity polynomial by adding the following rule:

$$\mathtt{id}(\mathtt{x}) \to \mathtt{id}(\mathtt{id}(\mathtt{x}))$$

One can check that the corresponding inequality constraints its quasi-interpretation to be equal to $(\!|\mathtt{id}|\!)(X) = X$ over $\mathbb{N}$. Moreover we can add arbitrary rules of the shape:

$$\begin{aligned}
\mathtt{id}(\mathtt{c}_n(\mathtt{x}, \ldots, \mathtt{x})) &\to \mathtt{f}_n(\mathtt{x})\\
\mathtt{id}(\mathtt{c}_0) &\to \mathtt{f}_n(\mathtt{c}_0, \ldots, \mathtt{c}_0)\\
\mathtt{f}_n(\mathtt{c}_1(\mathtt{x})) &\to \underbrace{\mathtt{c}_1(\ldots(\mathtt{c}_1(\mathtt{f}_n(\mathtt{x}))\ldots)}_{n\ times}
\end{aligned}$$

[2]This will be shown explicitly in the next Subsection.

in order to force the following interpretation $(\!|\mathtt{f}_n|\!)(X) = n \times X$. In the same spirit we can encode addition by:

$$\mathtt{id}(\mathtt{c}_2(\mathtt{x},\mathtt{y})) \to \mathtt{add}(\mathtt{x},\mathtt{y})$$
$$\mathtt{add}(\mathtt{c}_1(\mathtt{x}),\mathtt{c}_1(\mathtt{y})) \to \mathtt{c}_1(\mathtt{c}_1(\mathtt{add}(\mathtt{x},\mathtt{y})))$$

in order to force $(\!|\mathtt{add}|\!)(X,Y) = X + Y$ and we can encode multiplication by:

$$\mathtt{f}_n(\mathtt{x}) \to \mathtt{mult}(\mathtt{x},\underline{n})$$
$$\mathtt{f}_n(\mathtt{x}) \to \mathtt{mult}(\underline{n},\mathtt{x})$$
$$\mathtt{mult}(\mathtt{c}_1(\mathtt{x}),\mathtt{y}) \to \mathtt{add}(\mathtt{y},\mathtt{mult}(\mathtt{x},\mathtt{y}))$$

in order to force $(\!|\mathtt{mult}|\!)(X,Y) = X \times Y$. We let the reader check that this reasoning can be generalized to any degree. Finally, if $\mathtt{f}$ is the symbol whose interpretation has been forced to encode the polynomial P^2, we add the rule:

$$\mathtt{f}(\mathtt{x}_1,\cdots,\mathtt{x}_n) \to \mathtt{c}_1(\mathtt{c}_0)$$

to encode the inequality $P^2 \geq 1$. Finally, let us remark that the same (but more technical) kind of encoding can be performed over $\mathbb{Q}^+$. □

Since the encoding presented in the proof of previous Theorem does not depend on the use of a max operator we obtain the following corollary:

Corollary 2. *The QI synthesis problem is undecidable over* $\mathbb{N}[\overline{X}]$ *and* $\mathbb{Q}^+[\overline{X}]$.

5.4. Decidable synthesis over Max-Poly$\{\mathbb{R}^+\}$

In order to get a precise upper bound, we define two notions of degree. The first notion, called $\times$-degree, corresponds to the maximal power of a polynomial whereas the second notion, called max-degree, corresponds to the maximal arity of the max function.

Definition 12 (Degrees)**.** *Given a function*[3] $Q \neq 0 \in \mathsf{MaxPoly}\{\mathbb{K}\}$ *of arity* n *and normal form* $\max(P_1,\ldots,P_k)$*, with* P_i *polynomials, then the* max-degree *of* Q *is equal to* k*.*
Moreover, if P_i *is a polynomial of degree* d_i*, where the degree of a* n*-ary polynomial of the shape* $\sum_{l=1}^{k} \alpha_l X_1^{i_1^l} X_2^{i_2^l} \ldots X_n^{i_n^l}$*, with* $\forall l \in [1,k], \alpha_l \neq 0$*, is equal to* $\max_{l\in[1,k]}(\sum_{j=1}^{n} i_j^l)$*, then the* $\times$-degree *is equal to* $\max_{i\in[1,k]} d_i$*.*

These notions of degree are extended to assignments, the degree of an assignment being the maximal degree of a polynomial in its image.

Definition 13. *The assignment* $(\!|-|\!) \in \mathsf{MaxPoly}\{\mathbb{K}\}$ *is in* $\mathsf{MaxPoly}^{(\mathsf{k},\mathsf{d})}\{\mathbb{K}\}$ *if its* $\times$*-degree and its* max*-degree are respectively bounded by the constants* d *and* k*.*

[3]The polynomial 0 will have degrees equal to 0.

Given an assignment $(\!|-|\!) \in \mathsf{MaxPoly}^{(\mathsf{k},\mathsf{d})}\{\mathbb{R}^+\}$ and a function symbol $\mathtt{f}$ of arity n such that $\mathtt{f}$ is in the definition domain of $(\!|-|\!)$. By Proposition 1, the assignment of $\mathtt{f}$ can be written as follows:

$$(\!|\mathtt{f}|\!)(\overline{X}) = \max(P[\mathtt{f},1](\overline{X}),\ldots,P[\mathtt{f},\mathsf{k}](\overline{X}))$$

where $\overline{X} = X_1,\ldots,X_n$ and $P[\mathtt{f},i]$ are polynomials of degree at most d. In other words:

$$P[\mathtt{f},i](\overline{X}) = \sum a[\mathtt{f},i,j_1,\ldots,j_n]X_1^{j_1}\times\cdots\times X_n^{j_n}$$

with $1 \leq i \leq \mathsf{k}$ and $\sum_{\ell=1}^{n} j_\ell \leq \mathsf{d}$ and where the variable $a[\mathtt{f},i,j_1,\ldots,j_n] \in \mathbb{R}^+$. Now we show some intermediate lemmata:

Lemma 2 (Subterm encoding)**.** *Given $\mathtt{f}$ of arity n and an assignment $(\!|-|\!) \in \mathsf{MaxPoly}^{(\mathsf{k},\mathsf{d})}\{\mathbb{R}^+\}$ such that $(\!|\mathtt{f}|\!)$ is defined, the subterm property can be encoded by the following first order formula:*

$$S[\mathtt{f}] = \bigwedge_{j\in[1,n]} S[\mathtt{f},j]$$

with $S[\mathtt{f},j] = \forall X_1,\ldots,X_n, \bigvee_{i\in[1,\mathsf{k}]} P[\mathtt{f},i](\overline{X}) \geq X_j$.
In other words, $S[\mathtt{f}]$ if and only if $(\!|\mathtt{f}|\!)$ is subterm.

Proof. $(\!|\mathtt{f}|\!)$ is subterm iff $\forall X_1,\cdots,X_n,\ (\!|\mathtt{f}|\!)(X_1,\cdots,X_n) \geq \max(X_1,\cdots,X_n)$ iff $\max(P[\mathtt{f},1](\overline{X}),\ldots,P[\mathtt{f},\mathsf{k}](\overline{X})) \geq \max(X_1,\cdots,X_n)$ which is equivalent to $S[\mathtt{f}]$, by Proposition 2. □

Lemma 3 (Monotonicity encoding)**.** *Given $\mathtt{f} \in \mathcal{D}$ of arity n and an assignment $(\!|-|\!) \in \mathsf{MaxPoly}^{(\mathsf{k},\mathsf{d})}\{\mathbb{R}^+\}$ such that $(\!|\mathtt{f}|\!)$ is defined, the monotonicity property can be encoded by the following first order formula:*

$$M[\mathtt{f}] = \forall X_1,\ldots,X_n,\forall Y_1,\ldots,Y_n,$$
$$\bigwedge_{l\in[1,n]} X_l \geq Y_l \implies \bigwedge_{j\in[1,\mathsf{k}]}\bigvee_{i\in[1,\mathsf{k}]} P[\mathtt{f},i](\overline{X}) \geq P[\mathtt{f},j](\overline{Y})$$

In other words, $M[\mathtt{f}]$ if and only if $(\!|\mathtt{f}|\!)$ is monotonic.

Proof. The proof is just an application of Proposition 2. □

Now we relate the degrees of an expression interpretation with respect to the degree of its symbol interpretations. The main reason for doing so is that we need to encode expression interpretations and not only symbol interpretation in order to encode the rewrite rules of a program.

Proposition 3. *Given $(\!|-|\!) \in \mathsf{MaxPoly}^{(\mathsf{k},\mathsf{d})}\{\mathbb{R}^+\}$ and a term t, we have $(\!|t|\!) \in \mathsf{MaxPlus}^{(\mathsf{k}^{|t|},\mathsf{d}^{|t|})}\{\mathbb{R}^+\}$.*

Proof. By induction on the size of a term t. □

Proposition 3 shows that polynomials can be extended to terms. We write:

$$(\!|t|\!)(\overline{X}) = \max(P[t,1](\overline{X}), \ldots, P[t,\mathsf{k}'](\overline{X}))$$

for some $\mathsf{k}' \leq \mathsf{k}^{|t|}$ and with $P[t,j]$ polynomials of degree bounded by $\mathsf{d}^{|t|}$, whenever the considered assignment is of max-degree k and $\times$-degree d.

Lemma 4 (Rule encoding)**.** *Given a TRS $\langle \mathcal{D} \uplus \mathcal{C}, \mathcal{R} \rangle$ and an assignment $(\!|-|\!) \in \mathsf{MaxPoly}^{(\mathsf{k},\mathsf{d})}\{\mathbb{R}^+\}$, for each rule $l \to_{\mathcal{R}} r$, each inequality can be encoded by:*

$$R[l \to r] = \forall X_1, \ldots X_n, \bigwedge_{j \in [1,l]} \bigvee_{i \in [1,n]} P[l,i](\overline{X}) \geq P[r,j](\overline{X})$$

with $n \leq \mathsf{k}^{|l|}$ and $l \leq \mathsf{k}^{|r|}$.
In other words, $R[l \to r]$ if and only if $(\!|l|\!) \geq (\!|r|\!)$ is satisfied.

Proof. By combining Propositions 2 and 3. □

Proposition 4 (QI encoding)**.** *Given a TRS $\langle \mathcal{D} \uplus \mathcal{C}, \mathcal{R} \rangle$, whose symbols have maximal arity n, define the first order formula:*

$$QI[\langle \mathcal{D} \uplus \mathcal{C}, \mathcal{R} \rangle] = \exists a[\mathtt{f}, i, j_1, \ldots, j_n] \in \mathbb{R}^+, (\bigwedge_{\mathtt{g} \in \mathcal{D}} (S[\mathtt{g}] \wedge M[\mathtt{g}])) \wedge (\bigwedge_{l \to_{\mathcal{R}} r \in \mathcal{R}} R[l \to r])$$

$QI[\langle \mathcal{D} \uplus \mathcal{C}, \mathcal{R} \rangle]$ is true if and only if there is an assignment $(\!|-|\!)$ that is a quasi-interpretation of $\langle \mathcal{D} \uplus \mathcal{C}, \mathcal{R} \rangle$.

Proof. All the properties of QI are satisfied by Lemmata 2, 3, and 4 □

Theorem 16. *$\forall \mathsf{k}, \mathsf{d} \in \mathbb{N}$, the QI synthesis problem is decidable in exponential time (in the size of the program) over $\mathsf{MaxPoly}^{(\mathsf{k},\mathsf{d})}\{\mathbb{R}^+\}$.*

Proof. Given a TRS $\langle \mathcal{D} \uplus \mathcal{C}, \mathcal{R} \rangle$, by Proposition 4, the QI synthesis problem can be turned into checking the satisfaction of the formula $QI[\langle \mathcal{D} \uplus \mathcal{C}, \mathcal{R} \rangle]$. Note that we can extrude all the quantifiers of the formula $QI[\langle \mathcal{D} \uplus \mathcal{C}, \mathcal{R} \rangle]$, after a careful α-conversion, obtaining a new formula under prenex normal form with only one alternation between a block of existential quantifiers (encoding the polynomials multiplicative coefficients) and one block of universal quantifiers (encoding program variables) and we apply the same reasoning than in Theorem 13 (using Theorem 12 again). Note that the exponential upper bound lies in the fact that there are only two blocks of quantifiers ($m = 2$). □

Corollary 3. *The QI synthesis problem is decidable in exponential time over $\mathbb{R}^+[\overline{X}]$.*

5.5. Another interest in the use of reals

The interest of considering quasi-interpretations over the reals does not only rely on the decidability result of Theorem 16. Indeed, we have an analog result to Theorem 8 over MaxPoly quasi-interpretations. It states that there exist programs that do not have any quasi-interpretation over $\mathsf{MaxPoly}\{\mathbb{Q}^+\}$ and, a fortiori $\mathsf{MaxPoly}\{\mathbb{N}\}$, but that admit a quasi-interpretation over $\mathsf{MaxPoly}\{\mathbb{R}^+\}$.

Theorem 17. *There are TRS having a quasi-interpretation over* $\mathsf{MaxPoly}\{\mathbb{R}^+\}$*, whereas they do not have any quasi-interpretation over* $\mathsf{MaxPoly}\{\mathbb{Q}^+\}$*.*

Proof. We build such a TRS in order to enforce its quasi-interpretation $(\!|-|\!) \in \mathsf{MaxPoly}\,\{\mathbb{R}^+\}$ to have an irrational coefficient. Our proof is based on additive QI but we claim that there is a similar proof for the general case. The existence of an infinite number of such TRS follows since we can add infinitely many rules with fresh function symbols on such a program. Moreover we may add the following rule:

$$\mathtt{id}(x) \to \mathtt{id}(\mathtt{id}(x))$$

It enforces the function symbol $\mathtt{id}$ to have a quasi-interpretation of the shape $(\!|\mathtt{id}|\!)(X) = X$ (otherwise if $(\!|\mathtt{id}|\!)(X) > X$, we have $(\!|\mathtt{id}|\!)((\!|\mathtt{id}|\!)(X)) > (\!|\mathtt{id}|\!)(X)$ and there is no QI for such a program). Consider a fresh 2-ary function symbol $\mathtt{g}$, a 0-ary constructor symbol $\mathtt{0}$ and a 2-ary constructor symbol $\mathtt{c}$ such that: $(\!|\mathtt{0}|\!) = 0$ and $(\!|\mathtt{c}|\!)(X, Y) = X + Y + 1$. Consider the following rule:

$$\mathtt{id}(\mathtt{0}) \to \mathtt{g}(\mathtt{0}, \mathtt{0})$$

If $(\!|-|\!)$ is a quasi-interpretation of the TRS then the following inequality holds:

$$0 \geq (\!|\mathtt{g}|\!)(0, 0)$$

Now consider adding the rule:

$$\mathtt{id}(\mathtt{c}(\mathtt{c}(y, y), \mathtt{c}(y, y))) \to \mathtt{g}(\mathtt{c}(\mathtt{0}, \mathtt{0}), y)$$

$(\!|-|\!)$ has to satisfy that:

$$4 \times Y + 3 \geq (\!|\mathtt{g}|\!)(1, Y)$$

Consequently, $(\!|\mathtt{g}|\!)(X, Y)$ has a $\times$-degree at most 1 in Y. Otherwise, for an arbitrary large Y, the above inequality is no longer satisfied. Consequently, there is a set I of indexes and polynomials R_i and S_i such that $(\!|\mathtt{g}|\!)(X, Y) = \max_{i \in I}(R_i(X) \times Y + S_i(X))$ and $(\!|\mathtt{g}|\!)(0, 0) = \max_{i \in I}(S_i(0)) = 0$.
Now consider two 1-ary fresh constructor symbols $\mathtt{a}$ and $\mathtt{b}$ such that $(\!|\mathtt{a}|\!)(X) = X + k$ and $(\!|\mathtt{b}|\!)(X) = X + k'$, for some $k, k' \in \mathbb{N}$. Finally, add the following rules:

$$\begin{aligned}
\mathtt{id}(\mathtt{b}(\mathtt{b}(\mathtt{0}))) &\to \mathtt{g}(\mathtt{0}, \mathtt{g}(\mathtt{0}, \mathtt{b}(\mathtt{0}))) \\
\mathtt{id}(\mathtt{b}(\mathtt{0})) &\to \mathtt{g}(\mathtt{0}, \mathtt{a}(\mathtt{0})) \\
\mathtt{g}(\mathtt{0}, \mathtt{a}(\mathtt{0})) &\to \mathtt{b}(\mathtt{0}) \\
\mathtt{g}(\mathtt{0}, \mathtt{b}(\mathtt{0})) &\to \mathtt{a}(\mathtt{a}(\mathtt{0}))
\end{aligned}$$

All these rules correspond to the following inequalities:

$$\begin{aligned} 2 \times k' &\geq \max_{i\in I}(R_i(0))^2 \times k' \\ k' &\geq \max_{i\in I}(R_i(0)) \times k \\ \max_{i\in I}(R_i(0)) \times k &\geq k' \\ \max_{i\in I}(R_i(0)) \times k' &\geq 2 \times k \end{aligned}$$

The first inequality guarantees that $2 \geq \max_{i\in I}(R_i(0))^2$ since $k' \geq 1$. We deduce from second and third inequalities that $k' = \max_{i\in I}(R_i(0)) \times k$. Substituting $\max_{i\in I}(R_i(0)) \times k$ to k' in the last inequality, we obtain $\max_{i\in I}(R_i(0))^2 \times k \geq 2 \times k$ and, consequently, $\max_{i\in I}(R_i(0))^2 \geq 2$, since $k \geq 1$. Finally, $\max_{i\in I}(R_i(0)) = \sqrt{2}$ and the program only admits irrational quasi-interpretations. In particular, it admits the following quasi-interpretation: $(\!|\mathsf{0}|\!) = 0$, $(\!|\mathsf{a}|\!)(X) = X + 1$, $(\!|\mathsf{b}|\!)(X) = X + \sqrt{2}$, $(\!|\mathsf{c}|\!)(X, Y) = X + Y + 1$, $(\!|\mathsf{id}|\!)(X) = X$ and $(\!|\mathsf{g}|\!)(X, Y) = \max(\sqrt{2}(X + 1)Y, X, Y)$. □

5.6. The QI synthesis problem over MaxPlus

5.6.1. NP*-hardness results*

The complexity of the QI synthesis problem over MaxPoly encourage us to consider smaller function sets. In this perspective, Amadio [18] has considered assignments in MaxPlus $\{\mathbb{N}\}$[4]. He has demonstrated that the QI synthesis problem is still a hard problem even on such a small set of functions.

Definition 14. *Let* MaxPlus $\{\mathbb{K}\}$ *be the set of functions obtained using constants and variables ranging over* $\mathbb{K}$ *and arbitrary compositions of the operators* $+$ *and max.*

Now we state a normalization result that is just a corollary of Proposition 1:

Proposition 5. *Each function* $Q \in$ MaxPlus$\{\mathbb{K}\}, Q \neq 0$*, can be written into the following normal form:*

$$Q(X_1, \ldots, X_n) = \max_{i\in I}(\sum_{j=1}^{n} \alpha_{i,j} X_j + a_i)$$

for some finite set of indexes $I \subset \mathbb{N}$ *and coefficients* $\alpha_{i,j}, a_i \in \mathbb{K}$, $\forall i \in I, j \in [1, n]$.

Theorem 18 (Amadio [18])**.** *The additive QI synthesis problem is* NP*-hard over* MaxPlus $\{\mathbb{N}\}$.

[4]Indeed Amadio considers polynomials with variables and additive coefficients over $\mathbb{Q}^+$ but with multiplicative coefficients over $\mathbb{N}$, consequently restricting the shape of allowed interpretations, whereas we will explicitly consider all coefficients in $\mathbb{Q}^+$ when referring to MaxPlus $\{\mathbb{Q}^+\}$. Also note that real numbers are not considered in Amadio's result.

In what follows, we will show that the QI synthesis problem remains NP-hard over MaxPlus $\{\mathbb{R}^+\}$. One could have expected a better result by a naive analogy with linear programming that is P-complete over $\mathbb{R}^+$ and NP-complete over $\mathbb{N}$. This result is inspired by the NP-hardness proof suggested in Amadio [18]. However since the quasi-interpretation coefficients are ranging over $\mathbb{R}^+$ instead of $\mathbb{N}$ it generates some technical encoding problems. Indeed, properties of the shape "If $x + y = 1$ then either $x = 1$ and $y = 0$ or the converse" hold over $\mathbb{N}$ but not over $\mathbb{R}^+$. More constraints are thus needed on the considered TRS to encode a reduction from a NP-complete problem.

Theorem 19. *The additive quasi-interpretation synthesis problem is* NP*-hard over* MaxPlus $\{\mathbb{R}^+\}$.

The complete proof with key-ingredients is in the Subsection 5.6.2. It proceeds by reducing a 3-CNF problem into a synthesis problem for MaxPlus $\{\mathbb{R}^+\}$. The reduction follows Amadio [18]. The main difference is that the property $\sum_{j=1}^{n} \alpha_{i,j} = 1 \Rightarrow (\exists j$ such that $\alpha_{i,j} = 1$ and $\forall k \neq j\ \alpha_{i,k} = 0)$ holds over $\mathbb{N}$ but no longer holds over reals or rationals. We overcome this problem by adding new rules that give sufficient constraints on the considered assignments to allow us to recover such a property. The end of our proof follows Amadio's proof that encodes literals into a synthesis problem: a function symbol $\mathtt{f}_i$ having a quasi-interpretation $(\!|\mathtt{f}_i|\!) = \alpha_1 X_1 + \alpha_2 X_2$ satisfying ($\alpha_1 = 1$ and $\alpha_2 = 2$) or ($\alpha_1 = 2$ and $\alpha_2 = 1$) is associated to each literal $\mathtt{x}_i$ of a 3-CNF formula ϕ. We suppose that some fixed constant $k \geq 1$ (respectively $2k$) is the additive constant corresponding to the interpretation of a constructor symbol $\mathtt{c}$ and is an encoding of the truth value **True** (resp. **False**). If the first literal of a disjunction D in ϕ is $\mathtt{x}_i$, we associate inputs $(\mathtt{c}(\mathtt{0}), \mathtt{0})$ to the function symbol $\mathtt{f}_i$. In this case, we have $(\!|\mathtt{f}_i(\mathtt{c}(\mathtt{0}), \mathtt{0})|\!) = \alpha_1 \times k$ and $(\!|\mathtt{f}_i|\!)$ will correspond to **True** if and only if $\alpha_1 = 1$, that is $(\!|\mathtt{f}_i|\!)(X_1, X_2) = X_1 + 2 \times X_2$. If the first literal of D is $\neg\mathtt{x}_i$, we associate inputs $(\mathtt{0}, \mathtt{c}(\mathtt{0}))$ to the function symbol $\mathtt{f}_i$. In this case, we have $(\!|\mathtt{f}_i(\mathtt{c}(\mathtt{0}), \mathtt{0})|\!) = \alpha_2 \times k$ and $(\!|\mathtt{f}_i|\!)$ will correspond to **True** if and only if $\alpha_2 = 1$, that is $(\!|\mathtt{f}_i|\!)(X_1, X_2) = 2 \times X_1 + X_2$. Finally we require, using constraints (generated by fresh rules) on the QI, that at least one literal (or its negation) is evaluated to k in each disjunction of ϕ by requiring that at most 2 literals of each disjunction are evaluated to **False**. The provided reduction is polynomial in the size of the formula ϕ.

Corollary 4. *The additive quasi-interpretation synthesis problem is* NP*-hard over* MaxPlus $\{\mathbb{Q}^+\}$.

Proof. Just notice that the proof presented in Subsection 5.6.2 also holds on $\mathbb{Q}^+$. □

5.6.2. Proof of NP*-hardness over* MaxPlus$\{\mathbb{R}^+\}$

In this section, we show the NP-hardness of the synthesis problem over MaxPlus$\{\mathbb{R}^+\}$ by exhibiting a reduction of every 3-CNF formula satisfiability problem into a quasi-interpretation synthesis problem. For that purpose, we need some intermediate and technical propositions.

Proposition 6. *Given a TRS $\langle \mathcal{D} \uplus \mathcal{C}, \mathcal{R} \rangle$ having a quasi-interpretation $(\!|-|\!) \in \mathsf{MaxPlus}\{\mathbb{R}^+\}$. For every $\mathtt{f} \in \mathcal{D}$ such that $(\!|\mathtt{f}|\!)(X_1, \cdots, X_n) = \max_{i \in I}(\sum_{j=1}^{n} \alpha_{i,j} \times X_j + a_i)$ we have:*

$$\forall j \leq n,\ \exists i \in I,\ \alpha_{i,j} \geq 1$$

Proof. Suppose that $\exists j \leq n, \forall i \in I,\ \alpha_{i,j} < 1$ holds and let j_0 be the value of index j on which it holds.
Now take the particular values $x_k = 0,\ \forall k \neq j_0$ and $x_{j_0} > \max_{i \in I}(a_i/(1-\alpha_{i,j_0}))$ we have:

$$\begin{aligned}(\!|\mathtt{f}|\!)(x_1, \ldots, x_{j_0}, \ldots, x_n) &= \max_{i \in I}(\alpha_{i,j_0} \times x_{j_0} + a_i) \\ &< \max_{i \in I}(\alpha_{i,j_0} \times x_{j_0} + (1 - \alpha_{i,j_0}) \times x_{j_0}) \\ &< x_{j_0}\end{aligned}$$

Note that x_{j_0} is clearly defined since $\forall i \in I,\ \alpha_{i,j_0} < 1$. Consequently, this contradicts the subterm property stating that $\forall j \leq n,\ \forall X_j \in \mathbb{R}^+,\ (\!|\mathtt{f}|\!)(X_1, \cdots, X_n) \geq X_j$. □

Proposition 7. *There exist a TRS $\langle \mathcal{D} \uplus \mathcal{C}, \mathcal{R} \rangle$ and a function symbol $\mathtt{f} \in \mathcal{D}$ such that if $\langle \mathcal{D} \uplus \mathcal{C}, \mathcal{R} \rangle$ has an additive quasi-interpretation $(\!|-|\!) \in \mathsf{MaxPlus}\{\mathbb{R}^+\}$ then at least one of the following conditions holds:*

1. $(\!|\mathtt{f}|\!)(X_1, \ldots, X_n) = \max(X_1, \ldots, X_n)$
2. $(\!|\mathtt{f}|\!)(X_1, \ldots, X_n) = \max_{i \in I}(\sum_{j=1}^{n} \alpha_{i,j} \times X_j)$ *(i.e. $(\!|\mathtt{f}|\!)(0, \ldots, 0) = 0$)*
3. $(\!|\mathtt{f}|\!)(X_1, \ldots, X_n) = \sum_{j=1}^{n} \alpha_{i,j} \times X_j$

Proof. 1. We show the first equality by generating the rules of $\mathcal{R}$ in order to constraint the quasi-interpretation of $\mathtt{f}$. Suppose that $\mathtt{f}$ admits a quasi-interpretation of the shape $(\!|\mathtt{f}|\!)(X_1, \cdots, X_n) = \max_{i \in I'}(\sum_{j=1}^{n} \alpha_{i,j} \times X_j + a_i)$ and consider adding the following rule:

$$\mathtt{f}(\mathtt{x}_1, \ldots, \mathtt{x}_n) \to \mathtt{f}(\mathtt{f}(\mathtt{x}_1, \ldots, \mathtt{x}_n), \ldots, \mathtt{f}(\mathtt{x}_1, \ldots, \mathtt{x}_n))$$

If $(\!|-|\!)$ is a quasi-interpretation then it has to satisfy:

$$(\!|\mathtt{f}(\mathtt{x}_1, \ldots, \mathtt{x}_n)|\!) \geq \max_{i \in I'}((\sum_{j=1}^{n} \alpha_{i,j}) \times (\!|\mathtt{f}(\mathtt{x}_1, \ldots, \mathtt{x}_n)|\!) + a_i)$$

Consequently, $\forall i \in I', \sum_{j=1}^{n} \alpha_{i,j} \leq 1$. Using Proposition 6, we have that for each j there is a particular $i_j \in I'$ such that $\alpha_{i_j,j} \geq 1$. Combined with previous inequality, it implies that $\alpha_{i_j,j} = 1$ and $\forall l,\ l \neq j, \alpha_{i_j,l} = 0$.
So we can write the quasi-interpretation of $\mathtt{f}$ as follows:

$$(\!|\mathtt{f}|\!)(X_1, \cdots, X_n) = \max(X_1 + a_{i_1}, \ldots, X_n + a_{i_n}, \max_{i \in I}(\sum_{j=1}^{n} \alpha_{i,j} \times X_j + a_i))$$

with $I = I' - \{i_1, \cdots, i_n\}$ and $\forall j \leq n, \forall i \in I,\ \alpha_{i,j} < 1$.
For an arbitrarily large value $x \in \mathbb{R}^+$ (take $x > \max_{i \in I}((a_i - a_{i_1})/(1 -$

$\alpha_{i,1}))))$, we have $(\!|\mathtt{f}|\!)(x,0,\ldots,0) = x + a_{i_1}$, with $a_{i_1} \geq 0$. Indeed $\forall i \in I,\ \alpha_{i,1} \times x + a_i < x + a_{i_1}$. It implies that:

$$\begin{aligned}(\!|\mathtt{f}|\!)(x,0,\ldots,0) &= x + a_{i_1}\\ &\geq (\!|\mathtt{f}|\!)((\!|\mathtt{f}|\!)(x,0,\ldots,0),\ldots,(\!|\mathtt{f}|\!)(x,0,\ldots,0))\\ &\geq (\!|\mathtt{f}|\!)(x+a_{i_1},\ldots,x+a_{i_1})\\ &\geq x + 2 \times a_{i_1}\end{aligned}$$

Consequently, $a_{i_1} = 0$. Since we can perform the same reasoning for each constant a_{i_k}, the quasi-interpretation can be written:

$$(\!|\mathtt{f}|\!)(X_1,\cdots,X_n) = \max(X_1,\ldots,X_n,\max_{i\in I}(\sum_{j=1}^{n}\alpha_{i,j}\times X_j + a_i))$$

with $\sum_{j=1}^{n}\alpha_{i,j} \leq 1$. Now consider adding the following rule to the program:

$$\mathtt{f}(\mathtt{b}(\mathtt{x}_1,\mathtt{0}),\ldots\mathtt{b}(\mathtt{x}_n,\mathtt{0})) \to \mathtt{f}(\mathtt{b}(\mathtt{x}_1,\mathtt{f}(\mathtt{0},\ldots\mathtt{0})),\ldots\mathtt{b}(\mathtt{x}_n,\mathtt{f}(\mathtt{0},\ldots,\mathtt{0})))$$

with $\mathtt{b}$ a constructor symbol such that $(\!|\mathtt{b}|\!)(X,Y) = X + Y + k_{\mathtt{b}}$, $k_{\mathtt{b}} \geq 1$. In order for $(\!|-|\!)$ to be a QI, it is necessary to check that

$$(\!|\mathtt{f}(\mathtt{b}(\mathtt{x}_1,\mathtt{0}),\ldots\mathtt{b}(\mathtt{x}_n,\mathtt{0}))|\!) \geq (\!|\mathtt{f}(\mathtt{b}(\mathtt{x}_1,\mathtt{f}(\mathtt{0},\ldots\mathtt{0})),\ldots\mathtt{b}(\mathtt{x}_n,\mathtt{f}(\mathtt{0},\ldots,\mathtt{0})))|\!)$$

It implies by choosing the particular values $(\!|\mathtt{x}_1|\!) = \ldots = (\!|\mathtt{x}_n|\!) = x \in \mathbb{R}^+$:

$$\max_{i\in I}((\sum_{j=1}^{n}\alpha_{i,j})\times(x+k_{\mathtt{b}})+a_i) \geq \max_{i\in I}((\sum_{j=1}^{n}\alpha_{i,j})\times(x+k_{\mathtt{b}}+\max_{k\in I}(a_k))+a_i)$$

Suppose that l is the index for which $\max_{i\in I}(\sum_{j=1}^{n}\alpha_{i,j})$ is reached. For an arbitrary large x and since $\sum_{j=1}^{n}\alpha_{l,j} = 1$ and $a_l = 0$, we have $x + k_{\mathtt{b}} \geq x + k_{\mathtt{b}} + \max_{k\in I}(a_k)$. It implies $a_k = 0,\ \forall k \in I$. Finally we have:

$$(\!|\mathtt{f}|\!)(X_1,\cdots,X_n) = \max(\max(X_1,\ldots,X_n),\max_{i\in I}(\sum_{j=1}^{n}\alpha_{i,j}\times X_j))$$

with $\sum_{j=1}^{n}\alpha_{i,j} \leq 1$. Since $\forall X_1,\ldots,\forall X_n \in \mathbb{R}^+$, $\max(X_1,\ldots,X_n) \geq \max_{i\in I}(\sum_{j=1}^{n}\alpha_{i,j}\times X_j)$ holds, we obtain:

$$(\!|\mathtt{f}|\!)(X_1,\cdots,X_n) = \max(X_1,\ldots,X_n)$$

2. Now we show the second equality. Given $\mathtt{g}$ a function symbol such that $(\!|\mathtt{g}|\!)(X_1,\cdots,X_n) = \max_{i\in I}(\sum_{j=1}^{n}\alpha_{i,j}\times X_j + a_i)$. We add the rule:

$$\mathtt{id}(\mathtt{d}(\mathtt{x}_1,\ldots,\mathtt{x}_n)) \to \mathtt{d}(\mathtt{g}(\mathtt{0},\ldots,\mathtt{0}),\mathtt{0},\ldots,\mathtt{0})$$

with $\mathtt{id}$ a function symbol such that $(\!|\mathtt{id}|\!)(X) = X$ (There exists such a function symbol by Proposition 7, item (1)) and $\mathtt{d}$ a n-ary constructor symbol such that $(\!|\mathtt{d}|\!)(X_1,\cdots,X_n) = \sum_{i=1}^{n}X_i + k_{\mathtt{d}}$, $k_{\mathtt{d}} \geq 1$. The corresponding assignment has to satisfy $k_{\mathtt{d}} + \sum_{j=1}^{n}X_j \geq k_{\mathtt{d}} + \max_{i\in I}(a_i)$. It implies that $\forall i \in I,\ a_i = 0$. Consequently, $(\!|\mathtt{g}|\!)(X_1,\cdots,X_n) = \max_{i\in I}(\sum_{j=1}^{n}\alpha_{i,j}X_j)$.

3. Consider a function symbol $\mathtt{f}$ of arity n. Its quasi-interpretation can be constrained to be of the shape:

$$(\!|\mathtt{f}|\!)(X_1, \cdots, X_n) = \alpha_1 \times X_1 + \cdots + \alpha_n \times X_n$$

by adding the following rules to the program:

$$\mathtt{id}(\mathtt{c}(\mathtt{x})) \to \mathtt{c}(\mathtt{f}(\mathtt{0}, \ldots, \mathtt{0}, \mathtt{x}, \mathtt{0}, \ldots \mathtt{0}))$$

with $\mathtt{x}$ appearing at the i-th position in the right hand side of the rule, $\forall i \in [1, n]$, $\mathtt{c}$ a 1-ary constructor symbol such that $(\!|\mathtt{c}|\!)(X) = X + k,\ k \geq 1$, and with $\mathtt{id}$ a function symbol such that $(\!|\mathtt{id}|\!)(X) = X$.

□

Proposition 8. *There exist a TRS $\langle \mathcal{D} \uplus \mathcal{C}, \mathcal{R} \rangle$ and a function symbol $\mathtt{f} \in \mathcal{D}$ of arity 2 such that if $\langle \mathcal{D} \uplus \mathcal{C}, \mathcal{R} \rangle$ has an additive quasi-interpretation $(\!|-|\!) \in \mathsf{MaxPlus}\{\mathbb{R}^+\}$ then the following conditions both hold:*

- $(\!|\mathtt{f}|\!)(X_1, X_2) = \alpha_1 \times X_1 + \alpha_2 \times X_2$
- $(\alpha_1 = 1 \wedge \alpha_2 = 2) \vee (\alpha_1 = 2 \wedge \alpha_2 = 1)$

Proof. By Proposition 7, we can enforce a 2-ary function symbol $\mathtt{f}$ to have the following quasi-interpretation by adding arbitrary rules to constraint its quasi-interpretation:

$$(\!|\mathtt{f}|\!)(X_1, X_2) = \max_{i \in I}(\alpha_{i,1} \times X_1 + \alpha_{i,2} \times X_2)$$

We define $\alpha_j = \max_{i \in I}(\alpha_{i,j})$, for $j \in \{1, 2\}$, and $\alpha = \max_{i \in I}(\alpha_{i,1} + \alpha_{i,2})$. These constants satisfy the following inequality $\alpha_1 + \alpha_2 \geq \alpha$. Now add the following rule to the considered TRS:

$$\mathtt{f}(\mathtt{b}(\mathtt{x}_1, \mathtt{0}), \mathtt{b}(\mathtt{x}_2, \mathtt{0})) \to \mathtt{b}(\mathtt{f}(\mathtt{x}_1, \mathtt{0}), \mathtt{f}(\mathtt{0}, \mathtt{x}_1))$$

with $\mathtt{b}$ a 2-ary constructor symbol such that $(\!|\mathtt{b}|\!)(X) = X + k$ and $(\!|\mathtt{0}|\!) = 0$. For the particular values $(\!|\mathtt{x}_1|\!) = (\!|\mathtt{x}_2|\!) = x \in \mathbb{R}^+$, the corresponding quasi-interpretation has to satisfy $\alpha \times (x + k) \geq k + (\alpha_1 + \alpha_2) \times x$. Consequently, for an arbitrarily large x, $\alpha = \alpha_1 + \alpha_2$ and we can write:

$$(\!|\mathtt{f}|\!)(X_1, X_2) = \alpha_1 \times X_1 + \alpha_2 \times X_2$$

since $\exists j \in I,\ \alpha_{j,1} = \alpha_1$ and $\alpha_{j,2} = \alpha_2$. Indeed it implies that $\forall X_1, X_2 \in \mathbb{R}^+,\ \forall i \in I\, i \neq j,\ \alpha_{j,1} \times X_1 + \alpha_{j,2} \times X_2 \geq \alpha_{i,1} \times X_1 + \alpha_{i,2} \times X_2$.
By virtue of the subterm condition, $\alpha_1, \alpha_2 \geq 1$. We add new rules over $\mathtt{f}$ in order to constraint α_1 and α_2 to satisfy the following condition $(\alpha_1 = 1 \wedge \alpha_2 = 2) \vee (\alpha_1 = 2 \wedge \alpha_2 = 1)$:

$$\mathtt{f}(\mathtt{c}(\mathtt{x}_1), \mathtt{c}(\mathtt{x}_2)) \to \mathtt{c}(\mathtt{c}(\mathtt{c}(\mathtt{0})))$$
$$\mathtt{id}(\mathtt{c}(\mathtt{c}(\mathtt{c}(\mathtt{x})))) \to \mathtt{f}(\mathtt{c}(\mathtt{0}), \mathtt{c}(\mathtt{0}))$$

If $(\!|-|\!)$ is a quasi-interpretation, $\mathtt{c}$ is a 1-ary constructor symbol such that $(\!|\mathtt{c}|\!)(X) = X + k$ and $\mathtt{id}$ is a function symbol such that $(\!|\mathtt{id}|\!)(X) = X$ (such a symbol exists by Proposition 7), we deduce from these rules that $\alpha_1 + \alpha_2 = 3$. By adding the rule:

$$\mathtt{id}(\mathtt{c}(\mathtt{c}(\mathtt{x}))) \to \mathtt{f}(\mathtt{f}(\mathtt{0}, \mathtt{c}(\mathtt{0})), \mathtt{0})$$

we check that $2 \times k + x \geq \alpha_1 \times \alpha_2 \times k$. In other words, $2 \geq \alpha_1 \times \alpha_2$. Since $\alpha_1 = 3 - \alpha_2$, we have to check the inequality $\alpha_1^2 - 3 \times \alpha_1 + 2 \geq 0$ with $2 \geq \alpha_1 \geq 1$. The only corresponding solutions are $(\alpha_1 = 1 \wedge \alpha_2 = 2) \vee (\alpha_1 = 2 \wedge \alpha_2 = 1)$. □

Theorem 19. *The additive quasi-interpretation synthesis problem is* NP*-hard over* MaxPlus $\{\mathbb{R}^+\}$.

Proof. We encode the satisfiability of a 3-SAT problem under 3-CNF into a QI synthesis problem. Given a 3-CNF formula ϕ, we generate a TRS $\langle \mathcal{D} \uplus \mathcal{C}, \mathcal{R} \rangle$ such that $\langle \mathcal{D} \uplus \mathcal{C}, \mathcal{R} \rangle$ admits a quasi-interpretation if and only if ϕ is satisfiable. In this perspective, we associate to each literal $\mathtt{x}_i$ appearing in a given 3-CNF formula ϕ, a fresh 2-ary function symbol $\mathtt{f}_i$ and it corresponding rules such that $(\!|\mathtt{f}_i|\!)(X_1, X_2) = \alpha_1^i \times X_1 + \alpha_2^i \times X_2$, with $(\alpha_1^i = 1 \wedge \alpha_2^i = 2) \vee (\alpha_1^i = 2 \wedge \alpha_2^i = 1)$. Note that this is made possible by Proposition 8.
The table of Figure 2 subsumes the distinct values taken by the quasi-interpretation $(\!|\mathtt{f}_i|\!)$ wrt its coefficients and its inputs, for some constructor symbols $\mathtt{c}$ and $\mathtt{0}$ such that $(\!|\mathtt{c}|\!)(X) = X + k$ and $(\!|\mathtt{0}|\!) = 0$.

Coefficients of $(\!\|\mathtt{f}_i\|\!)$: (α_1^i, α_2^i)	inputs: $(\mathtt{x}_1, \mathtt{x}_2)$	Value of: $(\!\|\mathtt{f}_i(\mathtt{x}_1, \mathtt{x}_2)\|\!)$
(1,2)	$(\mathtt{c}(\mathtt{0}), \mathtt{0})$	k
(1,2)	$(\mathtt{0}, \mathtt{c}(\mathtt{0}))$	$2 \times k$
(2,1)	$(\mathtt{c}(\mathtt{0}), \mathtt{0})$	$2 \times k$
(2,1)	$(\mathtt{0}, \mathtt{c}(\mathtt{0}))$	k

Figure 2: Values of $(\!|\mathtt{f}_i|\!)$ wrt its coefficients and inputs

Let the constant k (respectively $2 \times k$) encode the truth value **True** (respectively **False**). If a literal corresponds to **True** (resp. **False**) then we will encode this information by constraining $\mathtt{f}_i$ to have a quasi-interpretation equal to $X_1 + 2 \times X_2$ (resp. $2 \times X_1 + X_2$).
Given a disjunction D of the formula ϕ, there are two possibilities:

(i) If the first literal of D is $\mathtt{x}_i$, we associate inputs $(\mathtt{c}(\mathtt{0}), \mathtt{0})$ to the function symbol $\mathtt{f}_i$. In this case, we have $(\!|\mathtt{f}_i(\mathtt{c}(\mathtt{0}), \mathtt{0})|\!) = \alpha_1^i \times k$ and $(\!|\mathtt{f}_i|\!)$ will correspond to **True** if and only if $\alpha_1^i = 1$, that is $(\!|\mathtt{f}_i|\!)(X_1, X_2) = X_1 + 2 \times X_2$.

(ii) If the first literal of D is $\neg \mathtt{x}_i$, we associate inputs $(\mathtt{0}, \mathtt{c}(\mathtt{0}))$ to the function symbol $\mathtt{f}_i$. In this case, we have $(\!|\mathtt{f}_i(\mathtt{c}(\mathtt{0}), \mathtt{0})|\!) = \alpha_2^i \times k$ and $(\!|\mathtt{f}_i|\!)$ will correspond to **True** if and only if $\alpha_2^i = 1$, that is $(\!|\mathtt{f}_i|\!)(X_1, X_2) = 2 \times X_1 + X_2$.

Using the notation $\phi_D^{\mathtt{x}}$ to represent the arguments of the function symbol encoding $\mathtt{x}$ in the disjunction D, we have:

$$\phi_D^{\mathtt{x}} = \begin{cases} \mathtt{c}(\mathtt{0}), \mathtt{0} \text{ if } \mathtt{x} \text{ appears in } D \\ \mathtt{0}, \mathtt{c}(\mathtt{0}) \text{ if } \neg\mathtt{x} \text{ appears in } D \end{cases}$$

$(\!|\mathtt{f}(\phi_D^{\mathtt{x}})|\!)$ is equal to k if ($(\!|\mathtt{f}|\!)$ corresponds to **True** and $\mathtt{x}$ appears in D) or ($(\!|\mathtt{f}|\!)$ corresponds to **False** and $\neg\mathtt{x}$ appears in D).
$(\!|\mathtt{f}(\phi_D^{\mathtt{x}})|\!)$ is equal to $2 \times k$ if ($(\!|\mathtt{f}|\!)$ corresponds to **True** and $\neg\mathtt{x}$ appears in D) or ($(\!|\mathtt{f}|\!)$ corresponds to **False** and $\mathtt{x}$ appears in D).
It remains to encode disjunctions: To each disjunction D in the formula ϕ and containing literals $\mathtt{x}_i$, $\mathtt{x}_j$ and $\mathtt{x}_l$, we associate the following rule:

$$\mathtt{id}(\mathtt{c}(\mathtt{c}(\mathtt{c}(\mathtt{c}(\mathtt{c}(\mathtt{x})))))) \to \mathtt{f}(\mathtt{f}_i(\phi_D^{\mathtt{x}_i}), \mathtt{f}_j(\phi_D^{\mathtt{x}_j}), \mathtt{f}_l(\phi_D^{\mathtt{x}_l}))$$

$\mathtt{f}$ and $\mathtt{id}$ being symbols defined by rewrite rules such that their quasi-interpretations are defined by $(\!|\mathtt{id}|\!)(X) = X$ and $(\!|\mathtt{f}|\!)(X_1, X_2, X_3) = \alpha_1 \times X_1 + \alpha_2 \times X_2 + \alpha_3 \times X_3$. Note that such symbols exist by items (1) and (3) of Proposition 7. Moreover, by Proposition 6, $\alpha_1, \alpha_2, \alpha_3 \geq 1$. The quasi-interpretation of the obtained TRS has to satisfy:

$$5 \times k + X \geq (\!|\mathtt{f}_i(\phi_D(\mathtt{x}_i))|\!) + (\!|\mathtt{f}_j(\phi_D(\mathtt{x}_j))|\!) + (\!|\mathtt{f}_l(\phi_D(\mathtt{x}_l))|\!)$$

This inequality enforces at least one of the $(\!|\mathtt{f}_p(\phi_D(\mathtt{x}_p))|\!)$ (for $p \in \{i, j, l\}$) to have value k (i.e. to be **True**) and enforces at most two to have value $2k$. Otherwise it is not satisfied because $\neg(5 \times k \geq 6 \times k)$. We encode in the same spirit all the disjunctions of ϕ. Every assignment satisfying ϕ will clearly correspond to the existence of a suitable quasi-interpretation for the program. Indeed, just take $(\!|\mathtt{f}_i|\!)(X_1, X_2) = X_1 + 2 \times X_2$ (resp. $2 \times X_1 + X_2$) for each litteral assigned to **True** (resp. **False**). Conversely, if the program admits a quasi-interpretation then every disjunction maybe evaluated to true by assigning the truth value **True** to each literal corresponding to a quasi-interpretation of the shape $X_1 + 2 \times X_2$. Finally, we have encoded a 3-CNF problem into a QI synthesis problem over MaxPlus$\{\mathbb{R}^+\}$ using a polynomial time reduction. Indeed the number of added rules is linear in the the size the formula since each intermediate proposition only introduce a constant number of new rules in the considered TRS. Consequently, this problem is NP-hard. □

5.6.3. NP*-completeness over* MaxPlus

After studying NP-hardness results over MaxPlus, we are interested in completeness results on this function space. We start to introduce the first result demonstrated by Amadio in [18] over MaxPlus$\{0, 1\}$. Let MaxPlus$\{0, 1\}$ be

the set of functions obtained using constants ranging over $\{0,1\}$ and variables ranging over $\mathbb{Q}^+$ and arbitrary compositions of the operators $+$ and max[5].

Theorem 20 (Amadio [18])**.** *The additive QI synthesis problem is* NP*-complete over* MaxPlus $\{0,1\}$.

We try to extend this result to $\mathbb{N}$ and $\mathbb{Q}^+$. For that purpose, we focus on the *QI verification problem* that consists in checking that an assignment of a given TRS is a quasi-interpretation. We show that this problem can be solved in polynomial time over MaxPlus if we consider assignment of max-degree k and $+$-degree d polynomially bounded by the TRS size. This is not a restrictive condition since most of the TRS admitting a quasi-interpretation in MaxPlus satisfy it. Indeed arity of the max is indexed by the number of rules in the TRS. Each rule may create a new constraint and may consequently increase the max arity by 1. Finally, the arity of the $+$-degree is trivially indexed by the size of expressions in the rules.

Definition 15 ($+$-degree and max-degree)**.** *Given a function Q of arity n in* MaxPlus$\{\mathbb{K}\}$ *of normal form* $\max(P_1,\ldots,P_m)$ *with:*

$$P_i = \sum_{j\in[1,n]} \alpha_{i,j} \times X_j + \alpha_{i,0}$$

its $+$-degree is equal to $\max_{j\in[0,n],i\in[1,m]}\alpha_{i,j}$*. In other words, the $+$-degree of Q is its greatest multiplicative coefficient. Its* max*-degree is equal to m.*

Definition 16. *Let* $\mathsf{MaxPlus}^{(\mathsf{k},\mathsf{d})}\{\mathbb{K}\}$ *be the set of* MaxPlus$\{\mathbb{K}\}$ *functions of $+$-degree bounded by the constant* d *and* max*-degree bounded by the constant* k*. Given a TRS $\langle \mathcal{D}\uplus\mathcal{C},\mathcal{R}\rangle$ and an assignment $(\!|-|\!)$, $(\!|-|\!) \in \mathsf{MaxPlus}^{(\mathsf{k},\mathsf{d})}\{\mathbb{K}\}$ if:*

$$\forall l \to r \in \mathcal{R},\ (\!|l|\!), (\!|r|\!) \in \mathsf{MaxPlus}^{(\mathsf{k},\mathsf{d})}\{\mathbb{K}\}$$

Theorem 21 (Verification)**.** *Given a TRS $\langle \mathcal{D}\uplus\mathcal{C},\mathcal{R}\rangle$ and an assignment $(\!|-|\!) \in$* $\mathsf{MaxPlus}^{(\mathsf{k},\mathsf{d})}\{\mathbb{R}^+\}$*, we can check in polynomial time in* d *and* k *that $(\!|-|\!)$ is a quasi-interpretation of $\langle \mathcal{D}\uplus\mathcal{C},\mathcal{R}\rangle$.*

Proof. Given a TRS $\langle \mathcal{D}\uplus\mathcal{C},\mathcal{R}\rangle$ and an assignment $(\!|-|\!)$, for each rule of the shape $l \to r$, we can compute $(\!|l|\!)$ and $(\!|r|\!)$ in polynomial time relatively to k and d, by definition of $\mathsf{MaxPlus}^{(\mathsf{k},\mathsf{d})}$ assignments. Consequently, it remains to check that the inequalities of the shape $(\!|l|\!) \geq (\!|r|\!)$ are satisfied (we also have to check some inequalities for monotonicity and subterm properties that we omit). The total number of such inequalities is polynomially bounded by the TRS size r. Moreover, by Proposition 2, we can eliminate the max operators so that each inequality is transformed into the conjunctions and disjunction of k^2 inequalities of the shape $P \geq Q$, with $P, Q \in \mathsf{MaxPlus}\{\mathbb{R}^+\}$. Such inequalities have size polynomially bounded by k and d. We can check their satisfaction in

[5]Such functions were called multi-linear polynomials in [18].

polynomial time in these two parameters using linear programming over $\mathbb{R}^+$, iterating this procedure at most $r \times \mathsf{k}^2$. □

Theorem 22. *The additive quasi-interpretation synthesis problem is* NP*-complete over* $\mathsf{MaxPlus}^{(\mathsf{k},\mathsf{d})}\ \{\mathbb{N}\}$*, for* $\mathsf{d} \geq 2$*.*

Proof. The NP-hardness has been demonstrated in Theorem 18. For that purpose, we need a +-degree of at least 2 in our encoding of 3-CNF. We have shown in Theorem 21, that the verification problem that consists in checking for a candidate assignment that it is a quasi-interpretation can be solved in polynomial time (if variables are extended to $\mathbb{R}^+$). It remains to see that the size of each solution is bounded polynomially by the input size (the TRS size): it is the case since its degrees are bounded by constants k and d. □

Theorem 23. *Let* $\mathbb{Q}^+_{\leq \mathsf{d}}$ *be the subset of* $\mathbb{Q}^+$ *such that every rational has both numerator and denominator bounded by* d*. The additive quasi-interpretation synthesis problem is* NP*-complete over* $\mathsf{MaxPlus}^{(\mathsf{k},\mathsf{d})}\ \left\{\mathbb{Q}^+_{\leq \mathsf{d}}\right\}$*, for* $\mathsf{d} \geq 2$*.*

Proof. Every rational from $\mathbb{Q}^+_{\leq \mathsf{d}}$ can be encoded by two integers smaller than d and, consequently, has a size bounded polynomially by d. □

Such a result does not hold in general for $\mathbb{R}^+$ because of the representation problem in such a space: we do not know how to encode the data since a real number is generally not bounded even if we have bounded degrees.[6]

6. Dependency Pair interpretations

6.1. DP-interpretations as sup-interpretations

The notion of sup-interpretation was introduced in [1] in order to increase the intensionality of interpretation methods. One of the main distinction with quasi-interpretations lies in the subterm property (cf. Definition 9): sup-interpretations do not need to satisfy such a property. Consequently, the subterm property drastically restricts the sup-interpretation space. For example, a function defined by $\mathtt{f}(x, y) \to x$ has a QI at least equal to $(\!|\mathtt{f}|\!)(X, Y) = \max(X, Y)$ whereas one would expect its sup-interpretation to be equal to $\theta(\mathtt{f})(X, Y) = X$ since the second parameter is dropped. To overcome this problem, we introduce a new notion of sup-interpretations, namely DP-interpretations, based on the notion of dependency pair (DP) introduced by Arts and Giesl [6] for showing program termination. Note that a similar notion was introduced in [17] for characterizing FPTIME but was not related to the notion of sup-interpretation. A last point to mention is that DP-interpretations are not a DP-method since they do not ensure termination but rather a method for space analysis inspired by DP-methods. We start by briefly reviewing the notion of dependency pair:

[6] This is Corrigendum to [7] where it was wrongly stated that the QI synthesis problem is NP-complete over $\mathsf{MaxPlus}^{(\mathsf{k},\mathsf{d})}\ \{\mathbb{R}^+\}$.

Definition 17 (DP)**.** *Given a TRS $\langle \mathcal{D} \uplus \mathcal{C}, \mathcal{R}\rangle$, the set of dependency pair symbols $\mathcal{D}^\sharp$ is defined by $\mathcal{D}^\sharp = \mathcal{D} \cup \{\mathtt{f}^\sharp \mid \mathtt{f} \in \mathcal{D}\}$, $\mathtt{f}^\sharp$ being a fresh function symbol of the same arity as $\mathtt{f}$. Given a term $t = \mathtt{f}(t_1, \cdots, t_n)$, let $t^\sharp$ be a notation for $\mathtt{f}^\sharp(t_1, \cdots, t_n)$.*
A dependency pair is a pair $l^\sharp \to u^\sharp$ if $u^\sharp = \mathtt{g}^\sharp(t_1, \cdots, t_n)$, for some $\mathtt{g} \in \mathcal{D}$, and if there is a context $\mathsf{C}[\diamond]$ such that $l \to \mathsf{C}[u] \in \mathcal{R}$ and u is not a proper subterm of l. Let $DP(\mathcal{R})$ be the set of all dependency pairs in $\langle \mathcal{D} \uplus \mathcal{C}, \mathcal{R}\rangle$.

Definition 18 (DP-interpretation)**.** *Given a TRS $\langle \mathcal{D} \uplus \mathcal{C}, \mathcal{R}\rangle$, a (additive) DP-interpretation (DPI for short) is a monotonic (additive) assignment $[\![-]\!]$ over $\mathbb{K}$ extended to $\mathcal{D}^\sharp$ by $\forall \mathtt{f}, \in \mathcal{D},\ [\![\mathtt{f}^\sharp]\!] = [\![\mathtt{f}]\!]$ and which satisfies:*

1. $\forall l \to r \in \mathcal{R},\ [\![l]\!] \geq [\![r]\!]$
2. $\forall l^\sharp \to u^\sharp \in DP(\mathcal{R}),\ [\![l^\sharp]\!] \geq [\![u^\sharp]\!]$

where the DP-interpretation $[\![-]\!]$ is extended canonically to terms as usual.

Notice that the main distinction with QI is that the subterm property has been replaced by Condition 2. We obtain a result similar to Theorem 14:

Theorem 24. *Given a TRS $\langle \mathcal{D} \uplus \mathcal{C}, \mathcal{R}\rangle$ having an additive DP-interpretation $[\![-]\!]$ then $[\![-]\!]$ is a sup-interpretation.*

Moreover, we can show that every quasi-interpretation is a DP-interpretation.

Theorem 25. *Given a TRS $\langle \mathcal{D} \uplus \mathcal{C}, \mathcal{R}\rangle$ having a quasi-interpretation $(\!|-|\!)$, $(\!|-|\!)$ is a DP-interpretation.*

Proof. By Definition 9, $(\!|-|\!)$ is a monotonic assignment which satisfies $\forall l \to r \in \mathcal{R},\ (\!|l|\!) \geq (\!|r|\!)$. Now, take $s^\sharp \to t^\sharp \in DP(\mathcal{R})$. By definition, there is a context $\mathsf{C}[\diamond]$ such that $s \to_\mathcal{R} \mathsf{C}[t] \in \mathcal{R}$. For each term t, $(\!|\mathsf{C}[t]|\!) \geq (\!|t|\!)$ since $(\!|\mathsf{C}[t]|\!)$ is obtained by composition of subterm functions (the subterm property is stable by composition). Consequently, $(\!|s^\sharp|\!) = (\!|s|\!) \geq (\!|\mathsf{C}[t]|\!) \geq (\!|t|\!) = (\!|t^\sharp|\!)$. □

As expected, the converse property does not hold. There are TRS that admit an (additive) DP-interpretation but no (additive) quasi-interpretation, as illustrated by the following example:

Example 4.

$$\begin{array}{ll} \mathtt{half}(0) \to 0 & \mathtt{half}(1) \to 0 \\ \mathtt{half}(x+2) \to \mathtt{half}(x) + 1 & \\ \mathtt{log}(x+2) \to \mathtt{log}(\mathtt{half}(x+2)) + 1 & \mathtt{log}(1) \to 0 \end{array}$$

The above TRS has no additive quasi-interpretation since an additive quasi-interpretation such that $(\!|+1|\!)(X) = X + k$, for $k \geq 1$, would have to satisfy the following inequalities:

$$\begin{aligned} (\!|\mathtt{log}(x+2)|\!) &\geq (\!|\mathtt{log}(\mathtt{half}(x+2)) + 1|\!) \\ &\geq (\!|\mathtt{log}(\mathtt{half}(x+2))|\!) + k \geq (\!|\mathtt{log}(x+2)|\!) + k \end{aligned}$$

By subterm and monotonicity properties. However, we let the reader check that it admits the following additive DP-interpretation $[\![0]\!] = 1$, $[\![+1]\!](X) = X + 1$, $[\![\mathtt{half}]\!](X) = (X + 1)/2$ *and* $[\![\mathtt{log}]\!](X) = 2 \times X$.

6.2. Decidability results over $\mathbb{K}[\overline{X}]$, $\mathsf{MaxPoly}\{\mathbb{K}\}$ *and* $\mathsf{MaxPlus}\{\mathbb{K}\}$

In this section, we review the results of the DPI synthesis problem.

Definition 19 (DPI synthesis problem)**.** *Given a TRS* $\langle \mathcal{D} \uplus \mathcal{C}, \mathcal{R} \rangle$*, is there an assignment* $[\![-]\!]$ *such that* $(\!|-|\!)$ *is a DP-interpretation of* $\langle \mathcal{D} \uplus \mathcal{C}, \mathcal{R} \rangle$*?*

Theorem 26. *The DPI synthesis problem is:*

1. *undecidable over* $\mathsf{MaxPoly}\{\mathbb{N}\}$ *and* $\mathsf{MaxPoly}\{\mathbb{Q}^+\}$
2. *decidable in exponential time over* $\mathsf{MaxPoly}^{(\mathsf{k},\mathsf{d})}\{\mathbb{R}^+\}$
3. *undecidable over* $\mathbb{N}[\overline{X}]$ *and* $\mathbb{Q}^+[\overline{X}]$
4. *decidable in exponential time over* $\mathbb{R}^+[\overline{X}]$

Proof. (1) is a corollary of Theorem 15. The subterm property is withdrawn and replaced by inequalities on dependency pairs. These inequalities do not change the undecidability of the problem. (2) is also a corollary of Theorem 16 using the same reasoning: the encoding of the subterm property is no longer needed and replaced by the encoding of inequalities on DP. Since the number of DP is at most linear in the size of the program, these new inequalities does not impact the complexity of the algorithm. (3) is a consequence of (1) and (4) is a consequence of (2) because polynomials are functions in $\mathsf{MaxPoly}$. □

6.3. NP*-hardness over* $\mathsf{MaxPlus}\{\mathbb{K}\}$

Now we show NP-hardness and NP-completeness results:

Theorem 27. *The additive DPI synthesis problem is:*

1. NP*-hard over* $\mathsf{MaxPlus}\{\mathbb{K}\}$, $\mathbb{K} \in \{\mathbb{N}, \mathbb{Q}^+, \mathbb{R}^+\}$
2. NP*-complete over* $\mathsf{MaxPlus}^{(\mathsf{k},\mathsf{d})}\{\mathbb{N}\}$, $\mathsf{d} \geq 2$
3. NP*-complete over* $\mathsf{MaxPlus}^{(\mathsf{k},\mathsf{d})}\left\{\mathbb{Q}^+_{\leq \mathsf{d}}\right\}$, $\mathsf{d} \geq 2^7$

Proof. (1) We use the encoding in the proof of Theorem 19. There is just one difficulty to face: By Theorem 25 every QI of a given program is a DPI but the converse does not hold. Consequently, it might be easier to find the DPI of a given program than to find its QI, the solution space being greater. Consequently, we have to enforce that each DPI of the reduction is also a QI. This can be done by adding the following rules to the program, $\forall \mathtt{f} \in \mathcal{D}$ of arity n and $\forall i \in \{1, \ldots, n\}$:

$$\mathtt{f}(\mathtt{x}_1, \cdots, \mathtt{x}_n) \to \mathtt{x}_i$$

[7] Cf. previous section

It enforces that the corresponding additive DPI has to satisfy:

$$\forall X_1, \cdots, X_n,\ [\![\mathtt{f}]\!](X_1, \cdots, X_n) \geq \max(X_1, \cdots, X_n)$$

Consequently, $[\![-]\!]$ is DPI then it is a quasi-interpretation and we obtain that the DPI synthesis problem is NP-hard over $\mathsf{MaxPlus}\,\{\mathbb{K}\}$, $\mathbb{K} \in \{\mathbb{N}, \mathbb{Q}^+, \mathbb{Q}^+\}$. (2) is a direct consequence of (1) and Theorem 22 whereas (3) is a consequence of (1) and Theorem 23. □

To conclude, we have found a better notion than the one of quasi-interpretation from an intensional point of view (i.e. in terms of algorithms) in order to get a sup-interpretation at equal cost from a synthesis point of view.

7. Runtime complexity

7.1. Runtime complexity functions as sup-interpretations

As previously stated, sup-interpretation is a tool that inherently deals with space consumption in an extensional way. Consequently, it is natural to link this notion with studies on time consumption of TRS. In an analogy with classical complexity theory, one could expect that a TRS running in polynomial time would lead the programmer to get a polynomial upper bound on the size of the computed value.
A good candidate for the notion of time complexity of a TRS is the notion of *runtime complexity* function, a function providing an upper bound on the length of the longest derivation with respect to the size of the initial term. Many studies have demonstrated that termination techniques can be used to study the runtime complexity of a given TRS. See [11, 12, 22, 36], among others. In this subsection, we show that, as expected, bounding the runtime complexity of a TRS, allows us to recover a sup-interpretation. For that purpose, we introduce usual definitions:

Definition 20. *The* derivational length *of a terminating term t with respect to a rewrite relation $\to_{\mathcal{R}}$ is defined by:*

$$\mathsf{dl}(t, \to_{\mathcal{R}}) = \max\{n \in \mathbb{N} \mid \exists s,\ t \to_{\mathcal{R}}^{n} s\}$$

The runtime complexity function *with respect to a rewrite relation $\to_{\mathcal{S}}$ on a set of terms T is defined by:*

$$\mathsf{rc}(n, T, \to_{\mathcal{S}}) = \max\{\mathsf{dl}(t, \to_{\mathcal{S}}) \mid t \in T \text{ and } |t| \leq n\}$$

The runtime complexity function *with respect to a TRS $\langle \mathcal{D} \uplus \mathcal{C}, \mathcal{R} \rangle$ is defined by*

$$\mathsf{rc}_{\mathcal{R}}(n) = \mathsf{rc}(n, T_b, \to_{\mathcal{R}})$$

where T_b is the set of basic terms of the shape $t = \mathtt{f}(v_1, \cdots, v_n)$ with $\mathtt{f} \in \mathcal{D}$ and $v_1, \cdots, v_n \in Ter(\mathcal{C})$.

We start to show an intermediate result that links the size of a term with respect to the length of its derivation. For that purpose, the size of a TRS $|\langle \mathcal{D} \uplus \mathcal{C}, \mathcal{R}\rangle|$ is defined by $|\langle \mathcal{D} \uplus \mathcal{C}, \mathcal{R}\rangle| = \sum_{l \to r \in \mathcal{R}} |l| + |r|$.

Lemma 5. *Given a TRS $\langle \mathcal{D} \uplus \mathcal{C}, \mathcal{R}\rangle$, for every terms t, s and every $n \in \mathbb{N}$ such that $t \to_{\mathcal{R}}^{n} s$ we have:*

$$|s| \leq |t| \times |\langle \mathcal{D} \uplus \mathcal{C}, \mathcal{R}\rangle|^n$$

Proof. By induction on the derivation length. If $n = 0$ then $t = s$ and we have $|t| \leq |t|$. Now suppose that it holds for a derivation of length $n-1$ by induction hypothesis, that is $t \to_{\mathcal{R}}^{n-1} s$ and $|s| \leq |t| \times |\langle \mathcal{D} \uplus \mathcal{C}, \mathcal{R}\rangle|^{n-1}$ and consider one more rewrite step $s \to_{\mathcal{R}}^{1} u$. By definition of a rewrite step, there is a one-hole context $\mathsf{C}[\diamond]$, a rule $l \to r \in \mathcal{R}$ and a substitution σ such that $s = \mathsf{C}[l\sigma] \to_{\mathcal{R}} \mathsf{C}[r\sigma] = u$. As a result, we obtain that:

$$\begin{array}{lll}
|u| = |\mathsf{C}[r\sigma]| = |\mathsf{C}[\diamond]| + |r\sigma| & & \\
\quad \leq |\mathsf{C}[\diamond]| + |\langle \mathcal{D} \uplus \mathcal{C}, \mathcal{R}\rangle| \times \max_{\mathtt{x} \in Var(r)} |\mathtt{x}\sigma| & \text{Since } |r| \leq |\langle \mathcal{D} \uplus \mathcal{C}, \mathcal{R}\rangle| \\
\quad \leq |\mathsf{C}[\diamond]| + |\langle \mathcal{D} \uplus \mathcal{C}, \mathcal{R}\rangle| \times |l\sigma| & \text{Since } Var(r) \subseteq Var(l) \\
\quad \leq |s| \times |\langle \mathcal{D} \uplus \mathcal{C}, \mathcal{R}\rangle| & \text{Since } |\langle \mathcal{D} \uplus \mathcal{C}, \mathcal{R}\rangle| \geq 1
\end{array}$$

Combining both inequalities, we obtain $|u| \leq |s| \times |\langle \mathcal{D} \uplus \mathcal{C}, \mathcal{R}\rangle| \leq |t| \times |\langle \mathcal{D} \uplus \mathcal{C}, \mathcal{R}\rangle|^{n-1} \times |\langle \mathcal{D} \uplus \mathcal{C}, \mathcal{R}\rangle|$ and so the result. □

Now we relate runtime complexity to sup-interpretations:

Theorem 28. *Given a terminating TRS $\langle \mathcal{D} \uplus \mathcal{C}, \mathcal{R}\rangle$, then the assignment θ defined by:*

- $\theta(\mathtt{c}) = 0$, *if* $\mathtt{c} \in \mathcal{C}$ *is of arity* 0
- $\theta(\mathtt{c})(X_1, \cdots, X_n) = \sum_{i=1}^{n} X_i + 1$, *if* $\mathtt{c} \in \mathcal{C}$ *is of arity* $n > 0$
- $\theta(\mathtt{f})(X_1, \cdots, X_n) = (\sum_{i=1}^{n} X_i + 1) \times |\langle \mathcal{D} \uplus \mathcal{C}, \mathcal{R}\rangle|^{\mathsf{rc}_{\mathcal{R}}(\sum_{i=1}^{n} X_i + 1)}$, *if* $\mathtt{f} \in \mathcal{D}$

is a sup-interpretation.

Proof. Note that the assignment defined is clearly additive and monotonic. Consequently, we have to show that given a symbol $\mathtt{f} \in \mathcal{D}$ of arity n and values $v_1, \cdots, v_n \in Ter(\mathcal{C})$, if $\mathtt{f}(v_1, \cdots, v_n) \downarrow$ then $\theta(\mathtt{f}(v_1, \cdots, v_n)) \geq \theta([\![\mathtt{f}]\!](v_1, \cdots, v_n))$. Note that by definition of θ, $\forall v \in Ter(\mathcal{C})$, $\theta(v) = |v|$. Consider the reduction $\mathtt{f}(v_1, \cdots, v_n) \to_{\mathcal{R}}^{*} [\![\mathtt{f}]\!](v_1, \cdots, v_n)$, we know that there exists $m \in \mathbb{N}$ such that $\mathtt{f}(v_1, \cdots, v_n) \to_{\mathcal{R}}^{m} [\![\mathtt{f}]\!](v_1, \cdots, v_n)$ since the TRS is terminating. Consequently:

$$\begin{array}{lll}
\theta([\![\mathtt{f}]\!](v_1, \cdots, v_n)) = |[\![\mathtt{f}]\!](v_1, \cdots, v_n)| & & \\
\qquad \leq |\mathtt{f}(v_1, \cdots, v_n)| \times |\langle \mathcal{D} \uplus \mathcal{C}, \mathcal{R}\rangle|^{m} & \text{By Lemma 5} \\
\qquad \leq (\sum_{i=1}^{n} |v_i| + 1) \times |\langle \mathcal{D} \uplus \mathcal{C}, \mathcal{R}\rangle|^{rc_{\mathcal{R}}(\sum_{i=1}^{n} |v_i| + 1)} & \text{By Def. 20} \\
\qquad \leq \theta(\mathtt{f})(|v_1|, \ldots, |v_n|) = \theta(\mathtt{f})(\theta(v_1), \ldots, \theta(v_n)) &
\end{array}$$

and so the conclusion. □

7.2. Sup-interpretations through termination techniques

Note that the bound provided in Theorem 28 is exponential in the length of a derivation. In general, it cannot be improved since it relies on the fact that most TRS do not compute terms of size polynomial in the reduction length because they make a strong use of variable duplication. However it can be improved by either syntactically restricting the set of considered TRS, considering for example, linear TRS, TRS that do not replicate their variables, or by semantically restricting the shape of the captured TRS wrt some termination technique fixed in advance. Additive polynomial interpretations described in Section 4 are an example of such a tool. Indeed they only capture programs computing polynomial size values. Note that the restriction lies in the additivity condition and no longer holds if we consider arbitrary interpretations.
We subsume the main termination techniques that can be used to infer sup-interpretations in the Figure 7.2.

Termination technique	SI upper bound	Synthesis problem
Polynomial interpretations	$O(2^{2^n})$	Undecidable
Additive Polynomial interpretations	$O(n^k)$, $k \in \mathbb{N}$	Undecidable
Linear additive interpretations	$O(n)$	P
Restricted Matrix interpretations	$O(n^k)$, $k \in \mathbb{N}$	NP
RPO	$O(f(n))$, $f \in \mathsf{MR}$	NP-complete†
DP-based methods	$O(2^{f(n)})$, $f \in \mathcal{C}_{\mathrm{DP}}$	NP

Figure 3:

In this Figure, the first column lists the termination tool under consideration. The second column provides an upper bound $O(g)$ on the sup-interpretations functions that can be computed with respect to the termination technique under consideration. More precisely, for a n-ary function symbol $\mathtt{f}$ of a TRS whose termination has been shown using some fixed technique, it means that $\theta(\mathtt{f})(X_1, \ldots, X_n) = h(\max_{1 \leq i \leq n}(X_i))$ is a sup-interpretation, for some function h such that $h = O(g)$. Finally, the last column corresponds to the complexity of the respective termination problem.
Now we briefly explain the results of Figure 7.2 line-by-line:

- For polynomial interpretations, the doubly exponential upper bound on the derivation length of a terminating TRS was shown by Hofbauer and

Lautemann in [11]. An important point to stress is that the obtained result for SI is finer than the one that could be expected by a naive application of Theorem 28: indeed the SI upper bound remains doubly exponential (and not a triple exponential). The undecidability result of the synthesis is demonstrated in Section 4 and was suggested by Lankford [8].

- If we consider additive polynomial interpretations, the synthesis remains undecidable however the sup-interpretation codomain is restricted to polynomials because the size of a value in $Ter(\mathcal{C})$ is exactly equal to its size. This result is due to Bonfante et al. [37].

- As a consequence, restriction to linear functions yields a linear upper bound computable in polynomial time using linear programming.

- For matrix interpretation techniques, [38] demonstrates that the runtime complexity is exponentially bounded in the height of a term. As a consequence, we obtain a double exponential upper bound when the height equals the size, by a naive application of Theorem 28. Note that this general framework can be restricted to $O(n^k),\ k \in \mathbb{N}$ using polynomially bounded matrix interpretations of [39, 40] or context dependent interpretations [41] together with restrictions on the interpretation of constructor symbols, in the same spirit as additive polynomial interpretations. See also [42] for a generalization of matrix interpretations. The complexity of the synthesis is in NP because the algorithm that shows the termination of a TRS with matrix interpretations uses a SAT solver.

- The RPO termination technique gives an upper bound exponential in a function $f \in \mathsf{MR}$, where MR stands for the set of multiple recursive functions. This upper bound relies on the lexicographic comparison which yields a multiple recursive derivation length as demonstrated by [36]. This bound can be improved to primitive recursive functions PR if we restrict to Multiset Path Ordering (MPO) as demonstrated by [43]. Note that we obtain the required upper bound on SI since both MR and PR are closed under exponentiation. The NP-completeness of RPO was demonstrated in [44]. († : Note that contrarily to previous mentioned techniques, this technique shows the existence of a SI but does not provide it explicitly.)

- For DP-based methods, Hirokawa and Moser have demonstrated in [45] that techniques combining Dependency Pairs and restricted Interpretations, named SLI for Strongly Linear Interpretations, yields $O(n^2)$ runtime complexity and, consequently, we obtain sup-interpretations in $O(2^{n^2})$ in this particular case. This technique can be generalized to arbitrarily large upper bounds on SI, depending on the base termination technique used. Consequently, the upper bound is $O(2^{f(n)})$ for some $f \in \mathcal{C}_{\mathrm{DP}}$, where $\mathcal{C}_{\mathrm{DP}}$ is a set of runtime complexity functions induced by the DP-method under consideration. For example, the use of RPO as base technique would

give a SI in MR. Note that in this case, the exponential gap can also be withdrawn by putting extra restrictions on the termination system. The synthesis is also in NP because the verification is based on SAT solvers.

One of the main drawbacks in the use of runtime complexity in order to infer sup-interpretations is that we are restricted to terminating TRS and so, to total functions. From that point of view, it is important to stress that QI and DPI based techniques of Sections 5 and 6 allow for such a treatment because they do not imply termination even if they are based on polynomial interpretation methods. Consequently, they may allow the programmer to infer (polynomial) space upper bounds on the computed values (and also the intermediate values in the case of QI) even if the derivation length of the considered term is bounded by a function of high complexity.

8. Conclusion

In this paper, we have studied three methods (interpretations, quasi-interpretations and DP-interpretations) that define a sup-interpretation. Moreover, we have studied the complexity of the sup-interpretation synthesis problem on sets of polynomials including a max operator and we have shown that some termination techniques may allow the programmer to build sup-interpretations. One important issue that falls outside of the scope of this paper concerns the automation of the synthesis problem: in particular the search of efficient algorithms that could allow the programmer to obtain the sup-interpretation of programs (or TRS computing partial functions). Another important issue is to synthesize sup-interpretations through other techniques (type systems, ...). Moreover, we have restricted our discussion to monotonic sup-interpretations. An interesting challenge would be to obtain tighter upper bounds by considering non monotonic functions. It is a very difficult problem since all the techniques known to the author are based on monotonicity conditions. Finally, due to lack of space, we have not studied the synthesis problem over other paradigms. However we let the reader check that finding a sup-interpretation can always be turned into a constraint satisfaction, see [46] for example. Consequently, the complexity results presented in this paper have an impact that is not restricted to term rewriting.

[1] J.-Y. Marion, R. Péchoux, Sup-interpretations, a semantic method for static analysis of program resources, ACM TOCL 10 (4) (2009) 1–31.

[2] M. Bezem, J. Klop, R. de Vrijer, Term rewriting systems, Cambridge University Press, 2003.

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

[4] H. Rogers Jr, Theory of recursive functions and effective computability, MIT Press, 1987.

[5] N. D. Jones, Computability and complexity, from a programming perspective, MIT press, 1997.

[6] T. Arts, J. Giesl, Termination of term rewriting using dependency pairs, Theor. Comput. Sci. 236 (2000) 133–178.

[7] G. Bonfante, J.-Y. Marion, J.-Y. Moyen, R. Péchoux, Synthesis of quasi-interpretations, LCC2005, LICS affiliated Workshop.

[8] D. Lankford, On proving term rewriting systems are noetherian, Tech. rep. (1979).

[9] Z. Manna, S. Ness, On the termination of Markov algorithms, in: Third hawaii international conference on system science, 1970, pp. 789–792.

[10] E. Cichon, P. Lescanne, Polynomial interpretations and the complexity of algorithms, in: CADE, no. 607 in LNAI, 1992, pp. 139–147.

[11] D. Hofbauer, C. Lautemann, Termination proofs and the length of derivations (preliminary version), in: RTA, Vol. 355 of LNCS, Springer, 1989, pp. 167–177.

[12] G. Moser, A. Schnabl, Proving quadratic derivational complexities using context dependent interpretations, in: RTA, Vol. 5117 of LNCS, Springer, 2008, pp. 276–290.

[13] J.-Y. Marion, J.-Y. Moyen, Efficient first order functional program interpreter with time bound certifications, in: LPAR, Vol. 1955 of LNCS, Springer, 2000, pp. 25–42.

[14] G. Bonfante, F. Deloup, Complexity Invariance of Real Interpretations, in: TAMC, Vol. 6108 of LNCS, Springer, 2010, pp. 139–150.

[15] G. Bonfante, J.-Y. Marion, J.-Y. Moyen, On lexicographic termination ordering with space bound certifications, in: PSI, Vol. 2244 of LNCS, Springer, 2001, pp. 482–493.

[16] G. Bonfante, J.-Y. Marion, J.-Y. Moyen, Quasi-interpretations and small space bounds., in: RTA, Vol. 3467 of LNCS, Springer, 2005, pp. 150–164.

[17] J.-Y. Marion, R. Péchoux, Characterizations of polynomial complexity classes with a better intensionality, in: PPDP, ACM, 2008, pp. 79–88.

[18] R. Amadio, Synthesis of max-plus quasi-interpretations, Fundamenta Informaticae 65 (1–2).

[19] C. Fuhs, J. Giesl, A. Middeldorp, P. Schneider-Kamp, R. Thiemann, H. Zankl, Sat solving for termination analysis with polynomial interpretations, in: SAT, LNCS, Springer, 2007, pp. 340–354.

[20] C. Fuhs, J. Giesl, A. Middeldorp, P. Schneider-Kamp, R. Thiemann, H. Zankl, Maximal termination, in: RTA, Vol. 5117 of LNCS, Springer, 2008, pp. 110–125.

[21] C. Borralleras, S. Lucas, A. Oliveras, E. Rodríguez-Carbonell, A. Rubio, Sat modulo linear arithmetic for solving polynomial constraints, J. Autom. Reasoning 48 (1) (2012) 107–131.

[22] M. Avanzini, G. Moser, A. Schnabl, Automated implicit computational complexity analysis (system description), in: IJCAR, Vol. 5195 of LNCS, 2008, pp. 132–138.

[23] G. Bonfante, J.-Y. Marion, R. Péchoux, Quasi-interpretation synthesis by decomposition, in: ICTAC, Vol. 4711 of LNCS, Springer, 2007, pp. 410–424.

[24] S. Kleene, Introduction to metamathematics, Wolters-Noordhoff, 1988.

[25] N. Dershowitz, A note on simplification orderings, Information Processing Letters 9 (5) (1979) 212–215.

[26] S. 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) (2006) 49–73.

[27] E. Contejean, C. Marché, A. Tomás, X. Urbain, Mechanically proving termination using polynomial interpretations, J. Autom. Reasoning 34 (4) (2005) 325–363.

[28] S. Lucas, Practical use of polynomials over the reals in proofs of termination, in: PPDP, ACM, 2007, pp. 39–50.

[29] F. Neurauter, A. Middeldorp, Polynomial interpretations over the reals do not subsume polynomial interpretations over the integers, in: RTA, Vol. 6 of LIPIcs, Springer, 2010, pp. 243–258.

[30] Y. V. Matiyasevich, Hilbert's 10th Problem, Foundations of Computing Series, MIT Press, 1993.

[31] A. Tarski, A Decision Method for Elementary Algebra and Geometry, University of California Press, 1951.

[32] G. E. Collins, Quantifier elimination for real closed fields by cylindrical algebraic decomposition, in: Conference on Automata Theory and Formal Languages, Vol. 33 of LNCS, 1975.

[33] J. Heintz, M.-F. Roy, P. Solerno, Sur la complexité du principe de Tarski-Seidenberg, Bulletin de la S.M.F., tome 118 (1990) 101–126.

[34] G. Bonfante, J.-Y. Marion, J.-Y. Moyen, Quasi-interpretations a way to control resources, Theor. Comput. Sci. 412 (25) (2011) 2776–2796.

[35] Y. Matiyasevich, M. Davis, Hilbert's tenth problem, Vol. 94, MIT press, 1993.

[36] A. Weiermann, Termination proofs for term rewriting systems by lexicographic path orderings imply multiply recursive derivation lengths, Theor. Comput. Sci. 139 (1&2) (1995) 355–362.

[37] G. Bonfante, A. Cichon, J.-Y. Marion, H. Touzet, Algorithms with polynomial interpretation termination proof, Journal of Functional Programming 11 (1) (2001) 33–53.

[38] J. Endrullis, J. Waldmann, H. Zantema, Matrix interpretations for proving termination of term rewriting, J. Autom. Reasoning 40 (2-3) (2008) 195–220.

[39] J. Waldmann, Polynomially bounded matrix interpretations, in: RTA, Vol. 6 of LIPIcs, 2010, pp. 357–372.

[40] F. Neurauter, H. Zankl, A. Middeldorp, Revisiting matrix interpretations for polynomial derivational complexity of term rewriting, in: LPAR (Yogyakarta), Vol. 6397 of LNCS, Springer, 2010, pp. 550–564.

[41] G. Moser, A. Schnabl, J. Waldmann, Complexity analysis of term rewriting based on matrix and context dependent interpretations, in: FSTTCS, Vol. 2 of LIPIcs, 2008, pp. 304–315.

[42] A. Middeldorp, G. Moser, F. Neurauter, J. Waldmann, H. Zankl, Joint spectral radius theory for automated complexity analysis of rewrite systems, in: CAI, Vol. 6742 of LNCS, Springer, 2011, pp. 1–20.

[43] D. Hofbauer, Termination proofs with multiset path orderings imply primitive recursive derivation lengths, Theor. Comput. Sci. 105 (1) (1992) 129–140.

[44] M. S. Krishnamoorthy, P. Narendran, On recursive path ordering, Theor. Comput. Sci. 40 (2-3) (1985) 323–328.

[45] N. Hirokawa, G. Moser, Automated complexity analysis based on the dependency pair method, in: IJCAR, Vol. 5195 of LNCS, Springer, 2008, pp. 364–379.

[46] J.-Y. Marion, R. Péchoux, Analyzing the implicit computational complexity of object-oriented programs, in: FSTTCS, Vol. 2 of LIPIcs, 2008, pp. 316–327.