

# Polynomial Interpretations as a Basis for Termination Analysis of Logic Programs

*Manh Thang Nguyen*

*Danny De Schreye*

*Report CW 412, May 2005*



Katholieke Universiteit Leuven  
Department of Computer Science  
Celestijnenlaan 200A – B-3001 Heverlee (Belgium)

# Polynomial Interpretations as a Basis for Termination Analysis of Logic Programs

*Manh Thang Nguyen*  
*Danny De Schreye*

*Report CW 412, May 2005*

Department of Computer Science, K.U.Leuven

## **Abstract**

This paper introduces a new technique for termination analysis of logic programs based on polynomial interpretations. The principle of this technique is to map each function and predicate symbol to a polynomial over some domain of natural numbers, like it has been done in proving termination of term rewriting systems. Such polynomial interpretations can be seen as a direct generalisation of the traditional techniques in termination analysis of LPs, where (semi-) linear norms and level mappings are used. Our extension generalises these to arbitrary polynomials. We extend a number of standard concepts and results on termination analysis to the context of polynomial interpretations. We propose a constraint based approach for automatically generating polynomial interpretations that satisfy termination conditions.

**Keywords :** Termination analysis, acceptability, polynomial interpretations.

# Polynomial Interpretations as a Basis for Termination Analysis of Logic Programs

Manh Thang Nguyen, Danny De Schreye  
{ManhThang.Nguyen, Danny.DeSchreye}@cs.kuleuven.ac.be

Department of Computer Science, K.U.Leuven  
Celestijnenlaan 200A, B-3001, Heverlee, Belgium

Technical Report CW 412

**Abstract.** This paper introduces a new technique for termination analysis of logic programs based on polynomial interpretations. The principle of this technique is to map each function and predicate symbol to a polynomial over some domain of natural numbers, like it has been done in proving termination of term rewriting systems. Such polynomial interpretations can be seen as a direct generalisation of the traditional techniques in termination analysis of LPs, where (semi-) linear norms and level mappings are used. Our extension generalises these to arbitrary polynomials. We extend a number of standard concepts and results on termination analysis to the context of polynomial interpretations. We propose a constraint based approach for automatically generating polynomial interpretations that satisfy termination conditions.

**Keywords:** Termination analysis, acceptability, polynomial interpretations.

## 1 Introduction

In the last 20 years, the work on termination analysis has been most active for declarative programming languages, with an emphasis on two specific paradigms: logic programming (LP) and term rewrite systems (TRSs). In both areas, the work has been extensive and successful, with many powerful techniques developed and automated tools for these techniques available. However, termination analysis research has evolved very independently for these two paradigms. This has led to two collections of techniques and tools that co-exist without a reasonable level of cross-fertilization between them, nor an acceptable understanding of the portability of these techniques from one paradigm to the other.

Independent of the paradigm, almost every termination analysis is based on a mapping from computational states to some well-founded ordered set. A main difference between LP and TRS is the class of well-founded orderings that are being considered as a basis for the termination proof. For LP, computational states are usually mapped to a well-founded order of the natural numbers. This is usually done through "norms" and "level mappings", that respectively map

terms and atoms to corresponding natural numbers (see [6]). In TRSs, a considerably wider range of well-ordered sets is being considered in the literature, including polynomial interpretations, recursive and lexicographic path orders, and Knuth-Bendix orders (see [24]).

On a most general, methodological level, cross-fertilization of techniques could be organized using two alternative routes: a transformational approach or a direct approach. With a transformational approach, the starting point is the design of a program transformation from one paradigm to the other, such that the transformation preserves the termination behavior. After a transformation, termination analysis techniques for the target language is used on the transformed program in attempt to establish a termination proof of the original program. This line of thought has been successfully used by several researchers in the past (for an overview see [18]). A disadvantage of the transformational approach is that it somewhat obscures the intuitions regarding the termination argument. Often, one is not merely interested in finding a proof of termination as such, but it is more helpful if the proof - or the absence of it - helps us in better understanding the termination behavior of the program at hand.

In this paper we focus on a direct approach of porting techniques, in our case from TRS to LP. Within this context, an initial result to allow porting of more general orderings to the LP setting is presented in [7]. This work provides a new termination condition for logic programs based on general term orders. As such, it can be used as a framework in which different orderings considered in TRSs could be ported to LP directly and be evaluated.

The current paper provides a first step in this study: the use of polynomial interpretations for LP termination analysis. Using polynomial interpretations, as a basic for ordering terms in TRSs, was first introduced by Lankford in [16]. It is currently one of the best known and most widely used techniques in TRS termination analysis.

In this paper, we develop the approach within an LP context. We redefine and extend several known concepts and results from LP termination analysis to polynomial interpretations. We show how polynomial interpretations can be seen as a direct generalisation of currently used techniques in LP termination based on (semi-) linear norms and linear level-mappings. As one would expect, the generalisation is a move from linear polynomial functions to arbitrary polynomials, while the concepts that link the two approaches (standard LP techniques versus TRS polynomial interpretations) are those of the “abstract norm” and “abstract level mapping” [23]. The paper is organised as follows. In the next section, we present some preliminaries on norms, level mappings, rigidity, interargument relations and order acceptability. In section 3, we introduce basic definitions of polynomial interpretations and show how this approach can be used to prove termination with some examples. Next, we discuss the automation of the approach in section 4. We end with a conclusion in section 5.

## 2 Preliminaries

### 2.1 Notations and Terminology

We assume familiarity with logic programming concepts and with the main results of logic programming [1, 17]. In the following,  $L_P$  denotes the language underlying a definite logic program  $P$ . We use  $Var_P$ ,  $Const_P$ ,  $Fun_P$  and  $Pred_P$  to denote the set of variables, constant, function, and predicate symbols of the language  $L_P$ . Given an atom  $A$ ,  $rel(A)$  denotes the predicate occurring in  $A$ . Let  $p, q$  be predicates occurring in the program  $P$ ,  $p$  and  $q$  are called mutually recursive, denoted  $p \simeq q$ , if  $p$  depends on  $q$  in  $P$  and vice versa [2, 20]. Let  $Term_P$  and  $Atom_P$  denote, respectively, sets of all terms and atoms that can be constructed from  $L_P$ . The *extended Herbrand Universe*  $U_P^E$ , and the *extended Herbrand Base*  $B_P^E$  are the quotient sets of  $Term_P$ , and  $Atom_P$  modulo the variant relation [12]. Given two expressions  $E$  and  $F$  (terms, atoms, n-tuples of terms or n-tuples of atoms), we denote  $mgu(E, F)$  their most general unifier.

We focus our attention on SLD-derivations where the left-to-right selection rule is used. Such derivations are referred to as LD-derivations; the corresponding derivation tree as the LD-tree. We say that a query  $Q$  *LD-terminates* for a program  $P$ , if the LD-tree for  $Q \cup P$  is finite (left-termination [17]).

### 2.2 Norms and Level Mappings

Proving termination of logic programs is typically based on *size functions* which map terms and atoms to elements of the well-founded domain. In practice, these are defined in term of a *norm* and in term of a *level-mapping*, which respectively map terms and atoms to corresponding natural numbers.

**Definition 1 (norm, level mapping).** A norm is a mapping  $\|\cdot\| : U_P^E \rightarrow \mathbb{N}$ . A level-mapping is a mapping  $|\cdot| : B_P^E \rightarrow \mathbb{N}$ .

Several examples of norms can be found in literature [4],[9]. One of the most commonly used norms is the list-length norm which maps lists to their lengths and any other term to 0. Another frequently used norm is *term-size* which counts the number of function symbols in the tree representation of a term. Both the list-length and term-size norms are the so-called linear norms [20]. In general, a linear norm is defined as follows:

**Definition 2 (linear norm).** A norm  $\|\cdot\|$  is a linear norm if it is recursively defined by means of the following schema:

- $\|X\| = 0$  for any variable  $X$ ,
- $\|f(t_1, \dots, t_n)\| = f_0 + \sum_{i \in I_f} f_i \|t_i\|$  where  $f_i \in \mathbb{N}$  and the index set  $I_f \subseteq \{1, \dots, n\}$  depend only on the  $n$ -ary function symbol  $f/n$ ,  $n \geq 0$ .

The linear norm generalizes the earlier defined norms used in termination analysis of logic programs. Specifically, it extends the definition of linear norms

in [19] where all the  $f_i, i = \overline{1, \dots, n}$ , are equal to 1 and the definition of semi-linear norms in [4] where each  $f_i, i = \overline{1, \dots, n}$ , is chosen to be 0 or 1.

A more general approach introduced in [23] is to map terms and atoms to corresponding abstract terms in the language  $\mathfrak{L}_{<0,1;+>}$ , which consists of two constant symbols  $\{0, 1\}$  interpreted as the numbers 0 and 1, and the infix operator  $+/2$  interpreted as the usual addition on natural numbers, in addition to the set of variables in the first order language of the given logic program. It is done through a notion of an abstract norm (in [23], it was originally called a norm based abstraction). Here, terms in the language  $\mathfrak{L}_{<0,1;+>}$  are defined as usual. A shorthand form is used to denoted terms, i.e.  $3X + Y + 2$  denotes  $X + X + X + Y + 1 + 1$ . The symbol  $\mathfrak{S}_{<0,1;+>}$  is used for the set of all terms in  $\mathfrak{L}_{<0,1;+>}$ .

**Definition 3 (abstract norm).** *Let  $\|\cdot\|$  be a norm. Terms are mapped to abstract terms by means of the abstract norm  $\|\cdot\|_{abs} : Term_P \rightarrow \mathfrak{S}_{<0,1;+>}$  which is defined recursively as:*

- $\|X\|_{abs} = X$  if  $X$  is a variable,
- $\|c\|_{abs} = \|c\|$  if  $c$  is a constant,
- $\|f(t_1, \dots, t_n)\|_{abs} = f_{\|\cdot\|}(\|t_{i_1}\|_{abs}, \dots, \|t_{i_m}\|_{abs})$ .

where  $f_{\|\cdot\|}$  is a function  $(\mathfrak{S}_{<0,1;+>})^m \rightarrow \mathfrak{S}_{<0,1;+>}$  and  $I = \{i_1, \dots, i_m\}$  is a set of selected argument positions.  $f_{\|\cdot\|}$  and  $I$  depend only on  $f/n$  and  $\|\cdot\|$ .

### 2.3 Conditions for Termination w.r.t. General Orderings

*Acceptability w.r.t. a call set* is a key concept in the automatic termination analysis framework of [8]. It is introduced as a sufficient and necessary condition for termination of a logic program w.r.t. a call set, where computational states are mapped to a well-founded ordering of natural numbers. Another variant of acceptability is the notion of *order-acceptability w.r.t. a call set* in [7]. This concept generalizes the notion of *acceptability w.r.t. a call set* in two ways: (1) it generalizes it to general orderings on  $U_P^E \cup B_P^E$ , (2) it generalizes it to mutual recursion - the original definition of acceptability considers only programs with direct recursion. The results for polynomial interpretations represented in this paper are based on the order-acceptability theory. We will present some main points of this approach. We start with the notion of orderings.

A *quasi-ordering* on a set  $S$  is a reflexive and transitive binary relation  $\succeq$  defined on elements of  $S$ . We define the associated equivalence relation  $\preceq_{\succeq}$  as  $s \preceq_{\succeq} t$  if and only if  $s \succeq t$  and  $t \succeq s$ . If neither  $s \succeq t$ , nor  $t \succeq s$  we write  $\|\cdot\|_{\succeq}$ . To each quasi-ordering  $\succeq$  on  $S$ , we can associate a strict ordering  $\succ$  on  $S$  as  $s \succ t$  if and only if  $s \succeq t$  and it is not the case that  $t \succeq s$ . A strict ordering  $\succ$  is called *well-founded* if there is no infinite sequence  $s_0 \succ s_1 \succ \dots$  with  $s_i \in S$ . Let  $T$  be a set such that  $S \subseteq T$ . A quasi-ordering  $\supseteq$  defined on  $T$  is called a *proper extension* of  $\succeq$  if

- $t_1 \supseteq t_2$  implies  $t_1 \supseteq t_2$  for all  $t_1, t_2 \in S$ .

- $t_1 \succ t_2$  implies  $t_1 \triangleright t_2$  for all  $t_1, t_2 \in S$ .

Before reviewing the notion of order-acceptability w.r.t. a set, we need the following notion.

**Definition 4 (call set).** Let  $P$  be a definite program and  $S$  be a set of atomic queries. The call set,  $\text{Call}(P, S)$ , is the set of all atoms  $A$ , such that a variant of  $A$  is selected atom in some derivation for  $(P, Q)$ , for some  $Q \in S$  and under the left-to-right selection rule.

In practice, the query set  $S$  is specified as a call pattern. The set  $\text{Call}(P, S)$  can be computed by using a type inference technique (e.g.[14]).

**Definition 5 (order-acceptability w.r.t. a set).** [7] Let  $S$  be a set of atomic queries and  $P$  be a definite recursive program.  $P$  is order-acceptable w.r.t.  $S$  if there exists a well-founded ordering  $\succ$  such that

- for any  $A \in \text{Call}(P, S)$ ,
- for any clause  $A' \leftarrow B_1, \dots, B_n$ , such that  $\text{mgu}(A, A') = \theta$  exists,
- for any atom  $B_i$ , such that  $\text{rel}(B_i) \preceq \text{rel}(A)$ ,
- for any computed answer substitution  $\sigma$  for  $\leftarrow (B_1, \dots, B_{i-1})\theta$ :

$$A \succ B_i \theta \sigma$$

The following theorem establishes the link between order acceptability and LD-termination of a logic program.

**Theorem 1.** [7] A program  $P$  LD-terminates under the left-to-right selection rule for any query in  $S$  if and only if  $P$  is order-acceptable w.r.t.  $S$ .

A computed answer substitution is considered as the procedural counterpart of a *correct answer substitution*, that corresponds to a logical consequence of a logic program. In termination analysis, it commonly appears in the form of a *valid interargument relation*. The following notion of *interargument relation* in [7] generalizes the notion of *interargument relation* in [8] to the case of general orderings.

**Definition 6 (interargument relation).** Let  $P$  be a definite program,  $p/n$  be a predicate in  $P$  and  $\succ$  be an ordering on  $U_P^E$ . An interargument relation for  $p$  is a relation  $R_p \subseteq \{(t_1, \dots, t_n) \mid \varphi_p(t_1, \dots, t_n)\}$ , where:

- $\varphi_p(t_1, \dots, t_n)$  is a formula in a disjunctive normal form
- each conjunct in  $\varphi_p$  is either  $s_i \succeq s_j$ ,  $s_i \succ s_j$ ,  $s_i \preceq s_j$  or  $s_i \parallel_{\succ} s_j$ , where  $s_i, s_j$  are constructed from  $t_1, \dots, t_n$  by applying functors of  $P$ .

$R_p$  is a valid interargument relation for  $p/n$  w.r.t. the ordering  $\succ$  if and only if for every  $p(t_1, \dots, t_n) \in B_P^E$ :  $P \models p(t_1, \dots, t_n)$  implies  $p(t_1, \dots, t_n) \in R_p$ .

The concept of *rigidity* is also generalized to adapt to general orderings.

**Definition 7 (rigidity).** A term or atom  $A \in U_P^E \cup B_P^E$  is called rigid w.r.t. a quasi-ordering  $\succeq$  if  $\forall \sigma \in \text{Subs}, A \preceq_{\succeq} A\sigma$ . In this case,  $\succeq$  is said to be rigid on A. A set of terms (or atoms) S is called rigid w.r.t. a quasi-ordering  $\succeq$  if all its elements are rigid w.r.t.  $\succeq$ .

*Example 1.* The list  $[X|t]$  ( $X$  is a variable,  $t$  is a ground term) is rigid w.r.t. the quasi-ordering  $\succeq$  implied by the list-length norm  $\|\cdot\|_l$ . For any substitution  $\sigma$ ,  $\|[X|t]\sigma\|_l = 1 + \|t\|_l = \|[X|t]\|_l$ . Therefore,  $[X|t]\sigma \preceq_{\succeq} [X|t]$ . However, this list is not rigid w.r.t. the quasi-ordering  $\succeq$  implied by the term-size norm  $\|\cdot\|_t$ . For instance, with  $\sigma_1 = \{X/a_1\}$ ,  $a_1$  is a constant,  $\|[X|t]\sigma_1\|_t = 1 + \|t\|_t$ , while with  $\sigma_2 = \{X/[a_1, a_2]\}$   $a_1, a_2$  are constants,  $\|[X|t]\sigma_2\|_t = 3 + \|t\|_t$ . That implies  $[X|t]\sigma_2 \triangleright [X|t]\sigma_1$ .  $\square$

The following notion of *rigid order-acceptability w.r.t. a set of atoms* no longer forces to reason on  $\text{Call}(P, S)$ . Instead, we only need to consider the rigidity of the call set. Furthermore, the condition in this notion is fully at the clause level and the condition on computed answer substitution is replaced by one on valid interargument relations. It is an extension of the notion *rigid-acceptability* in [9] to the case of general orderings.

**Definition 8 (rigid order-acceptability w.r.t. a set).** [7] Let S be a set of atomic queries and P a definite recursive program. Let  $\succeq$  be a quasi-ordering on  $U_P^E$  and for each predicate p in P, let  $R_p$  be a valid interargument relation for p w.r.t.  $\succeq$ . P is rigid order-acceptable w.r.t. S if there exists a proper extension  $\triangleright$  of  $\succeq$  on  $U_P^E \cup B_P^E$ , which is rigid on  $\text{Call}(P, S)$  such that

- for any clause  $H \leftarrow B_1, B_2, \dots, B_n$ ,
- for any atom  $B_i$  in its body such that  $\text{rel}(B_i) \succeq \text{rel}(H)$ ,
- for any substitution  $\theta$  such that the arguments of the atoms in  $(B_1, \dots, B_{i-1})\theta$  all satisfy their associated interargument relations  $R_{B_1}, \dots, R_{B_{i-1}}$ :

$$H\theta \triangleright B_i\theta$$

**Theorem 2.** [7] If P is rigid order-acceptable w.r.t. S, then P is order-acceptable w.r.t. S.

The stated condition of rigid order-acceptability is sufficient for acceptability, but is not necessary for it. A complete proof for this theorem can be found in [21].

### 3 Polynomial Interpretations in Logic Programming

Recall that our objective is to develop and discuss the basic definitions and properties of polynomial interpretations and apply them to prove termination of a logic program. Here, terms and atoms are mapped to polynomials, instead of natural numbers. This will allow to solve a class of problems that the traditional approach can not solve. To illustrate this point, consider the following program, *Der*, that formulates rules for computing the repeated derivative of a function in some variable u. This example was introduced in [7] (see also [11]).

*Example 2 (der).*

$$d(\text{der}(u), 1).$$

$$d(\text{der}(A), 0) : -\text{number}(A).$$

$$d(\text{der}(X + Y), DX + DY) : -d(\text{der}(X), DX), d(\text{der}(Y), DY).$$

$$d(\text{der}(X * Y), X * DY + Y * DX) : -d(\text{der}(X), DX), d(\text{der}(Y), DY).$$

$$d(\text{der}(\text{der}(X)), DDX) : -d(\text{der}(X), DX), d(\text{der}(DX), DDX).$$

We are interested in proving termination of this program w.r.t. the query set  $S = \{d(t_1, t_2) \mid t_1 \text{ is a ground term expressing a derivative of a function in the variable } u, \text{ and } t_2 \text{ is a free variable}\}$ . We consider the first argument of  $d/2$  as an input argument and the second as an output.

Doing this on the basis of a linear norm and level mapping is impossible. The function symbol  $\text{der}/1$  expresses a non-linear relation between the input and output of the original derivative function. In particular, assume that there exists such a linear norm  $\|\cdot\|$  and level mapping  $|\cdot|$  of general forms such that:  $\|u\| = 0$ ,  $\|t_1 + t_2\| = f_0^+ + f_1^+ \|t_1\| + f_2^+ \|t_2\|$ ,  $\|t_1 * t_2\| = f_0^* + f_1^* \|t_1\| + f_2^* \|t_2\|$ ,  $\|\text{der}(t)\| = f_0^d + f_1^d \|t\|$ ,  $|d(t_1, t_2)| = d_0 + d_1 \|t_1\| + d_2 \|t_2\|$ ,  $|\text{number}(t)| = n_0 + n_1 \|t\|$  where  $t, t_1, t_2$  are terms and  $f_0^+, f_1^+, f_2^+, f_0^*, f_1^*, f_2^*, f_0^d, f_1^d, d_0, d_1, d_2, n_0$  and  $n_1$  are non-negative integers. Applying the general constraint based method in [9] shows a contradiction: the system of inequalities that is set up from the acceptability condition is unsolvable. A complete proof can be found in appendix 1. Of course this only proves that one particular approach is unable to prove termination on the basis of linear mappings.  $\square$

Although we have not yet presented our polynomial interpretation approach, it should be intuitively clear, that we can capture the non-linear relation between  $\text{der}(X)$  and  $X$  by using a more complex function, instead of linear norm. It can be done by formally interpreting function and predicate symbols to polynomials over natural numbers. In the following, we will introduce this approach.

### 3.1 Polynomial Interpretations

We start by defining some notations in polynomial theory. Let  $A \subseteq \mathbb{N}$  be a domain. We denote  $\mathbb{P}_{\text{Var}_P}^A$  the set of all polynomials in  $\text{Var}_P$  over  $A$ , with coefficients in  $\mathbb{N}$ . The following definition establishes an ordering on  $\mathbb{P}_{\text{Var}_P}^A$ .

**Definition 9 (polynomial ordering).** *Let  $P$  be a definite logic program and  $A \subseteq \mathbb{N}$  be a domain. Let  $\mathbb{P}_{\text{Var}_P}^A$  be the set of all polynomials in  $\text{Var}_P$  over  $A$ . For polynomials  $P, Q \in \mathbb{P}_{\text{Var}_P}^A$  let  $X_1, \dots, X_n$  be the variables occurring in  $P$  or  $Q$ , and  $F_{P,Q}(X_1, \dots, X_n) = P - Q$ . We define an ordering  $\geq_A$  on  $\mathbb{P}_{\text{Var}_P}^A$  such that  $P \geq_A Q$  if and only if  $F_{P,Q}$  is not negative:  $F_{P,Q}(a_1, \dots, a_n) \geq 0$  for all  $a_1, \dots, a_n$  in  $A$ . The strict ordering  $>_A$  associated to  $\geq_A$  is defined as  $P >_A Q$  if and only if  $F_{P,Q}(a_1, \dots, a_n) > 0$  for all  $a_1, \dots, a_n$  in  $A$ . If  $F_{P,Q}(a_1, \dots, a_n) = 0$  for all  $a_1, \dots, a_n$  in  $A$ , we write  $P \leq_A Q$ . For any other cases,  $P \not\geq_A Q$ .*

It is clear from definition 9 that  $P \leq_A Q$  if and only if  $P \equiv Q$ . Often some coefficients of  $F_{P,Q}$  may be negative.

*Example 3.* Let  $P, Q$  be two polynomials in  $Var_P = \{X_1, X_2, X_3\}$  over a domain  $A$  such that:  $P = 2X_1^2 + 3X_2X_3 + 5X_3$  and  $Q = X_1^2 + 3X_2 + 2X_3 + 4$ . We define a function  $F_{P,Q}(X_1, X_2, X_3) = P - Q = X_1^2 + 3X_2X_3 - 3X_2 + 3X_3 - 4$ . Consider the following cases:

- $A = \mathbb{N} \setminus \{0\}$ . Obviously,  $F_{P,Q}(a_1, a_2, a_3) \geq 0$  for all  $a_1, a_2, a_3 \in A$ . Hence,  $P \geq_A Q$ .
- $A = \mathbb{N} \setminus \{0, 1\}$ . Obviously,  $F_{P,Q}(a_1, a_2, a_3) > 0$  for all  $a_1, a_2, a_3 \in A$ . Hence,  $P >_A Q$ .
- $A = \mathbb{N}$ . For  $a_1 = a_2 = a_3 = 0$ , we have  $F_{P,Q}(a_1, a_2, a_3) = -4 < 0$ . On the other hand, for  $a_1 = a_2 = a_3 = 2$ , we have  $F_{P,Q}(a_1, a_2, a_3) = 12 > 0$ . Hence,  $P \parallel_{\geq_A} Q$ .  $\square$

**Theorem 3.** *The ordering  $>_A$  on the set of polynomials  $\mathbb{P}_{Var_P}^A$  defined in definition 9 is a well founded ordering on  $\mathbb{P}_{Var_P}^A$ .*

*Proof.* The proof is by contradiction. Suppose that  $P_1 >_A P_2 >_A \dots$  is an infinite sequence of polynomials in  $\mathbb{P}_{Var_P}^A$ . Let  $\theta$  be an assignment to each variable in  $Var_P$  of a natural number and  $P_1\theta, P_2\theta, \dots$  be instantiations of  $P_1, P_2, \dots$  w.r.t.  $\theta$ . Obviously, definition 9 implies that  $P_1\theta > P_2\theta > \dots$  is an infinite decreasing sequence of natural numbers, which contradicts the fact that  $>$  is a well-founded ordering on  $\mathbb{N}$ .  $\square$

The idea of polynomial interpretation is to associate each function  $f/n$  in  $Fun_P$ , ( $n \geq 0$ ), with a polynomial  $P_f(X_1, \dots, X_m) \in \mathbb{P}_{Var_P}$ , ( $m \leq n$ ), and each predicate symbol  $p/n$  in  $Pred_P$ , ( $n \geq 1$ ), with a polynomial  $P_p(X_1, \dots, X_m) \in \mathbb{P}_{Var_P}$  ( $m \leq n$ ). Based on this polynomial interpretation, the size of an atom in  $B_P^E$  or a term in  $U_P^E$  is evaluated by using the polynomial corresponding to, respectively, its function or predicate symbol. A well founded ordering on the set of terms and atoms  $U_P^E \cup B_P^E$  implied by the well founded ordering on  $\mathbb{P}_{Var_P}$  is then established. Based on this ordering, we can use the order acceptability condition in [7] to prove termination of the logic program.

**Definition 10 (polynomial pre-interpretation).**

*A polynomial pre-interpretation  $J$  for a language of terms  $L$  consists of:*

- a well-founded domain  $A$  of natural numbers,  $A \subseteq \mathbb{N}$ ,
- an assignment that associates each  $n$ -ary function symbol  $f$  in  $L$  with a polynomial  $P_f(X_{f_1}, \dots, X_{f_m})$  from  $A^m$  to  $A$ , where the coefficients of the polynomial  $P_f/m$  are in  $\mathbb{N}$  and the index set  $I_f = \{f_1, \dots, f_m\} \subseteq \{1, \dots, n\}$  depends only on the  $n$ -ary function symbol  $f/n, n \geq 0$ .

Note that each constant  $c$  in  $L$  can be considered an 0-ary function symbol and is assigned to an element  $c_J$  of domain  $A$ .

The fact that terms are recursively defined from their subterms implies that the domain of a polynomial interpretation should be closed under evaluating the polynomials in  $\mathbb{P}_{Var_P}^A$ , i.e. for all  $f \in Fun_P$  and  $a_1, \dots, a_n \in A$  we have

$P_f(a_1, \dots, a_n) \in A$ . Thus, when selecting a polynomial pre-interpretation, one not only selects an appropriate polynomial associated with each function symbol but also makes sure that the above closure property is satisfied.

A polynomial pre-interpretation  $J$  for a language of terms  $L$  sets up symbolic versions of all function symbols. To transform every term into an abstract version based on  $J$ , we define the following variant of an abstract norm.

**Definition 11 (polynomial norm).** *The polynomial norm associated to a polynomial pre-interpretation  $J$  is a mapping  $\|\cdot\|_J : \text{Term}_P \rightarrow \mathbb{P}_{\text{Var}_P}^A$  which is defined recursively as:*

- $\|X\|_J = X$  if  $X$  is a variable,
- $\|c\|_J = c_J$  if  $c$  is a constant,
- $\|f(t_1, \dots, t_n)\|_J = P_f(\|t_{f_1}\|_J, \dots, \|t_{f_m}\|_J)$ .

where  $P_f(X_1, \dots, X_m)$  and  $I_f = \{f_1, \dots, f_m\}$  are the same as in the definition of the polynomial pre-interpretation  $J$ .

Similarly, we define the notion of a polynomial interpretation that sets up an abstract version of each predicate symbol.

**Definition 12 (polynomial interpretation).** *A polynomial interpretation  $I$  for a language  $L$  underlying a definite logic program  $P$  consists of a polynomial pre-interpretation  $J$  for the language of terms defined by  $L$  extended by*

- *an assignment to each predicate symbol  $p/n$  in  $L$  of a polynomial  $P_p(X_{p_1}, \dots, X_{p_m})$  from  $A^m$  to  $A$ , where the coefficients of the polynomial  $P_p/m$  are in  $\mathbb{N}$  and the index set  $I_p = \{p_1, \dots, p_m\} \subseteq \{1, \dots, n\}$  depends only on the  $n$ -ary predicate symbol  $p/n$ ,  $n \geq 0$ .*

In the same way as a polynomial norm forms an abstract version of a term, the polynomial level-mapping  $|\cdot|_I$  associated to a polynomial interpretation  $I$  can be defined as follows.

**Definition 13 (polynomial level-mapping).** *The polynomial level-mapping associated to a polynomial interpretation  $I$  is a mapping  $|\cdot|_I : \text{Atom}_P \rightarrow \mathbb{P}_{\text{Var}_P}^A$  which is defined as:*

- $|p(t_1, \dots, t_n)|_I = P_p(\|t_{p_1}\|_J, \dots, \|t_{p_m}\|_J)$ .

where  $P_p(X_1, \dots, X_m)$  and  $I_p = \{p_1, \dots, p_m\}$  are as in the definition of the polynomial interpretation  $I$ .

For each term  $t$  and atom  $A$ , we denote  $P_t = \|t\|_J$  and  $P_A = |A|_I$  as the polynomial interpretations of respectively  $t$  and  $A$  in  $I$ .

*Example 4 (Dist).* Consider the following distributive program *Dist*. This example was introduced in [7, 24]:

$$\begin{aligned} & \text{dist}(x, x). \\ & \text{dist}(x * x, x * x). \end{aligned} \tag{1}$$

$$\text{dist}(X + Y, U + V) : -\text{dist}(X, U), \text{dist}(Y, V). \tag{2}$$

$$\text{dist}(X * (Y + Z), T) : -\text{dist}(X * Y + X * Z, T). \tag{3}$$

$$\text{dist}((X + Y) * Z, T) : -\text{dist}(X * Z + Y * Z, T). \tag{4}$$

Let  $I$  be a polynomial interpretation that consists of a domain  $A \subseteq \mathbb{N}$ , an assignment that associates the function symbol  $*/2$  with the polynomial  $P_* := X_1 * X_2$ , the function symbol  $+/2$  with the polynomial  $P_+ := X_1 + X_2 + 1$ , the constant  $x$  with a constant  $c_x \in A$ , and an assignment that associates the predicate symbol  $\text{dist}/2$  with the polynomial  $P_{\text{dist}} := X$ , where the variable  $X$  corresponds to the first argument position of  $\text{dist}/2$ .

The polynomial interpretation of the atom  $A = \text{dist}(U * (X + Y), T)$  in  $I$  is

$$\begin{aligned} P_A &= |\text{dist}(U * (X + Y), T)|_I \\ &= \|U * (X + Y)\|_J \\ &= P_*(\|U\|_J, \|X + Y\|_J) \\ &= P_*(\|U\|_J, P_+(\|X\|_J, \|Y\|_J)) \\ &= \|U\|_J * (\|X\|_J + \|Y\|_J + 1) \\ &= U * (X + Y + 1) \end{aligned}$$

□

We denote  $\mathbb{P}_{\mathbb{I}}$  the set of polynomial interpretations of all elements in  $U_P^E \cup B_P^E$ . Obviously,  $\mathbb{P}_{\mathbb{I}} \subseteq \mathbb{P}_{V_{ar_P}^A}$  and  $>_A$  is a well founded ordering on  $\mathbb{P}_{\mathbb{I}}$ . We define a quasi-ordering on  $U_P^E \cup B_P^E$  implied by the ordering  $>_A$  on  $\mathbb{P}_{\mathbb{I}}$  as follows:

**Definition 14 (ordering on terms and atoms).** *Let  $P$  be a definite program. Let  $I$  be a polynomial interpretation. We define  $\succeq_I$  a quasi-ordering on  $U_P^E$  such that:*

- $t \succ_I s$  if and only if  $P_t >_A P_s$  for any  $t, s \in U_P^E$ ,
- $t \preceq_I s$  if and only if  $P_t \leq_A P_s$  for any  $t, s \in U_P^E$ ,

and  $\triangleright_I$  a proper extension of  $\succeq_I$  on  $U_P^E \cup B_P^E$  such that:

- $B \triangleright_I C$  if and only if  $P_B >_A P_C$  for any  $B, C \in B_P^E$ ,
- $B \triangleleft_I C$  if and only if  $P_B \leq_A P_C$  for any  $B, C \in B_P^E$ .

where  $P_t, P_s, P_B, P_C$  are polynomial interpretations of  $t, s, B$  and  $C$ .

**Theorem 4.** *The strict orderings  $\succ_I$  and  $\triangleright_I$  are respectively well founded orderings on  $U_P^E$  and  $U_P^E \cup B_P^E$ .*

*Proof.* It is easily verified that  $\succ_I$  and  $\triangleright_I$  are well defined orderings. Because  $\succ_A$  is a well-founded ordering on  $\mathbb{P}_1$ , it is trivial that  $\succ_I$  and  $\triangleright_I$  are respectively well-founded orderings on  $U_P^E$  and  $U_P^E \cup B_P^E$ .  $\square$

Integrated with definition 5 and theorem 1 we obtain:

**Proposition 1.** *Let P be a definite program and S be a set of atomic queries. If there exists a polynomial interpretation I such that*

- for any  $A \in \text{Call}(P, S)$ ,
- for any clause  $A' \leftarrow B_1, \dots, B_n$  in P, such that  $\text{mgu}(A, A') = \theta$  exists,
- for any atom  $B_i$ , such that  $\text{rel}(B_i) \preceq \text{rel}(A)$ ,
- for any computed answer substitution  $\alpha$  for  $\leftarrow (B_1, \dots, B_{i-1})\theta$ :

$$P_A \succ_A P_{B_i\theta\alpha}$$

where  $P_A$  denotes the polynomial interpretation of the atom A,

then P left-terminates w.r.t. S.

*Example 5.* Reconsider example 4. We prove termination of the program with the following set of queries:

$S = \{\text{dist}(t_1, t_2) \mid t_1 \text{ is an expression in a variable } x \text{ (} t_1 \text{ is a ground term in the program) and } t_2 \text{ is a free variable}\}$

We choose the polynomial interpretation  $I$  of example 4 except that  $A \subseteq \mathbb{N} \setminus \{0, 1\}$ . Then, the following property holds: for any term  $t \in \text{Term}_P$ ,  $\|t\|_J \succ_A 1$ . Observe that the set  $\text{Call}(S, P) = S$ . Therefore, the first argument of any selected atom in  $\text{Call}(S, P)$  is a ground term. Suppose  $A = \text{dist}(t, s)$  is a selected atom in  $\text{Call}(S, P)$ . There are 3 cases to consider:

- $A = \text{dist}(t_1 + t_2, s)$   $t_1, t_2$  are ground terms and clause (2) is selected. There exists a substitution  $\theta$  such that  $\theta = \text{mgu}(A, \text{dist}(X_1 + Y_1, U_1 + V_1))$ . That implies  $X_1\theta = t_1, Y_1\theta = t_2$ . Therefore,  $|\text{dist}(t_1 + t_2, s)|_I = \|t_1 + t_2\|_J = \|t_1\|_J + \|t_2\|_J + 1 \succ_A \|t_1\|_J = |\text{dist}(X_1, U_1)\theta|_I$  and  $|\text{dist}(t_1 + t_2, s)|_I = \|t_1\|_J + \|t_2\|_J + 1 \succ_A \|t_2\|_J = |\text{dist}(Y_1, V_1)\theta\alpha|_I$  for any computed answer substitution  $\alpha$  of  $\text{dist}(X_1, U_1)$
- $A = \text{dist}(t_1 * (t_2 + t_3), s)$   $t_1, t_2, t_3$  are ground terms and clause (3) is selected. There exists a substitution  $\theta$  such that  $\theta = \text{mgu}(A, \text{dist}(X_1 * (Y_1 + Z_1), T_1))$ . That implies  $X_1\theta = t_1, Y_1\theta = t_2, Z_1\theta = t_3$ . Therefore,  $|\text{dist}(t_1 * (t_2 + t_3), s)|_I = \|t_1 * (t_2 + t_3)\|_I = \|t_1\|_J * \|t_2 + t_3\|_J = \|t_1\|_J * \|t_2\|_J + \|t_1\|_J * \|t_3\|_J + \|t_1\|_J \succ_A \|t_1\|_J * \|t_2\|_J + \|t_1\|_J * \|t_3\|_J + 1 = \|t_1 * t_2 + t_1 * t_3\|_J = |\text{dist}(X_1 * Y_1 + X_1 * Z_1, T_1)\theta|_I$
- $A = \text{dist}((t_1 + t_2) * t_3, s)$   $t_1, t_2, t_3$  are ground terms and clause (4) is selected. There exists a substitution  $\theta$  such that  $\theta = \text{mgu}(A, \text{dist}((X_1 + Y_1) * Z_1, T_1))$ . That implies  $X_1\theta = t_1, Y_1\theta = t_2, Z_1\theta = t_3$ . Therefore,  $|\text{dist}((t_1 + t_2) * t_3, s)|_I = \|(t_1 + t_2) * t_3\|_J = \|t_1 + t_2\|_J * \|t_3\|_J = \|t_1\|_J * \|t_3\|_J + \|t_2\|_J * \|t_3\|_J + \|t_3\|_J \succ_A \|t_1\|_J * \|t_3\|_J + \|t_2\|_J * \|t_3\|_J + 1 = \|t_1 * t_3 + t_2 * t_3\|_J = |\text{dist}(X_1 * Z_1 + Y_1 * Z_1, T_1)\theta|_I$

Therefore,  $P$  is order-acceptable w.r.t.  $S$  and  $P$  terminates on  $S$ .  $\square$

Next, we study rigidity of a call set w.r.t. a polynomial interpretation and use it to verify rigid order acceptability. This provides a more syntactic way of proving termination of a logic program.

### 3.2 Rigidity

First we present the classical notion of monotone polynomials. This class of polynomials is discussed in [3, 24]. Next we study the rigidity of a set of terms w.r.t. a polynomial pre-interpretation that maps terms to monotone polynomials.

**Definition 15 (monotone polynomials).** *A polynomial  $P(X_1, \dots, X_n) \in \mathbb{P}_{Var_P}^A$  from  $A^n$  to  $A$  is called monotone if and only if  $t > s \Rightarrow P(a_1, \dots, a_{i-1}, t, a_{i+1}, \dots, a_n) > P(a_1, \dots, a_{i-1}, s, a_{i+1}, \dots, a_n)$  holds for all  $i, 1 \leq i \leq n$ , and all  $s, t, a_1, \dots, a_{i-1}, a_{i+1}, \dots, a_n \in A$ .*

*Example 6.* Reconsider example 3. Let  $A = \mathbb{N} \setminus \{0\}$  be a domain. It is obvious that both polynomials  $P(X_1, X_2, X_3) = 2X_1^2 + 3X_2X_3 + 5X_3$  and  $Q(X_1, X_2, X_3) = X_1^2 + 3X_2 + 2X_3 + 4$  are monotone polynomials. Meanwhile, the polynomial  $F_{P,Q}(X_1, X_2, X_3) = X_1^2 + 3X_2X_3 - 3X_2 + 3X_3 - 4$  is not monotone since  $F_{P,Q}(1, 1, 1) = F_{P,Q}(1, 2, 1) = 0$ .  $\square$

**Definition 16 (monotone polynomial (pre-)interpretation).** *A polynomial pre-interpretation is called monotone if it associates all function symbols in  $Fun_P$  with monotone polynomials. A polynomial interpretation is monotone if it consists of a monotone polynomial pre-interpretation and an assignment that associates all predicate symbols in  $Pred_P$  with monotone polynomials.*

Usually, when talking about rigidity, we are only interested in rigidity of a set of terms (or atoms) w.r.t. a particular norm (or level mapping). In [4], Bossi, Cocco and Fabris discussed rigidity of  $Call(P, S)$  w.r.t. a semi-linear norm for some  $P$  and  $S$ . It is then generally extended to the case of rigidity of  $Call(P, S)$  w.r.t. a general term ordering in [21]. In this paper, we discuss rigidity of terms (or atoms) w.r.t. a monotone polynomial interpretation and show that it is also an extension to the first one.

Let us recall and extend some basic notions defined in [4].

**Definition 17 (rigidity w.r.t. a polynomial (pre-)interpretation).** *A term  $t \in U_P^E$  is called rigid w.r.t. a polynomial pre-interpretation  $J$  if and only if for any substitution  $\theta$ ,  $\|t\|_J \leq_A \|t\theta\|_J$ . An atom  $A \in B_P^E$  is called rigid w.r.t. a polynomial interpretation  $I$  if and only if for any substitution  $\theta$ ,  $|A|_I \leq_A |A\theta|_I$ . In this case,  $J$  and  $I$  are said to be rigid on, respectively,  $t$  and  $A$ .*

The notion of rigidity on a term or an atom is naturally extended to the notion of rigidity on a set of terms or atoms. In particular, we are interested in polynomial interpretations that are rigid on a call set  $Call(P, S)$  for some  $P$  and  $S$ .

**Definition 18.** Let  $J$  be a polynomial pre-interpretation and  $t$  be a term. The  $i^{\text{th}}$  occurrence,  $X_{(i)}$ , of a variable  $X$  in  $t$  is called relevant w.r.t.  $J$  whenever there exists a replacement  $\{s \rightarrow X_{(i)}\}$  of a term  $s$  for  $X_{(i)}$  such that  $\|t\{s \rightarrow X_{(i)}\}\|_J \neq \|t\|_J$ . We call  $VREL(t)$  the set of all relevant occurrences of variables in  $t$ .

Recall that for two polynomials  $P$  and  $Q$ ,  $P \equiv Q \Leftrightarrow P \leq_A Q$ . Obviously from definition 18, if a term  $t$  is not rigid w.r.t.  $J$ , there must be some relevant occurrence of some variable in  $t$ .

*Example 7.* Consider the term  $t = [X|X]$  and the polynomial pre-interpretation implied by the list-length norm  $\|\cdot\|_l$ . Then,  $VREL(t) = \{X_{(2)}\}$ .  $\square$

The previous definition can be extended to terms as follows.

**Definition 19.** Let  $J$  be a polynomial pre-interpretation and  $t$  be a term. A subterm  $s$  in  $t$  is called relevant w.r.t.  $J$  whenever there exists a replacement  $\{s' \rightarrow s\}$  of the term  $s'$  for the term  $s$  in  $t$  such that  $\|t\{s' \rightarrow s\}\|_J \neq \|t\|_J$ . We call  $SREL(t)$  the set of all relevant subterms of  $t$ .

Clearly,  $VREL(t) \subseteq SREL(t)$ . If we denote  $VAR(t)$  the set of all occurrences of variables in  $t$ , then  $VREL(t) = SREL(t) \cap VAR(t)$ .

*Example 8.* Consider the term  $t = [X|X]$  and the polynomial pre-interpretation implied by the list-length norm  $\|\cdot\|_l$ . Then,  $SREL(t) = \{X_{(2)}, t\}$ .  $\square$

The following proposition shows that monotone polynomial pre-interpretations characterize relevant subterms in a pure syntactic way.

**Proposition 2.** For any monotone polynomial pre-interpretation  $J$ , for any term  $t$ , the following property holds:

- (i)  $SREL(t) = \{t\}$  if  $t$  is a variable
- (ii)  $SREL(t) = \{t\} \cup (\cup_{j=1, \dots, m} SREL(t_{ij}))$ , if  $t = f(t_1, \dots, t_m)$  and  $P_t = P_f(P_{t_{f_1}}, \dots, P_{t_{f_m}})$  is the polynomial interpretation of  $t$  ( $t_{f_j}, 1 \leq j \leq m$ , are the selected subterms of  $t$  under  $J$ ).

*Proof.* The proof is similar to the proof in [4] except that it is extended to the case of polynomial pre-interpretations.  $\square$

**Proposition 3.** Let  $J$  be a monotone polynomial pre-interpretation. A term  $t$  is rigid w.r.t.  $J$  if and only if  $VREL(t) = \emptyset$ .

*Proof.* See appendix 2.

The major advantage of monotone polynomial pre-interpretations is that we can check the rigidity of a term  $t$  w.r.t. a given monotone polynomial pre-interpretation in a syntactic way: namely to verifying emptiness of  $VREL(t)$ . In the case of atoms, we can reduce verifying rigidity of an atom w.r.t. a monotone polynomial interpretation to verifying the rigidity of all its selected subterms w.r.t. the corresponding monotone polynomial pre-interpretation.

### 3.3 Applying Rigid Order Acceptability to Polynomial Interpretations

The idea of this part is to reduce the sufficient condition of rigid order acceptability in [7] to the case of polynomial interpretations. For the rest of this paper, we consider only the monotone polynomial interpretations. We need the following notion of polynomial interargument relations.

**Definition 20 (polynomial interargument relation).** *Let  $P$  be a definite program,  $p/n$  be a predicate in  $P$  and  $I$  be a polynomial interpretation for the language  $L$  underlying  $P$ . A polynomial interargument relation for  $p$  is a relation  $R_p \subseteq \{(t_1, \dots, t_n) \mid \varphi_p(P_{t_1}, \dots, P_{t_n})\}$ , where:*

- $\varphi_p(P_{t_1}, \dots, P_{t_n})$  is a formula in a disjunctive normal form
- each conjunct in  $\varphi_p$  is either  $P_{s_i} \geq_A P_{s_j}$ ,  $P_{s_i} >_A P_{s_j}$ ,  $P_{s_i} \leq \geq_A P_{s_j}$  or  $P_{s_i} \parallel_{\geq_A} P_{s_j}$ , where  $s_i, s_j$  are constructed from  $t_1, \dots, t_n$  by applying functors of  $P$ .

$R_p$  is a valid polynomial interargument relation for  $p/n$  w.r.t.  $I$  if and only if for every  $p(t_1, \dots, t_n) \in B_P^E$ :  $P \models p(t_1, \dots, t_n)$  implies  $(t_1, \dots, t_n) \in R_p$ .

Using the notions of rigidity and polynomial interargument relations w.r.t. a polynomial interpretation integrated with definition 8, theorem 2 and definition 14 we obtain:

**Proposition 4.** *Let  $S$  be a set of atomic queries,  $P$  be a definite program and  $I$  be a polynomial interpretation for the language  $L$  underlying  $P$ . For each predicate  $p$  in  $P$ , let  $R_p$  be a valid polynomial interargument relation for  $p$  w.r.t.  $I$ . If  $I$  is rigid on  $\text{Call}(P, S)$  such that*

- for any clause  $H \leftarrow B_1, \dots, B_n$ ,
  - for any atom  $B_i$  in its body, such that  $\text{rel}(B_i) \simeq \text{rel}(H)$ ,
  - for any substitution  $\theta$ , such that the arguments of the atoms in  $(B_1, \dots, B_{i-1})\theta$  all satisfy their associated polynomial interargument relations  $R_{\text{rel}(B_1)}, \dots, R_{\text{rel}(B_{i-1})}$ ,
- $$P_{H\theta} >_A P_{B_i\theta},$$

then  $P$  left-terminates w.r.t.  $S$ .

*Proof.* Let us consider the quasi-ordering  $\succeq_I$  on  $U_P^E$  and its proper extension  $\triangleright_I$  on  $U_P^E \cup B_P^E$  defined as in definition 14. Theorem 4 states that  $\succ_I$  and  $\triangleright_I$  are well founded orderings.

Since  $I$  is rigid on the call set  $\text{Call}(P, S)$ , the definition of the ordering  $\triangleright_I$  implies that for any atom  $q \in \text{Call}(P, S)$ , for any substitution  $\theta$ ,  $q \triangleleft \triangleright_I q\theta$ . Hence,  $\triangleright_I$  is rigid on  $\text{Call}(P, S)$ .

Moreover, for any predicate  $p$  in  $P$ , consider the set  $R_p^{\triangleright_I} = \{(t_1, \dots, t_n) \mid \varphi_p^{\triangleright_I}(t_1, \dots, t_n)\}$  where  $\varphi_p^{\triangleright_I}$  is defined as  $\varphi_p$ , except that each conjunct in  $\varphi_p^{\triangleright_I}$  is either  $s_i \triangleright_I s_j$ ,  $s_i \triangleright_I s_j$ ,  $s_i \triangleleft \triangleright_I s_j$  or  $s_i \parallel_{\triangleright_I} s_j$ . Intuitively,  $R_p^{\triangleright_I} = \{(t_1, \dots, t_n) \mid \varphi_p(P_{t_1}, \dots, P_{t_n})\}$ . Therefore, any valid interargument relation  $R_p$  w.r.t.  $I$  is also a valid interargument relation w.r.t.  $\triangleright_I$ .

Finally, the condition  $P_{H\theta} >_A P_{B_i\theta}$  implies  $H\theta \triangleright_I B_i\theta$ . All above conditions demonstrate  $P$  is rigid order-acceptability and left-terminates w.r.t.  $S$ .  $\square$

*Example 9.* Reconsider example 2. We are interested in proving termination of the program w.r.t. the query set  $S = \{d(t_1, t_2) \mid t_1 \text{ is a ground term and } t_2 \text{ is a free variable}\}$ . Observe, that  $Call(S, P)$  coincides with  $S$ .

Let  $I$  be a monotone polynomial interpretation that consists of a domain  $A = \mathbb{N} \setminus \{0, 1\}$ , an assignment that associates the function symbol  $der/1$  with the polynomial  $P_{der} := X^2$ , the function symbol  $+/2$  with the polynomial  $P_+ := X_1 + X_2$ , the function symbol  $*/2$  with the polynomial  $P_* := X_1 * X_2$ , the constant  $u$  with a constant  $c_u \in A$  and an assignment that associates the predicate symbol  $d/2$  with the polynomial  $P_d := X$ , where the variable  $X$  corresponds to the first argument position of  $d/2$ . Let  $R_d = \{(t_1, t_2) \mid t_1, t_2 \in Term_P \text{ and } P_{t_1} \geq_A P_{t_2}\}$  be a polynomial interargument relation w.r.t. the predicate  $d/2$ .

It is easy to verify that  $I$  is rigid on  $Call(S, P)$  and  $R_d$  is valid w.r.t.  $I$ . Then, the program terminates if the following hold:

$$\begin{aligned}
& |d(der(X + Y), DX + DY)\theta|_I >_A |d(der(X), DX)\theta|_I \\
& \quad d(der(X), DX)\theta \text{ satisfies } R_d \text{ implies} \\
& |d(der(X + Y), DX + DY)\theta|_I >_A |d(der(Y), DY)\theta|_I \\
& |d(der(X * Y), X * DY + Y * DX)\theta|_I >_A |d(der(X), DX)\theta|_I \\
& \quad d(der(X), DX)\theta \text{ satisfies } R_d \text{ implies} \\
& |d(der(X * Y), X * DY + Y * DX)\theta|_I >_A |d(der(Y), DY)\theta|_I \\
& |d(der(der(X)), DDY)\theta|_I >_A |d(der(X), DX)\theta|_I \\
& \quad d(der(X), DX)\theta \text{ satisfies } R_d \text{ implies} \\
& |d(der(der(X)), DDY)\theta|_I >_A |d(der(DX), DDY)\theta|_I
\end{aligned}$$

They are equivalent to the following inequalities on  $X, Y, DX \in Var_p$

$$\begin{array}{ll}
(X + Y)^2 >_A X^2 & X^2 >_A DX \Rightarrow (X * Y)^2 >_A Y^2 \\
X^2 >_A DX \Rightarrow (X + Y)^2 >_A Y^2 & X^4 >_A X^2 \\
(X * Y)^2 >_A X^2 & X^2 >_A DX \Rightarrow X^4 >_A DX^2
\end{array}$$

Since  $A = \mathbb{N} \setminus \{0, 1\}$ , the above inequalities are easily verified and the program left-terminates.  $\square$

## 4 Automation: the general idea

For automation of the approach, two sources of ideas and techniques are important:

- the generalisation of the constraint-based approach to termination analysis of [9] from linear norms and level mappings to monotonic polynomials.
- the integration of a number of useful results and heuristics from TRSs ([10, 13, 16, 22]).

The idea of the approach in [9] is to set up a symbolic form for all concepts involved in the termination conditions: in our case, the polynomial interpretation of each function and predicate symbol, the polynomial interargument relations and polynomial ordering conditions in proposition 4. Note that if we do not put a limit on the maximal degree of the polynomial, then there can be no finite general form of the polynomial associated to a term (there are infinitely many monomials  $a_i X^{i_1} X^{i_2} \dots X^{i_k}$  to consider). This is why we associate each function and predicate symbol with a *simple-mixed* polynomial, which is either a multivariate polynomial with all variables of at most degree 1 or a unary polynomial of at most degree 2.

From TRSs we borrow a sufficient condition for monotonicity of the polynomials:

**Proposition 5.** (see also [24]) *Let  $P = \sum_{i=1}^r a_i X_1^{k_{i,1}} X_2^{k_{i,2}} \dots X_m^{k_{i,m}}$  be a polynomial from  $A^m$  to  $A$  for which  $A = \mathbb{N} \setminus \{0\}$ ,  $m > 0$  and  $a_i \geq 0$  for all  $i = 1, \dots, r$ ,  $r > 0$ .  $P$  is monotone if  $\sum_{i=1}^r a_i k_{i,j} > 0$  for every  $j = 1, \dots, m$ .*

*Example 10.* Reconsider example 4. Let the first and the second argument positions of the predicate  $dist/2$  be, respectively, the input and output positions. For all other function symbols, let all arguments be the input arguments. Let  $I$  be a polynomial interpretation such that the constant  $x$  is associated with  $x_I \in A$ , the predicate symbol  $dist/2$  is associated with the polynomial  $P_{dist}(X) = f_0^d + f_1^d X + f_2^d X^2$ , the function symbol  $+/2$  is associated with the polynomial  $P_+(X, Y) = f_0^+ + f_1^+ X + f_2^+ Y + f_3^+ XY$  and the function symbol  $*/2$  is associated with the polynomial  $P_*(X, Y) = f_0^* + f_1^* X + f_2^* Y + f_3^* XY$ .  $I$  is monotone if  $f_1^d + f_2^d * 2 > 0$ ,  $f_1^+ + f_3^+ > 0$ ,  $f_2^+ + f_3^+ > 0$ ,  $f_1^* + f_3^* > 0$ ,  $f_2^* + f_3^* > 0$ .  $\square$

For the interargument relations, we only allow the linear interargument relations of [9]. The relations are of the form:

$$R_{p/n} = \{(t_1, \dots, t_n) \mid \sum_{i \in p_{inp}} p_i^e P_{t_i} \geq_A \sum_{j \in p_{out}} p_j^e P_{t_j} + p_0^e\},$$

with  $p_i^e \in \mathbb{N}$ ,  $i \in \{1, \dots, n\}$ ,  $p_{inp}$  and  $p_{out}$  respectively the sets of input and output argument positions of  $p/n$ .

But because they are applied to polynomial interpretations of terms, they still give rise to non-linear conditions in general.

*Example 10 (continued).* As an example, the condition for a valid interargument relation  $R_{dist}$  applied to clause 2 is of the following form:

$$(d_1^e X \geq_A d_2^e U + d_0^e) \wedge (d_1^e Y \geq_A d_2^e V + d_0^e) \Rightarrow d_1^e (f_0^+ + f_1^+ X + f_2^+ Y + f_3^+ XY) \geq_A d_2^e (f_0^+ + f_1^+ U + f_2^+ V + f_3^+ UV) + d_0^e. \quad \square$$

Next all other polynomial inequality conditions from proposition 4 are translated into constraints on the introduced symbols.

*Example 10(continued).* As an example, for recursive clause 2, the following constraints are imposed:

$$\begin{aligned}
& f_0^d + f_1^d (f_0^+ + f_1^+ X + f_2^+ Y + f_3^+ XY) \\
& + f_2^d (f_0^+ + f_1^+ X + f_2^+ Y + f_3^+ XY)^2 >_A f_0^d + f_1^d X + f_2^d X^2 \\
& d_1^e X \geq d_2^e U + d_0^e \Rightarrow f_0^d + f_1^d (f_0^+ + f_1^+ X + f_2^+ Y + f_3^+ XY) \\
& + f_2^d (f_0^+ + f_1^+ X + f_2^+ Y + f_3^+ XY)^2 >_A f_0^d + f_1^d Y + f_2^d Y^2
\end{aligned}$$

After normalisation, all the above constraints are transformed into the form:  $P(X_1, \dots, X_n) \geq_A 0 \Rightarrow Q(X_1, \dots, X_m) \geq_A 0$  or the form  $P(X_1, \dots, X_n) \geq_A 0$ . In [9] techniques are proposed to transform the constraints of the first type into constraints of the second type. These can be extended to polynomials.

Next we borrowed again techniques from TRS. The first technique is to move from  $A \subseteq \mathbb{N}$  to  $\mathbb{R}^+$ . Let  $a$  be  $\min\{c_I | c_I \in A\}$  is a polynomial interpretation of a constant  $c$ . Then instead of demanding that any of these constraints should hold (i.e.  $P(X_1, \dots, X_n) \geq 0$  for all  $X_1, \dots, X_n \in A$ ), it is sufficient to prove that  $P(X_1, \dots, X_n) \geq 0$  for all  $X_1, \dots, X_n \in A_R$ ,  $A_R = \mathbb{R}^+ \setminus [0, a)$ .

The following step is to transform all those constraints into *Diophantine inequalities* which contain only coefficients as variables. It can be done by applying repeatedly the following differentiation rules introduced in [5, 13, 15] in the context of TRSs:

$$\begin{array}{c}
\frac{P(\dots, X_i, \dots) > 0}{P(\dots, a, \dots) > 0, \frac{\partial P(\dots, X_i, \dots)}{\partial X_i} \geq 0} \\
\frac{P(\dots, X_i, \dots) \geq 0}{P(\dots, a, \dots) \geq 0, \frac{\partial P(\dots, X_i, \dots)}{\partial X_i} \geq 0}
\end{array}$$

Note the introduction of the inequations on the derivatives, which are actually extra constraints. Within TRS it has been argued that imposing these extra constraints is most often reasonable as it allows to eliminate all variables  $X_i$  and because, if a solution to the original problem exists, the solution space is usually large enough to also contain an element that respects the extra constraints.

Since the question for dealing with such general Diophantine inequalities is undecidable [16], an idea is to put an arbitrary bound on the values of variable coefficients (e.g.,  $[0, B]$ ). Then, the problem becomes solving a *finite domain constraint satisfaction problem* for a finite set of variables. Here finite-domain constraint solvers provide a variety of techniques to solve the remaining inequalities.

## 5 Conclusions

Since a few years ago, the LP termination analysis community and the TRS termination analysis community jointly organize the Workshop on Termination Analysis (WTA). These workshops have raised a considerable interest in gaining a better understanding of each others approaches. It soon became clear that there has to be a close relationship between the most popular technique in TRS, polynomial interpretations, and one of the key techniques in LP, acceptability

with linear norms and level mappings. However, partly because of the distinction between orderings over the natural numbers (LP) versus orderings over polynomials (TRS), the actual relation between the approaches was unclear.

One main conclusion of the research that led to this paper is that the distinction is a superficial one. Although termination conditions in LP are formulated in terms of mappings to natural numbers, the actual termination proofs do not reason on natural numbers. They are formulated in terms of linear inequalities. In fact, LP termination analysis systems never work on the basis of the norm and the level mapping; they work on the level of the *abstract norm* and *abstract level mapping* (see [23]). As such, one outcome of the work is that, indeed, the polynomial interpretations of TRS are a direct generalization of the current LP practice.

On the more technical level, the contribution of this paper is that we provide a complete theoretical framework for polynomial interpretations in LP termination analysis. Part of this builds strongly on the results in [7] on order acceptability, another part extends the results of Bossi et al. [4] on syntactic characterization of rigidity.

In the paper we only provide two examples of the class of programs for which the extension from linear to polynomial interpretations is important. Note that typical examples in LP termination analysis are often deliberately chosen to be linear, to remain in the scope of the designed techniques. Non-linear polynomial functions are present in many real word problems and programs encoding these problems are bound to require polynomial interpretations for their termination proofs.

It remains to be studied how we can benefit from the huge amount of work that people in TRS termination analysis have spent on automating proofs with polynomial interpretations and how integration of these techniques with the best approaches of LP termination analysis can lead to even more powerful techniques. We expect that this will lead to the development of a powerful new termination analyzer in the near future.

## 6 Acknowledgements

Manh Thang Nguyen is supported by GOA/2003/08.

## References

1. APT, K. R. Logic programming. In *Handbook of theoretical computer science (vol. B): formal models and semantics* (1990), MIT Press, pp. 493–574.
2. APT, K. R. *From logic programming to Prolog*. Prentice-Hall, Inc., 1996.
3. BAADER, F., AND NIPKOW, T. *Term rewriting and all that*. Cambridge University Press, 1998.
4. BOSSI, A., COCCO, N., AND FABRIS, M. Proving termination of logic programs by exploiting term properties. In *TAPSOFIT, Vol.2* (1991), pp. 153–180.
5. CONTEJEAN, E., MARCHÉ, C., TOMÁS, A. P., AND URBAIN, X. Mechanically proving termination using polynomial interpretations. Tech. Rep. 1382, 2004.

6. DE SCHREYE, D., AND DECORTE, S. Termination of logic programs: the never-ending story. *J. Log. Program.* 19-20 (1994), 199–260.
7. DE SCHREYE, D., AND SEREBRENIK, A. Acceptability with general orderings. In *Computational Logic: Logic Programming and Beyond*. Springer Verlag, 2002, pp. 187–210.
8. DE SCHREYE, D., AND VANDECASTEELE, H. Termination analysis of definite logic programs with respect to call patterns. Tech. Rep. Tech.Rep.CW 138, Department of Computer Science, K.U.Leuven, Belgium, January 1992.
9. DECORTE, S., DE SCHREYE, D., AND VANDECASTEELE, H. Constraint based automatic termination analysis of logic programs. *ACM Trans. Program. Lang. Syst.* 21, 6 (November 1999), 1137–1195.
10. DERSHOWITZ, N. Termination of rewriting. *J. Symb. Comput.* 3, 1-2 (1987), 69–116.
11. DERSHOWITZ, N. 33 examples of termination. *Lecture Notes in Computer Science* 909 (1995), 16–26.
12. FALASCHI, M., LEVI, G., MARTELLI, M., AND PALAMIDESSI, C. Declarative modeling of the operational behaviour of logic languages. *Theor. Comput. Sci.* 63, 3 (1989), 289–318.
13. GIESL, J. Generating polynomial orderings for termination proofs. In *RTA (1995)*, pp. 426–431.
14. JANSSEN, G., AND BRUYNOGHE, M. Deriving descriptions of possible values of program variables by means of abstract interpretation. *J. Log. Program.* 13, 2&3 (1992), 205–258.
15. LANKFORD, D. S. A finite termination algorithm. Tech. rep., Southwestern University, Georgetown, TX. Internal Memo.
16. LANKFORD, D. S. On proving term rewriting systems are noetherian. Tech. rep., Mathematics Department, Louisiana Tech. University, Ruston, LA. Memo MTP-3.
17. LLOYD, J. W. *Foundations of Logic Programming*. Springer Verlag, Berlin, 1987.
18. OHLEBUSCH, E. Implementing conditional term rewriting by graph rewriting. *Theor. Comput. Sci.* 262, 1 (2001), 311–331.
19. PLÜMER, L. *Termination proofs for logic programs*. Springer-Verlag New York, Inc., 1990.
20. SEREBRENIK, A. *Termination Analysis of Logic Programs*. PhD thesis, Department of Computer Science, K.U.Leuven, Belgium, 2003.
21. SEREBRENIK, A., AND DE SCHREYE, D. Termination analysis of logic programs using acceptability with general term orders. Tech. rep., Department of Computer Science, K.U.Leuven, Belgium, May 2000.
22. STEINBACH, J. Generating polynomial orderings. *Inf. Process. Lett.* 49, 2 (1994), 85–93.
23. VERSCHAETSE, K., AND DE SCHREYE, D. Deriving termination proofs for logic programs, using abstract procedures. In *Proceedings 8th ICLP (1991)*, pp. 301–315.
24. ZANTEMA, H. Termination of term rewriting. Tech. Rep. UU-CS (Ext.r.no.2000-04), Institute of Information and Computing Sciences, Utrecht University, Utrecht, The Netherlands, 2000.

## Appendix 1: Example (Der)

$$d(\text{der}(u), 1).$$

$$d(\text{der}(A), 0) : -\text{number}(A).$$

$$d(\text{der}(X + Y), DX + DY) : -d(\text{der}(X), DX), d(\text{der}(Y), DY). \quad (5)$$

$$d(\text{der}(X * Y), X * DY + Y * DX) : -d(\text{der}(X), DX), d(\text{der}(Y), DY). \quad (6)$$

$$d(\text{der}(\text{der}(X)), DD X) : -d(\text{der}(X), DX), d(\text{der}(DX), DD X). \quad (7)$$

We want to prove that no (semi) linear norm and level mapping are sufficient for proving termination of this program w.r.t. the following set of queries:

$$S = \{d(t_1, t_2) \mid t_1 \text{ is a ground term and } t_2 \text{ is a free variable}\}.$$

Using type analysis, we can infer that  $\text{Call}(\text{Der}, S) = S$ . We will consider the first argument of  $d/2$  as an input argument and the second as an output.

By contradiction, we assume that such a norm and level mapping exist. By linearity, the norm  $\|\cdot\|$  takes the form:

- $\|t\| = 0$  if  $t$  is a constant or a free variable,
- $\|t_1 + t_2\| = f_0^+ + f_1^+ \|t_1\| + f_2^+ \|t_2\|$ ,
- $\|t_1 * t_2\| = f_0^* + f_1^* \|t_1\| + f_2^* \|t_2\|$ ,
- $\|\text{der}(t)\| = f_0^d + f_1^d \|t\|$ .

And the following general level mapping  $|\cdot|$  is defined:

- $|d(t_1, t_2)| = d_0 + d_1 \|t_1\| + d_2 \|t_2\| = d_0 + d_1 \|t_1\|$ , because  $t_2$  is a free variable ( $\|t_2\| = 0$ ).
- $|\text{number}(t)| = n_0 + n_1 \|t\|$ .

where  $f_0^+, f_1^+, f_2^+, f_0^*, f_1^*, f_2^*, f_0^d, f_1^d, d_0, d_1, m_0$  and  $m_1$  are non-negative integers.

We also allow interargument relations to take the form of *linear inequality relation*, using the input-output information mentioned above, as follows([9]):

$$R_{d/2} = \{(a_1, a_2) \mid d_1^e a_1 \geq d_2^e a_2 + d_0^e\}$$

with  $d_i^e \in \mathbb{N}, i \in \{0, 1, 2\}$ ,  $a_1, a_2$  correspond to the sizes of the first and the second arguments respectively.

Hence, proving termination of the example is equivalent to proving that the following inequalities hold:

1. For any substitution  $\theta$ ,  $|d(\text{der}(X + Y), DX + DY)\theta| > |d(\text{der}(X), DX)\theta|$  or for any term  $t_1, t_2$ ,  $d_0 + d_1(f_0^d + f_1^d(f_0^+ + f_1^+ \|t_1\| + f_2^+ \|t_2\|)) > d_0 + d_1(f_0^d + f_1^d \|t_1\|)$  (for the first inequality in (5)). We rewrite the inequality into a normal form:  $d_1 f_1^d f_0^+ + d_1 f_1^d (f_1^+ - 1) \|t_1\| + d_1 f_1^d f_2^+ \|t_2\| > 0$ . Because  $\|t_1\|, \|t_2\|$  range over all natural numbers, we conclude that  $d_1 \geq 1, f_1^d \geq 1$  and  $f_1^+ \geq 1$ .

2.  $P \models d(\text{der}(X), DX)\theta$  implies  $\|d(\text{der}(X+Y), DX+DY)\theta\| > \|d(\text{der}(Y), DY)\theta\|$  or  $d_1^e(f_0^d + f_1^d \|t_1\|) \geq d_2^e \|dt_1\| + d_0^e \Rightarrow d_0 + d_1(f_0^d + f_1^d(f_0^+ + f_1^+ \|t_1\| + f_2^+ \|t_2\|)) > d_0 + d_1(f_0^d + f_1^d \|t_2\|)$  (for the second inequality in (5)). We rewrite it into a normal form:  $(d_1^e f_0^d - d_0^e) + d_1^e f_1^d \|t_1\| - d_2^e \|dt_1\| \geq 0 \Rightarrow d_1 f_1^d f_0^+ + d_1 f_1^d f_1^+ \|t_1\| + d_1 f_1^d (f_2^+ - 1) \|t_2\| > 0$ . Because the inequality is strict and  $\|t_1\|, \|t_2\|$  range over all natural numbers, the following holds:  $(d_1^e f_0^d - d_0^e) + d_1^e f_1^d \|t_1\| - d_2^e \|dt_1\| \geq 0 \Rightarrow d_1 \geq 1 \wedge f_1^d \geq 1 \wedge f_2^+ \geq 1$ .
3. For any substitution  $\theta$ ,  $|d(\text{der}(X*Y), X*DY+Y*DX)\theta| > |d(\text{der}(X), DX)\theta|$  or for any term  $t_1, t_2$ ,  $d_0 + d_1(f_0^d + f_1^d(f_0^* + f_1^* \|t_1\| + f_2^* \|t_2\|)) > d_0 + d_1(f_0^d + f_1^d \|t_1\|)$  (for the first inequality in (6)). We rewrite it into a normal form:  $d_1 f_1^d f_0^* + d_1 f_1^d (f_1^* - 1) \|t_1\| + d_1 f_1^d f_2^* \|t_2\| > 0$  Because  $\|t_1\|, \|t_2\|$  range over all natural numbers, we conclude that  $f_1^* \geq 1, d_1 \geq 1$  and  $f_1^d \geq 1$ .
4.  $P \models d(\text{der}(X), DX)\theta$  implies  $\|d(\text{der}(X*Y), X*DY+Y*DX)\theta\| > \|d(\text{der}(Y), DY)\theta\|$  or  $d_1^e(f_0^d + f_1^d \|t_1\|) \geq d_2^e \|dt_1\| + d_0^e \Rightarrow d_0 + d_1(f_0^d + f_1^d(f_0^* + f_1^* \|t_1\| + f_2^* \|t_2\|)) > d_0 + d_1(f_0^d + f_1^d \|t_2\|)$  (for the second inequality in (6)). We rewrite it into a normal form:  $(d_1^e f_0^d - d_0^e) + d_1^e f_1^d \|t_1\| - d_2^e \|dt_1\| \geq 0 \Rightarrow d_1 f_1^d f_0^* + d_1 f_1^d f_1^* \|t_1\| + d_1 f_1^d (f_2^* - 1) \|t_2\| > 0$ . Because  $\|t_1\|, \|t_2\|$  range over all natural numbers, the following holds:  $(d_1^e f_0^d - d_0^e) + d_1^e f_1^d \|t_1\| - d_2^e \|dt_1\| \geq 0 \Rightarrow d_1 \geq 1 \wedge f_1^d \geq 1 \wedge f_2^* \geq 1$ .
5. For any substitution  $\theta$ ,  $|d(\text{der}(\text{der}(X)), DD\text{der}(X))\theta| > |d(\text{der}(X), DX)\theta|$  or for any term  $t$ ,  $d_0 + d_1(f_0^d + f_1^d(f_0^d + f_1^d \|t\|)) > d_0 + d_1(f_0^d + f_1^d \|t\|)$  (for the first inequality in (7)). We rewrite it into a normal form:  $d_1 f_1^d f_0^d + d_1 f_1^d (f_1^d - 1) \|t\| > 0$ . Because  $\|t\|$  ranges over all natural numbers, it implies:  $d_1 \geq 1, f_1^d \geq 1$ .
6.  $P \models d(\text{der}(X), DX)\theta$  implies  $\|d(\text{der}(\text{der}(X)), DD\text{der}(X))\theta\| > \|d(\text{der}(DX), DD\text{der}(X))\theta\|$  or  $d_1^e(f_0^d + f_1^d \|t\|) \geq d_2^e \|dt\| + d_0^e \Rightarrow d_0 + d_1(f_0^d + f_1^d(f_0^d + f_1^d \|t\|)) > d_0 + d_1(f_0^d + f_1^d \|dt\|)$  (for the second inequality in (7)). We rewrite it into a normal form:  $(d_1^e f_0^d - d_0^e) + d_1^e f_1^d \|t\| - d_2^e \|dt\| \geq 0 \Rightarrow d_1 f_1^d f_0^d + d_1 f_1^d f_1^d \|t\| - d_1 f_1^d \|dt\| > 0$ . It is equivalent to:  $(d_1^e f_0^d - d_0^e) + d_1^e f_1^d \|t\| - d_2^e \|dt\| \geq 0 \Rightarrow d_1 \geq 1 \wedge f_1^d \geq 1 \wedge f_0^d + f_1^d \|t\| - \|dt\| > 0$ .
- 7a.  $d_1^e(f_0^d + f_1^d \|t_1\|) \geq d_0^e + d_2^e \|dt_1\| \wedge d_1^e(f_0^d + f_1^d \|t_2\|) \geq d_0^e + d_2^e \|dt_2\| \Rightarrow d_1^e(f_0^d + f_1^d(f_0^+ + f_1^+ \|t_1\| + f_2^+ \|t_2\|)) \geq d_0^e + d_2^e(f_0^+ + f_1^+ \|t_1\| + f_2^+ \|dt_2\|)$  (for valid interargument relation in (5)) or  $(d_1^e f_0^d - d_0^e) + d_1^e f_1^d \|t_1\| - d_2^e \|dt_1\| \geq 0 \wedge (d_1^e f_0^d - d_0^e) + d_1^e f_1^d \|t_2\| - d_2^e \|dt_2\| \geq 0 \Rightarrow (d_1^e f_0^d + d_1^e f_1^d f_0^+ - d_0^e - d_2^e f_0^+) + (d_1^e f_1^d f_1^+ \|t_1\| + (d_1^e f_1^d f_2^+) \|t_2\| - d_2^e f_1^+ \|dt_1\| - d_2^e f_2^+ \|dt_2\|) \geq 0$ .
- 7b.  $d_1^e(f_0^d + f_1^d \|t_1\|) \geq d_0^e + d_2^e \|dt_1\| \wedge d_1^e(f_0^d + f_1^d \|t_2\|) \geq d_0^e + d_2^e \|dt_2\| \Rightarrow d_1^e(f_0^d + f_1^d(f_0^* + f_1^* \|t_1\| + f_2^* \|t_2\|)) \geq d_0^e + d_2^e(f_0^* + f_1^* \|t_1\| + f_2^* \|dt_2\|) + f_2^* (f_0^* + f_1^* \|t_2\| + f_2^* \|dt_1\|)$  (for valid interargument relation in (6)) or  $(d_1^e f_0^d - d_0^e) + d_1^e f_1^d \|t_1\| - d_2^e \|dt_1\| \geq 0 \wedge (d_1^e f_0^d - d_0^e) + d_1^e f_1^d \|t_2\| - d_2^e \|dt_2\| \geq 0 \Rightarrow (d_1^e f_0^d + d_1^e f_1^d f_0^* - d_0^e - d_2^e f_0^+ - d_2^e f_1^+ f_0^* - d_2^e f_2^+ f_0^*) + (d_1^e f_1^d f_1^* - d_2^e f_1^+ f_1^*) \|t_1\| + (d_1^e f_1^d f_2^* - d_2^e f_2^+ f_1^*) \|t_2\| - d_2^e f_1^+ f_2^* \|dt_1\| - d_2^e f_1^+ f_2^* \|dt_2\| \geq 0$ .
- 7c.  $d_1^e(f_0^d + f_1^d \|t\|) \geq d_0^e + d_2^e \|dt\| \wedge d_1^e(f_0^d + f_1^d \|dt\|) \geq d_0^e + d_2^e \|ddt\| \Rightarrow d_1^e(f_0^d + f_1^d(f_0^d + f_1^d \|t\|)) \geq d_0^e + d_2^e \|ddt\|$  (for valid interargument relation in (7)) or  $(d_1^e f_0^d - d_0^e) + d_1^e f_1^d \|t\| - d_2^e \|dt\| \geq 0 \wedge (d_1^e f_0^d - d_0^e) + d_1^e f_1^d \|dt\| - d_2^e \|ddt\| \geq 0 \Rightarrow (d_1^e f_0^d + d_1^e f_1^d f_0^d - d_0^e) + d_1^e f_1^d f_1^d \|t\| - d_2^e \|ddt\| \geq 0$ .

Because the valid interargument relation condition is satisfied, the constraints [1, 2, 3, 4, 5] we derive that  $d_1 \geq 1, f_1^d \geq 1, f_1^+ \geq 1, f_2^+ \geq 1, f_1^* \geq 1, f_2^* \geq 1$ .

By subtracting the inequalities in the left-hand sides from the right-hand sides of [7a, 7b, 7c] and noting that those constraints only hold if the subtractions yield inequalities of the form  $\sum_{i=1}^n p_i \|x_i\| + p_0 \geq 0$  in their left-hand sides, the followings hold:

$$\begin{aligned}
7a'. & (d_1^e f_0^d + d_1^e f_1^d f_0^+ - d_0^e - d_2^e f_0^+ - f_1^+ d_1^e f_0^d + f_1^+ d_0^e - f_2^+ d_1^e f_0^d + f_2^+ d_0^e) + (d_1^e f_1^d f_1^+ - d_1^e f_1^+ f_1^d) \|t_1\| + (d_1^e f_1^d f_2^+ - d_1^e f_2^+ f_1^d - f_2^+ f_1^d d_1^e) \|t_2\| \geq 0. \\
7b'. & (d_1^e f_0^d + d_1^e f_1^d f_0^* - d_0^e - d_2^e f_0^+ - d_2^e f_1^+ f_0^* - d_2^e f_2^+ f_0^* - f_2^+ f_2^* d_1^e f_0^d + f_2^+ f_2^* d_0^e - f_1^+ f_2^* d_1^e f_0^d + f_1^+ f_2^* d_0^e) + (d_1^e f_1^d f_1^* - d_2^e f_1^+ f_1^* - f_2^+ f_2^* d_1^e f_1^d) \|t_1\| + (d_1^e f_1^d f_2^+ - d_2^e f_2^+ f_1^* - f_1^+ f_2^* d_1^e f_1^d) \|t_2\| \geq 0. \\
7c'. & (d_1^e f_0^d + d_1^e f_1^d f_0^d - d_0^e - d_1^e f_0^d + d_0^e - f_1^d d_1^e f_0^d + f_1^d d_0^e) \geq 0.
\end{aligned}$$

[7c'] is trivial. Since  $\|t_1\|, \|t_2\|$  range over all natural numbers, they [7b'] reduce to the followings:

$$d_1^e f_1^d f_1^* - d_2^e f_1^+ f_1^* - f_2^+ f_2^* d_1^e f_1^d \geq 0 \quad (8)$$

$$d_1^e f_1^d f_2^+ - d_2^e f_2^+ f_1^* - f_1^+ f_2^* d_1^e f_1^d \geq 0 \quad (9)$$

Inequality (9) yield  $d_1^e f_1^d f_2^+ (1 - f_1^+) - d_2^e f_2^+ f_1^* \geq 0$ . Because  $f_1^+ \geq 1$ , there are two possible cases:

- $d_1^e = 0, d_2^e = 0$ . Replace these equalities to [6] we derive  $-d_0^e \geq 0 \Rightarrow d_1 \geq 1 \wedge f_1^d \geq 1 \wedge f_0^d + f_1^d * \|t\| - \|dt\| > 0$ . It implies the inequality  $-1 < 0$  (contradiction).
- $d_1^e > 0, d_2^e = 0, f_1^+ = 1$ . Replace these constraints to [6], we derive  $(d_1^e f_0^d - d_0^e) + d_1^e f_1^d \|t\| \geq 0 \Rightarrow d_1 \geq 1 \wedge f_1^d \geq 1 \wedge f_0^d + f_1^d * \|t\| - \|dt\| > 0$ . It implies  $-1 < 0$  (contradiction).

Therefore, we can not find such (semi) linear norm and level mapping that are sufficient for proving termination of the program.  $\square$

## Appendix 2: Proof of Proposition 3

**Proposition 3** *Let  $J$  be a monotone polynomial pre-interpretation. A term  $t$  is rigid w.r.t.  $J$  if and only if  $VREL(t) = \emptyset$ .*

*Proof.* We consider the following necessary and sufficient condition:

- (i). If  $VREL(t) = \emptyset$  then  $t$  is rigid w.r.t.  $J$ . It is implied trivially from the definition of rigidity w.r.t. a polynomial pre-interpretation.
- (ii). If  $t$  is rigid w.r.t.  $J$  then  $VREL(t) = \emptyset$ . The proof is by contradiction. Let  $X_{(i)}$  be a relevant occurrence of a variable  $X$  of the term  $t$  w.r.t.  $J$  and  $s$  be a term such that  $P_t$  is not associated equivalent to  $P_{t\{s \rightarrow X_{(i)}\}}$ . Then, there exists an assignment  $\alpha$  to each variable in  $Var_P$  of a value in  $A$  such that  $P_t \alpha \neq P_{t\{s \rightarrow X_{(i)}\}} \alpha$  where  $P_t \alpha$  and  $P_{t\{s \rightarrow X_{(i)}\}} \alpha$  are instantiations of  $P_t$  and  $P_{t\{s \rightarrow X_{(i)}\}}$  w.r.t.  $\alpha$ . Let  $\theta = \{X/s\}$  be a substitution. We distinguish between the following cases:

1.  $P_{X_{(i)}}\alpha < P_s\alpha$ . We show by induction on the depth of the term  $t$ ,  $depth(t)$ , that  $P_t\alpha < P_{t\theta}\alpha$ .

For  $t$  such that  $depth(t) = 0$ . Since  $VREL(t) \neq \emptyset$ , then  $t$  is a variable. Obviously,  $P_t\alpha < P_{t\theta}\alpha$ .

Suppose for any term  $t'$ ,  $0 \leq depth(t') \leq n, n \in \mathbb{N}$ , such that there exists a relevant occurrence  $X_{(i)}$  of the variable  $X$  in  $t'$ , we have  $P_{t'}\alpha < P_{t'\theta}\alpha$ . We need to prove that for any term  $t \in Term_P$ ,  $depth(t) = n + 1$ , if there exists a relevant occurrence  $X_{(i)}$  of the variable  $X$  in  $t$ , then  $P_t\alpha < P_{t\theta}\alpha$ .

Because any relevant occurrence of the variable  $X$  is a relevant subterm of  $t$ , from proposition 2 it must be a relevant variable occurrence of one of the selected subterms of  $t$ . Let  $t$  be  $f(t_1, \dots, t_n)$ , and  $P_t = P_f(P_{t_1}, \dots, P_{t_n})$  be the polynomial interpretation of  $t$ . Let  $t_{i_1}, \dots, t_{i_k}, 0 < k \leq n$ , be the selected subterms of  $t$  containing relevant occurrences of the variable  $X$ . The induction hypothesis implies  $P_{t_{i_1}}\alpha < P_{t_{i_1}\theta}\alpha, \dots, P_{t_{i_k}}\alpha < P_{t_{i_k}\theta}\alpha$ . Because  $P_f(X_1, \dots, X_m)$  is a monotone polynomial, then

$$\begin{aligned}
P_t\alpha &= P_f(P_{t_1}\alpha, \dots, P_{t_n}\alpha) \\
&< P_f(P_{t_1}\alpha, \dots, P_{t_{i_1-1}}\alpha, P_{t_{i_1}\theta}\alpha, P_{t_{i_1+1}}\alpha, \dots, P_{t_n}\alpha) \dots \\
&< P_f(P_{t_1}\alpha, \dots, P_{t_{i_1}\theta}\alpha, \dots, P_{t_{i_k-1}}\alpha, P_{t_{i_k}\theta}\alpha, P_{t_{i_k+1}}\alpha, \dots, P_{t_n}\alpha) \\
&= P_f(P_{t_1}\theta\alpha, \dots, P_{t_n}\theta\alpha) \\
&= P_{t\theta}\alpha. \\
\Rightarrow P_t\alpha &\neq P_{t\theta}\alpha.
\end{aligned}$$

2.  $P_{X_{(i)}}\alpha > P_s\alpha$ . The proof is similar to the previous case.

Therefore,  $P_t$  is not associated equivalent to  $P_{t\theta}$  and  $t$  is not rigid w.r.t.  $J$ , providing a contradiction to the hypothesis that  $t$  is rigid w.r.t.  $J$ . Hence,  $VREL(t) = \emptyset$   $\square$