

**Non-transformational termination  
analysis of Logic Programs, based on  
general term-ordering**

*Alexander Serebrenik*

*Danny De Schreye*

*Report CW 284, January 2000*



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

# Non-transformational termination analysis of Logic Programs, based on general term-ordering

*Alexander Serebrenik*

*Danny De Schreye*

*Report CW 284, January 2000*

Department of Computer Science, K.U.Leuven

## **Abstract**

We present a new approach to termination analysis of logic programs. The essence of the approach is that we make use of general term-orderings (instead of level mappings), like it is done in transformational approaches to logic program termination analysis, but that we apply these orderings directly to the logic program and not to the term-rewrite system obtained through some transformation. We define a variant of acceptability, based on general term-orderings, and show how it is equivalent to LD-termination for well-moded, simply moded programs. We develop a demand driven, constraint-based approach to verify this acceptability-variant.

The advantage of the approach over standard acceptability is that in some cases, where complex level mappings are needed, fairly simple term-orderings may be easily generated. The advantage over transformational approaches is that it avoids the transformation step all together.

**Keywords :** termination analysis, acceptability, term-orderings.

# Non-transformational termination analysis of Logic Programs, based on general term-orderings

Alexander Serebrenik, Danny De Schreye

Department of Computer Science, K.U. Leuven  
Celestijnenlaan 200A, B-3001, Heverlee, Belgium  
Email: {Alexander.Serebrenik, Danny.DeSchreye}@cs.kuleuven.ac.be

Technical Report CW 284

**Abstract.** We present a new approach to termination analysis of logic programs. The essence of the approach is that we make use of general term-orderings (instead of level mappings), like it is done in transformational approaches to logic program termination analysis, but that we apply these orderings directly to the logic program and not to the term-rewrite system obtained through some transformation. We define a variant of acceptability, based on general term-orderings, and show how it is equivalent to LD-termination for well-moded, simply moded programs. We develop a demand driven, constraint-based approach to verify this acceptability-variant.

The advantage of the approach over standard acceptability is that in some cases, where complex level mappings are needed, fairly simple term-orderings may be easily generated. The advantage over transformational approaches is that it avoids the transformation step all together.

**Keywords:** termination analysis, acceptability, term-orderings.

## 1 Introduction

There are many different approaches to termination analysis of logic programs. One particular distinction is between *transformational* approaches and “*direct*” ones. A transformational approach first transforms the logic program into an “equivalent” term-rewrite system (or, in some cases, into an equivalent functional program). Here, equivalence means that, at the very least, the termination of the term-rewrite system should imply the termination of the logic program, for some predefined collection of queries<sup>1</sup>. Direct approaches do not include such a transformation, but prove the termination directly on the basis of the logic program.

Besides the transformation step itself, there is one other technical difference between these approaches. Direct approaches usually prove termination on the basis of a well-founded ordering over the natural numbers. More specifically,

---

<sup>1</sup> The approach of Arts [4] is exceptional in the sense that the termination of the logic program is concluded from a weaker property of *single-redex normalisation* of the term-rewrite system.

they use a *level mapping*, which maps atoms to natural numbers, and, they verify appropriate decreases of this level mapping on the atoms occurring in the clauses. On the other hand, transformational approaches make use of more general well-founded orderings over terms, such as reduction orders, or more specifically simplification order, or others (see [11]). Although transformational approaches have been quite successful (see [16] for an overview), they have one disadvantage: Due to the transformation, they provide little understanding of the termination proof in terms of the original program. This is not a problem if the goal is only to achieve (automatic) verification. However, in a context of understanding and reasoning about the termination behaviour of the logic program, the direct approach may be better suited.

In this paper, we present an initial study on how to merge the two approaches. We investigate the use of well-founded term-orderings as a means of directly proving the termination of logic programs—without intermediate transformation. Our motivation for doing this is both:

- to improve our understanding of the relation between these approaches, and
- to evaluate to what extent the use of the general term orderings (instead of level mappings) either improves or deteriorates the direct approaches.

To illustrate the latter point, consider the following program, that formulates some of the rules for computing the repeated derivative of a linear function in one variable  $t$  (see also [13]) :

$$\begin{aligned}
& d(\text{der}(t), 1). \\
& d(\text{der}(A), 0) \leftarrow \text{number}(A). \\
& d(\text{der}(X + Y), DX + DY) \leftarrow d(\text{der}(X), DX), d(\text{der}(Y), DY). \\
& d(\text{der}(X * Y), X * DY + Y * DX) \leftarrow d(\text{der}(X), DX), d(\text{der}(Y), DY). \\
& d(\text{der}(\text{der}(X)), DD X) \leftarrow d(\text{der}(X), DX), d(\text{der}(DX), DD X).
\end{aligned}$$

Proving termination of this program on the basis of a level-mapping is hard. For this example, the required level-mapping is a non-linear function. In particular, a level mapping

$$\begin{aligned}
|d(X, Y)| &= \|X\| \\
|\text{number}(X)| &= 0 \\
\|\text{der}(X)\| &= 2^{\|X\|} \\
\|X + Y\| &= \max(\|X\|, \|Y\|) + 1 \\
\|X * Y\| &= \max(\|X\|, \|Y\|) + 1 \\
\|t\| &= 2 \\
\|n\| &= 2, \text{ if } n \text{ is a number}
\end{aligned}$$

would be needed. No automatic system for proving termination on the basis of level mappings is able to generate such mappings. Moreover, we believe, that it would be very difficult to extend existing systems to support generation of appropriate non-linear mappings.

Although we have not yet presented our general-well-founded term ordering approach, it should be intuitively clear, that we can capture the decrease in order between the  $der(X)$  and  $DX$  by using an ordering on terms that gives the highest “priority” to the functor  $der$ .

As an example of the fact that moving to general ordering can also introduce deterioration, consider the following program from [7, 10].

$$\begin{aligned} \text{conf}(X) &\leftarrow \text{delete}_2(X, Z), \text{delete}(U, Y, Z), \text{conf}(Y). \\ \text{delete}_2(X, Y) &\leftarrow \text{delete}(U, X, Z), \text{delete}(V, Z, Y). \\ \text{delete}(X, [X|T], T). \\ \text{delete}(X, [H|T], [H|T1]) &\leftarrow \text{delete}(X, T, T1). \end{aligned}$$

Note that by reasoning in terms of sizes of terms, we can infer that the size decreases by 2 after the call to  $\text{delete}_2$  predicate in the first clause and then increases by 1 in the subsequent call to the  $\text{delete}$  predicate. In total, sizes allow to conclude a decrease. Reasoning in terms of order relations only, however, does not allow to conclude the overall decrease from the inequalities  $\text{arg}3 < \text{arg}2$  for the  $\text{delete}$  predicate and  $\text{arg}1 > \text{arg}2$  for the  $\text{delete}_2$  predicate.

In the remainder of this paper, we will start off from a variant of the notion of *acceptability*, as introduced by Apt and Pedreschi [3], obtained by replacing level mappings by term orderings. We show how this variant of acceptability remains equivalent to termination under the left-to-right selection rule, for certain goals. Then, we illustrate how this result can be used to prove termination with some examples. Next, we discuss automation of the approach. We elaborate on a demand-driven method to set-up and verify sufficient preconditions for termination. In this method, the aim is to derive—in, as much as possible, a constructive way—a well-founded ordering over the set of all atoms and terms of the language underlying the program, that satisfies the termination condition.

## 2 Preliminaries

### 2.1 Term ordering

An *order* over a set  $S$  is an irreflexive, asymmetric and transitive relation  $>$  defined on elements of  $S$ . As usual,  $s \geq t$  denotes that either  $s > t$  or  $s = t$ , and  $s \parallel t$  denotes that  $s$  and  $t$  are incomparable. A set  $S$  is said to be *well founded* if there are no infinite descending sequences  $s_1 > s_2 > \dots$  of elements of  $S$ . If the set  $S$  is clear from the context we will say that the order, defined on it, is well-founded.

The study of termination of term-rewriting systems caused intensive study of term orderings. A number of useful properties of term orderings were established.

**Definition 1.** *Let  $>$  be an ordering on terms.*

- *If  $s_1 > s_2$  implies  $f(\bar{t}_1, s_1, \bar{t}_2) > f(\bar{t}_1, s_2, \bar{t}_2)$  and  $p(\bar{t}_1, s_1, \bar{t}_2) > p(\bar{t}_1, s_2, \bar{t}_2)$  for any sequences of terms  $\bar{t}_1$  and  $\bar{t}_2$ , function symbol  $f$  and predicate  $p$ , then  $>$  is called monotone.*

- If for any term  $f(\bar{t}_1, s, \bar{t}_2)$  holds that  $f(\bar{t}_1, s, \bar{t}_2) > s$ , then  $>$  is said to have the subterm property.

The following are examples of order relations:  $>$  on the set of numbers, lexicographic order on the set of strings (this is a way the entries are ordered in dictionaries), multiset ordering and recursive path ordering [11].

**Definition 2.** [11] For a partially ordered set  $(S, \succ)$  the multiset ordering on  $\mathcal{M}(S)$ , a set of all finite multisets of elements of  $S$ , is defined as follows:

$$M \gg M'$$

if, and only if, for some multisets  $X, Y \in \mathcal{M}(S)$ , where  $X$  is a non-empty subset of  $M$ ,

$$M' = (M - X) \cup Y$$

and for all  $y \in Y$  there is and  $x \in X$  such that

$$x \succ y$$

**Definition 3.** [11] Let  $\succ$  be a partial ordering on a set of operators  $F$ . The recursive path ordering (rpo) on the set of terms over  $F$  is defined recursively as follows:  $s = f(s_1, \dots, s_m) \succ_{rpo} g(t_1, \dots, t_n) = t$  if one of the following holds

1.  $s_i \succeq_{rpo} t$  for some  $i = 1, \dots, m$
2.  $f \succ g$  and  $s \succ_{rpo} t_j$  for all  $j = 1, \dots, n$
3.  $f = g$  and  $\{s_1, \dots, s_m\} \gg_{rpo} \{t_1, \dots, t_n\}$ ,

where  $\gg_{rpo}$  is the extension of  $\succ_{rpo}$  to the multisets and  $\succeq_{rpo}$  means  $\succ_{rpo}$  or equivalent up to permutations of subterms.

Note, that the set of terms over  $F$ , mentioned in the definition above, in the context of the term-rewriting systems are ground. We are going to solve this problem by restricting our attention only to well-moded programs and taking in account only the input positions.

## 2.2 Logic Programs

We follow the standard notation for terms and atoms.  $Term_P$  and  $Atom_P$  denote, respectively, sets of all terms and atoms that can be constructed from the language underlying the program  $P$ . The extended Herbrand Universe  $U_P^E$  (the extended Herbrand base  $B_P^E$ ) is a quotient set of  $Term_P$  ( $Atom_P$ ) modulo the variant relation.

For a predicate  $p/n$  a mode is an atom  $p(m_1, \dots, m_n)$ , where  $m_i \in \{i, o\}$  for  $1 \leq i \leq n$ . Positions with  $i$  are called *input positions*, and positions with  $o$  are called *output positions* of  $p$ . We assume that a fixed mode is associated with each predicate in a program. To simplify the notation, an atom written as  $p(\mathbf{s}, \mathbf{t})$  means:  $\mathbf{s}$  is the vector of terms filling the input positions, and  $\mathbf{t}$  is the vector of terms filling the output positions.

A *query* is a finite sequence of atoms.  $\text{Var}(A)$  denotes a set of the variables appearing in  $A$ . Superscripts are used to denote the restriction of the atom only to its input (output) positions, i.e.,  $p(\mathbf{s}, \mathbf{t})^{inP}$  denotes  $p(\mathbf{s})$ .

We refer to an SLD-tree constructed using the left-to-right selection rule of Prolog, as an LD-tree. We will say that a goal  $G$  *LD-terminates* for a program  $P$ , if the LD-tree for  $(P, G)$  is finite.

**Definition 4.** [1]

1. A query  $p_1(\mathbf{s}_1, \mathbf{t}_1), \dots, p_n(\mathbf{s}_n, \mathbf{t}_n)$  is called *well moded* if for  $i \in [1, n]$

$$\text{Var}(\mathbf{s}_i) \subseteq \bigcup_{j=1}^{i-1} \text{Var}(\mathbf{t}_j)$$

2. A clause  $p_0(\mathbf{t}_0, \mathbf{s}_{n+1}) \leftarrow p_1(\mathbf{s}_1, \mathbf{t}_1), \dots, p_n(\mathbf{s}_n, \mathbf{t}_n)$  is called *well moded* if for  $i \in [1, n+1]$

$$\text{Var}(\mathbf{s}_i) \subseteq \bigcup_{j=0}^{i-1} \text{Var}(\mathbf{t}_j)$$

3. A program is called *well moded* if every clause of it is.

**Definition 5.** [2]

1. A query  $p_1(\mathbf{s}_1, \mathbf{t}_1), \dots, p_n(\mathbf{s}_n, \mathbf{t}_n)$  is called *simply moded* if  $\mathbf{t}_1, \dots, \mathbf{t}_n$  is a linear family of variables and for  $i \in [1, n]$

$$\text{Var}(\mathbf{s}_i) \cap \left( \bigcup_{j=i}^n \text{Var}(\mathbf{t}_j) \right) = \emptyset$$

2. A clause  $p_0(\mathbf{s}_0, \mathbf{t}_0) \leftarrow p_1(\mathbf{s}_1, \mathbf{t}_1), \dots, p_n(\mathbf{s}_n, \mathbf{t}_n)$  is called *simply moded* if  $p_1(\mathbf{s}_1, \mathbf{t}_1), \dots, p_n(\mathbf{s}_n, \mathbf{t}_n)$  is *simply moded* and

$$\text{Var}(\mathbf{s}_0) \cap \left( \bigcup_{j=1}^n \text{Var}(\mathbf{t}_j) \right) = \emptyset$$

3. A program is called *simply moded* if every clause of it is.

A number of concepts that will be useful for us have been introduced in [19].

**Definition 6.** Let  $G_0, G_1, G_2, \dots, \theta_1, \theta_2, \dots$  be a derivation with selected atoms  $A_0, A_1, A_2, \dots$  and applied renamed clauses  $H^i \leftarrow B_1^i, \dots, B_{n_i}^i$  ( $i = 1, 2, \dots$ ). We say that  $A_k$  is a *direct descendant* of  $A_i$ , if  $k > i$  and  $A_k$  is the atom  $B_j^{i+1} \theta_{i+1} \dots \theta_k$ , ( $1 \leq j \leq n_{i+1}$ ).

**Definition 7.** Let  $G_0, G_1, G_2, \dots, \theta_1, \theta_2, \dots$  be a derivation with selected atoms  $A_0, A_1, A_2, \dots$ . A subsequence of derivation steps,  $G_{i(0)}, G_{i(1)}, \dots, \theta_{i(0)+1}, \dots$  with selected atoms  $A_{i(0)}, A_{i(1)}, A_{i(2)}, \dots$  is *directed*, if for each  $k$  ( $k > 1$ ),  $A_{i(k)}$  is a *direct descendant* of  $A_{i(k-1)}$  in the given derivation.

**Definition 8.** A derivation  $G_0, G_1, G_2, \dots, \theta_1, \theta_2, \dots$  is directed if it is its own directed subsequence.

Verschaetse [19] proved also the following lemma:

**Lemma 1.** Let  $P$  be a definite program and  $A$  an atom. If  $(P, \leftarrow A)$  has an infinite derivation, then it has an infinite directed derivation.

### 3 A necessary and sufficient termination condition

In this section we present a termination condition based on orders on terms. Our first definition is motivated by the following example:

*Example 1.*

$$p(f(X)) \leftarrow p(X)$$

We can always define  $>$ , such that  $p(f(X)) > p(X)$ . However, this information can be used for proving termination only if we refer to the argument position of  $p$ , as to an input position.

Guided by this example we are going to define an order on terms that will refer only to their input positions. More precisely, we start by defining the equivalence relation on the set of atoms and terms, that will “ignore” the output positions, and then define an order on these equivalence classes.

**Definition 9.** Let  $P$  be a moded program, and  $A_1, A_2 \in B_P^E$ . We will say that  $A_1 \sim A_2$  if  $(A_1)^{inP}$  and  $(A_2)^{inP}$  are identical.

We denote by  $\mathcal{T}$  the set  $U_P^E \cup B_P^E / \sim$ .

**Definition 10.** Let  $P$  be a well-moded program,  $>$  a well-founded order on  $\mathcal{T}$  and  $I$  a model for  $P$ . The program  $P$  is called term-acceptable w.r.t.  $>$  and  $I$  if for all  $A \leftarrow B_1, \dots, B_n$  in  $P$  and all substitutions  $\theta$ , such that  $(A\theta)^{inP}$  and  $B_1\theta, \dots, B_{i-1}\theta$  are ground and  $I \models B_1\theta \wedge \dots \wedge B_{i-1}\theta$  holds:  $A\theta > B_i\theta$ .

$P$  is called *term-acceptable* if it is term-acceptable w.r.t. some well-founded order and some model. The following theorem states that term-acceptability of a well-moded program is sufficient for termination of well-moded goals w.r.t. this program.

**Theorem 1.** Let  $P$  be a well-moded program, that is term-acceptable w.r.t. a well-founded order  $>$  and a model  $I$ . Let  $G$  be a well-moded goal. Then  $G$  LD-terminates.

*Proof.* We base our proof on the notion of directed sequence [19]. Let  $G$  be non-terminating, i.e.,  $P \cup \{G\}$  has an infinite derivation. By Lemma 5.1.19 [19] it has an infinite directed derivation as well. Let  $G_0, G_1, \dots$  be this infinite directed derivation. We denote  $G_i = \leftarrow A_1^i, \dots, A_{n_i}^i$  and  $G_{i+1} = \leftarrow A_1^{i+1}, \dots, A_{n_{i+1}}^{i+1}$ . There is

a clause  $H \leftarrow B_1, \dots, B_n$ , and substitutions  $\sigma$  and  $\theta$  such that  $A_1^i = H\sigma$ , for some  $1 \leq j \leq n_{i+1}$ ,  $A_1^{i+1} = B_j\sigma\theta$ ,  $I \models B_1\sigma\theta \wedge \dots, B_{j-1}\sigma\theta$  and  $B_1\sigma\theta \wedge \dots, B_{j-1}\sigma\theta$  is ground. Note, that  $(H\sigma)^{inP}$  is ground due to the well-modedness. The term-acceptability condition implies that  $H\sigma\theta > B_j\sigma\theta$ , that is  $A_1^i\theta > A_1^{i+1}$ . Since  $P$  and  $G$  are well-founded  $(A_1^i)^{inP}$  is ground. Thus,  $A_1^i \sim A_1^i\theta$  and from the definition of  $>$ ,  $A_1^i > A_1^{i+1}$ . Thus, selected atoms of the goals in the infinite directed derivation form an infinite decreasing chain w.r.t.  $>$ , contradicting the well-foundedness of the order.

Note that if the requirement of well-modedness is relaxed the theorem does not hold anymore.

*Example 2.*

$$\begin{aligned} p(a) &\leftarrow q(X) \\ q(f(X)) &\leftarrow q(X) \end{aligned}$$

We assume the modes  $p(i)$  and  $q(i)$  to be given. This program is not well-moded w.r.t. the given modes, but it is term-acceptable with respect to the following order  $>$  on terms

$$p(a) > \dots > q(f(f(f(a)))) > q(f(f(a))) > q(f(a)) > q(a)$$

and a model  $I = \{p(a), q(a), q(f(a)), q(f(f(a))), \dots\}$ . However, note that the well-founded goal  $p(a)$  is non-terminating.

Unfortunately, well-modedness is not sufficient to make the converse to hold. That is, there is a well-moded program  $P$  and a well-moded goal  $G$ , such that  $G$  is LD-terminating w.r.t.  $P$ , but  $P$  is not term-acceptable.

*Example 3.* Consider the following program

$$p(f(X)) \leftarrow p(g(X))$$

with the mode  $p(o)$ . This program is well-moded, the well-moded goal  $p(X)$  terminates w.r.t. this program, but it is not term-acceptable, since  $p(f(X)) \sim p(g(X))$ .

Intuitively, the problem in the example occurred, since some information has been passed via the output positions, i.e,  $P$  is not simply moded. Indeed, if  $P$  is simply moded the second direction of the equivalence holds as well.

We start the proof with a number of useful lemmas.

**Lemma 2.** *Let  $P_0, \dots, P_m$  be directed sequence, such that  $P_i = \leftarrow A_1^i, \dots, A_{n_i}^i$ . Then, for any suffix  $\mathbf{S}$ , exist a sequence of substitutions  $\theta_1, \dots, \theta_m$  and a sequence of suffices  $\mathbf{R}_1, \dots, \mathbf{R}_m$  such that*

$$\leftarrow A_1^0, \mathbf{S}; \leftarrow A_1^1, \mathbf{R}_1\theta_1, \mathbf{S}\theta_1; \dots; \leftarrow A_{n_m}^m, \mathbf{R}_m\theta_1 \dots \theta_m, \mathbf{S}\theta_1 \dots \theta_m$$

*is directed.*

**Lemma 3.** *Let  $P_0, \dots, P_m$  and  $Q_0, \dots, Q_k$  be two directed sequences, such that  $P_m = \leftarrow A_1, \dots, A_s$  and  $Q_0 = \leftarrow B_1, \dots, B_t$  and  $A_1 = B_1$ . Then, exists a directed sequence  $R_0, \dots, R_{m+k}$ , such that the selected atom of  $R_0$  is the selected atom of  $P_0$ , and the selected atom of  $R_{m+k}$  is the selected atom of  $Q_k$ .*

*Proof.* We define  $R_0, \dots, R_{m+k}$  as following:

$$R_i = \begin{cases} P_i & \text{if } 0 \leq i \leq m \\ T_{i-m} & \text{if } m \leq i \leq m+k \end{cases}$$

where  $T_j$  is the  $j$ -th element in the sequence, generated by Lemma 2 for the directed sequence  $Q_0, \dots, Q_k$  with  $\mathbf{S} = A_2, \dots, A_s$ .

The sequence  $R_0, \dots, R_{m+k}$  is well-defined: if  $i = m$ , on one hand we get that  $R_m = P_m$ , and on the other hand,  $R_m = B_1, \mathbf{S}$ , that is  $P_m$ . For  $i \neq m$  only one of those definitions is applicable.

The requirement of the lemma are clearly fulfilled.

**Corollary 1.** *Let  $P$  be a simply moded program and  $Q, Q'$  - simply moded goals. Let  $P_0, \dots, P_m$  be a directed sequence obtained from one of the derivations for  $P \cup \{Q\}$  and  $Q_0, \dots, Q_k$  be a directed sequence obtained from one of the derivations for  $P \cup \{Q'\}$ . Let also  $P_m$  be  $\leftarrow A_1, \dots, A_s$ ,  $Q_0$  be  $\leftarrow B_1, \dots, B_t$  and  $A_1 \sim B_1$ . Then, exists a directed sequence  $R_0, \dots, R_{m+k}$ , such that the selected atom of  $R_0$  is the selected atom of  $P_0$ , and the selected atom of  $R_{m+k}$  is the selected atom of  $Q_k$ .*

*Proof.* Since  $P_m$  and  $Q_0$  are queries in some of the LD-derivations of the simply moded queries and of the simply moded program, they are simply moded [2]. Thus, the output positions, both of  $A_1$  and of  $B_1$ , are occupied by distinct variables. Since  $A_1^{inP} = B_1^{inP}$  we can claim that  $A_1 = B_1$ , up to variables renaming. Thus, Lemma 3 becomes applicable (note that we never required in the lemma that both directed sequences shall originate from the derivations of the same LD-tree), and we can obtain a new directed sequence as required.

**Theorem 2.** *Let  $P$  be a well-moded and simply moded program, LD-terminating for any well-moded and simply-moded goal. Then there exists a model  $I$  and a well-founded order  $>$ , such that  $P$  is term-acceptable w.r.t.  $I$  and  $>$ .*

*Proof.* We base the choice of  $>$  on the LD-trees. More precisely, we define  $A > B$  if there is a well-moded and simply moded goal  $G$  and there is a directed sequence  $P_0, \dots, P_m$  in the LD-tree for  $P \cup \{G\}$ , such that the selected atom of  $P_0$ ,  $A_1$ , is equivalent to  $A$  and the selected atom of  $P_m$ ,  $B_1$  is equivalent to  $B$ . Let  $I$  be a lest Herbrand model of  $P$ . We have to prove that:

1.  $>$  is an order relationship, i.e., is irreflexive, asymmetric and transitive;
  2.  $>$  is well-founded;
  3.  $P$  is term-acceptable w.r.t.  $>$  and  $I$
1.  $>$  is an order relationship, that is  $>$  is irreflexive, asymmetric and transitive.

- (a) Irreflexivity. If  $A > A$  holds, then exists a directed sequence  $P_0, \dots, P_k$ , such that the selected atom of  $P_0$  is  $A_1$ , the selected atom of  $P_k$  is  $A_2$  and  $A_1 \sim A$  and  $A_2 \sim A$ . By repetitive application of Corollary 1 an infinite branch is build and the contradiction to the finiteness of the LD-tree is obtained.
- (b) Asymmetry. If  $A > B$  holds, then exists a directed sequence  $P_0, \dots, P_k$ , such that the selected atom of  $P_0$  is  $A_1$ , the selected atom of  $P_k$  is  $B_1$  and  $A_1 \sim A$  and  $B_1 \sim B$ . If  $B > A$  holds, then exists a directed sequence  $Q_0, \dots, Q_m$ , such that the selected atom of  $Q_0$  is  $B_2$ , the selected atom of  $Q_m$  is  $A_2$  and  $A_2 \sim A$  and  $B_2 \sim B$ . By repetitive application of Corollary 1 an infinite branch is build and the contradiction to the finiteness of the LD-tree is obtained.
- (c) Transitivity. If  $A > B$  holds, then exists a directed sequence  $P_0, \dots, P_k$ , such that the selected atom of  $P_0$  is  $A_1$ , the selected atom of  $P_k$  is  $B_1$  and  $A_1 \sim A$  and  $B_1 \sim B$ . If  $B > C$  holds, then exists a directed sequence  $Q_0, \dots, Q_m$ , such that the selected atom of  $Q_0$  is  $B_2$ , the selected atom of  $Q_m$  is  $C_2$  and  $B_2 \sim B$  and  $C_2 \sim C$ . By applying the Corollary 1 new directed sequence is build, such that the selected atom of its first element is  $A_1$  and the selected atom of its last element is  $C_2$ . By definition of  $>$  holds that  $A > C$ .
2.  $>$  is well-founded. Assume that there is an infinitely decreasing chain  $A_1 > A_2 > \dots$ . This means that there is an infinite directed sequence in the tree (concatenation of infinitely many finite ones), contradicting the finiteness of the tree.
3.  $P$  is term-acceptable w.r.t.  $>$  and  $I$ . Let  $A \leftarrow B_1, \dots, B_n$ . Let  $\theta$  be a substitution, such that  $(A\theta)^{inP}, B_1\theta, \dots, B_{i-1}\theta$  are ground and  $I \models B_1\theta \wedge \dots \wedge B_{i-1}\theta$ . The goal  $\leftarrow A\theta$  is a well-moded goal, however, it is not necessary simply moded. Thus, we define a new goal  $A'$  such that it will coincide with  $A\theta$  on its input positions, and its output positions will be occupied by a linear set of variables.

More formally, let  $\theta_1$  be  $\theta$  restricted to  $\text{Var}(A)^{inP}$ . Then  $(A\theta_1)^{inP} = (A\theta)^{inP}$ , and thus,  $(A\theta_1)^{inP}$  is ground, while  $(A\theta_1)^{out} = A^{out}$ , and thus,  $(A\theta_1)^{out}$  is a linear sequence of variables. Summing up,  $\leftarrow A\theta_1$  is well-moded and simply moded goal. Thus, it terminates w.r.t.  $P$  and its derivations has been considered while defining  $>$ .

By definition of  $\theta_1$  exists some substitution  $\sigma$ , such that  $\theta = \theta_1\sigma$ . Thus,  $B_1\theta, \dots, B_{i-1}\theta = (B_1\theta_1, \dots, B_{i-1}\theta_1)\sigma$ . Since  $I$  is a least Herbrand model  $\sigma$  is a correct answer substitution of  $B_1\theta_1, \dots, B_{i-1}\theta_1$  and, since  $P$  and  $\leftarrow B_1\theta_1, \dots, B_{i-1}\theta_1$  are well-moded (the later as the LD-resolvent of the well-moded clause and the well-moded goal),  $\sigma$  is a computed answer substitution as well [1]. Thus, the next goal to be considered in the derivation is  $\leftarrow B_i\theta_1\sigma, \dots, B_n\theta_1\sigma$ . This is a directed descendant of  $A\theta_1$ , thus, by definition of  $>$ ,  $A\theta_1 > B_i\theta_1\sigma$ . By definition of  $\sigma$ ,  $B_i\theta_1\sigma = B_i\theta$ , and by the definition of  $\theta_1$ ,  $A\theta_1 \sim A\theta$ . Thus,  $A\theta > B_i\theta$ .

*Example 4.* The following example is common in proving termination and appeared, for example, in [1, 10].

$$\begin{aligned} & \text{permute}([], []). \\ & \text{permute}(L, [El|T]) \leftarrow \text{delete}(El, L, L_1), \text{permute}(L_1, T). \\ & \text{delete}(X, [X|T], T). \\ & \text{delete}(X, [H|T], [H|T_1]) \leftarrow \text{delete}(X, T, T_1). \end{aligned}$$

The following modes are assumed to be given:

$$\text{delete}(o, i, o), \text{permute}(i, o)$$

This program is well-moded w.r.t. these modes. Thus, in order to prove termination of well-moded goals w.r.t. it, we have to find a well-founded order  $>$  on  $\mathcal{T}$  and a model  $I$ , such that the program will be term-acceptable w.r.t.  $>$  and  $I$ .

For the sake of simplicity we write  $A^{inp}$  instead of  $A$  in the requirements for  $>$  on elements of  $B_P^E / \sim$ , i.e., omit the output positions. This brief notation is justified, since  $>$  respects  $\sim$ , and thus, cannot depend on the output positions.

Let  $>$  be any ordering on  $\mathcal{T}$  having the subterm and monotonicity properties and such that for all  $t_2, s_1 \in U_P^E$ :  $\text{delete}(t_2) < \text{permute}(s_1)$ . Let

$$I = \{\text{delete}(t_1, t_2, t_3) \in B_P \mid t_2 > t_3\} \cup \{\text{permute}(t_1, t_2) \mid t_1, t_2 \in U_P\}$$

First, we verify that  $I$  is a model. This can be done by checking that  $T_P(I) \subseteq I$ , where  $T_P$  is the immediate consequence operator. For *permute*/2 clauses, this is trivially fulfilled. For the non-recursive clause of *delete*/3, it requires verifying that for all  $t_1, t_2 \in U_P$  we have  $\text{delete}(t_1, [t_1|t_2], t_2) \in I$ . Thus, we need to check that  $[t_1|t_2] > t_2$ , which holds due to the subterm property of  $>$ .

To verify the recursive clause of *delete*/3, “if  $\text{delete}(t_1, t_2, t_3) \in I$  then also  $\text{delete}(t_1, [t|t_2], [t|t_3]) \in I$ ” has to be checked. This requires verifying that  $t_2 > t_3$  implies  $[t|t_2] > [t|t_3]$ , which follows from the monotonicity of  $>$ .

Next we verify the term-acceptability condition. For the recursive clause of *delete*/3, we need that  $\text{delete}([H|T])\theta > \text{delete}(T)\theta$ , for each grounding  $\theta$ . This follows from the subterm and monotonicity properties of  $>$ .

For the recursive clause for *permute*/2, first note that the decrease w.r.t. the *delete*/3-body atom is fulfilled, because we imposed it on  $>$ . For the recursive body atom, we need that, if  $I \models \text{delete}(El, L, L_1)\theta$ , then  $\text{permute}(L)\theta > \text{permute}(L_1)\theta$ . Due to the monotonicity of  $>$ , it is sufficient that  $L\theta > L_1\theta$  is implied by  $I \models \text{delete}(El, L, L_1)\theta$ , which holds because of  $I$ .

The following observation is readily generalised from the example above. Assume that we have a clause  $p(t_1, \dots, t_n) \leftarrow B_1, \dots, B_{i-1}, q(s_1, \dots, s_m), \dots, B_k$ , such that  $p/n$  and  $q/m$  are not mutually dependent on each other (such as the *delete* atom in the recursive clause for *permute*). Then, we can always impose that  $p^{inp}(u_1, \dots, u_{n'}) > q^{inp}(v_1, \dots, v_{m'})$ , for all  $u_1, \dots, u_{n'}, v_1, \dots, v_{m'} \in U_P$ . Such constraints on  $>$  will be orthogonal to all other constraint we may impose

on it. Therefore, in the remainder of the paper, we will only check the acceptability decrease  $A\theta > B_i\theta$  for body atoms  $B_i$  with a predicate symbol that is mutually dependent on that of  $A$ .

Note that, comparing with standard acceptability, this is not surprising. In standard acceptability, the decrease for non-mutually dependent predicates is only needed to ensure that a boundedness property is inherited from head to body-atoms in the clauses. Since we are focusing on well-moded programs and queries only, boundedness follows from groundedness of the input and is always inherited by body atoms.

## 4 Towards automation of the approach

In this section we present an approach to automatic verification of the term-acceptability condition. The basic idea for the approach is inspired on the “constraint based” termination analysis proposed in [10]. We start off from the conditions imposed by term-acceptability, and systematically reduce these conditions to more explicit constraints on the objects of our search: the order  $>$  and the model  $I$ .

This approach has been applied successfully to a number of examples that appear in the literature on termination, such as different versions of *permute* [5, 16, 10], *dis-con* [7], *transitive closure* [16], *add-mult* [17], *combine*, *reverse*, *odd-even*, *at\_least\_double* and *normalization* [10], *quicksort* program [18, 1], *derivative* [13], *distributive law* [12], *boolean ring* [15], *flatten* [4].

In the remainder of the paper, we explain the approach using some of these examples.

### 4.1 Permute revisited

The analysis begins with the  $>$ -decreases imposed in the term-acceptability. For the *permute* example, these are:

$$\text{delete}([H|T])\theta > \text{delete}(T)\theta \quad (1)$$

$$I \models \text{delete}(El, L, L_1)\theta \text{ implies } \text{permute}(L)\theta > \text{permute}(L_1)\theta \quad (2)$$

For each such condition, we have two options on how to enforce it:

Option 1): The decrease required in the condition can be achieved by imposing some property on  $>$ , which is consistent with the constraints that were already imposed on  $>$  before.

In the *permute* example, condition (1) is satisfied by imposing the subterm property on  $>$ .

In general we can select from a bunch of term-order properties, or even specific term-orders, that were proposed in the literature. Orderings that may be considered are general orderings, having subterm and monotonicity properties; recursive path ordering with status [14]; lexicographic ordering; etc.

Option 2): The required decrease is imposed as a constraint on the model  $I$ .

In the *permute* example, the decrease  $\text{permute}(L)\theta > \text{permute}(L_1)\theta$  cannot directly be achieved by imposing some constraint on  $>$ . Thus, we impose that the underlying decrease  $L\theta > L_1\theta$  should hold for the intermediate body atoms ( $\text{delete}(El, L, L_1)\theta$ ) that are in the model  $I$ .

Thus, in the example, the constraint is that  $I$  should be such that for all  $\text{delete}(t_1, t_2, t_3) \in I$ :  $t_2 > t_3$ . One way to constructively verify that an  $I$  exists such that the property  $t_2 > t_3$  holds for its  $\text{delete}(t_1, t_2, t_3)$  atoms is to simply impose that  $S = \{\text{delete}(t_1, t_2, t_3) \mid t_2 > t_3\}$  itself is a model for the *delete* clauses in the program.

So our new constraint on  $I$  is that it should include  $S$ . Practically we can enforce this by imposing that  $T_P(S) \subseteq S$  should hold. As shown in the previous section, this reduces to the constraints “[ $t_1|t_2$ ]  $>$   $t_2$ ” and “ $t_2 > t_3$  implies [ $t|t_2$ ]  $>$  [ $t|t_3$ ]”. These are again fed into our Option 1) step, imposing monotonicity and subterm properties on  $>$ . At this point the proof is complete.

## 4.2 Dealing with multiply intermediate body atoms

The *permute* example does not illustrate the approach in full generality. It might happen that more than one intermediate goal preceded the recursive atom in the body clause. In this case we refer to the whole conjunction as to “one” subgoal. Formally, given a sequence of intermediate body atoms  $B_1, \dots, B_n$  a (generalised) clause  $B_1, \dots, B_n \leftarrow B_1, \dots, B_n$  is constructed and one step of unfolding is performed on each atom in its body, producing a generalized program  $P'$ .

*Example 5.* The following is the version of the *permute* program that appeared in [16].

$$\begin{array}{ll} \text{perm}([], []). & \text{ap}_1([], L, L). \\ \text{perm}(L, [H|T]) \leftarrow & \text{ap}_1([H|L_1], L_2, [H|L_3]) \leftarrow \\ & \text{ap}_2(V, [H|U], L), \quad \text{ap}_1(L_1, L_2, L_3). \\ \text{ap}_1(V, U, W), & \text{ap}_2([], L, L). \\ \text{perm}(W, T). & \text{ap}_2([H|L_1], L_2, [H|L_3]) \leftarrow \\ & \text{ap}_2(L_1, L_2, L_3). \end{array}$$

Assume the modes

$$\text{perm}(i, o), \text{ap}_1(i, i, o), \text{ap}_2(o, o, i)$$

The term-acceptability imposes, among the others, the following  $>$ -decrease:  $I \models \text{ap}_2(V, [H|U], L)\theta \wedge \text{ap}_1(V, U, W)\theta$  implies  $\text{perm}(L)\theta > \text{perm}(W)\theta$ . Note that the underlying decrease  $L\theta > W\theta$  cannot be achieved by reasoning on  $\text{ap}_1/3$  or  $\text{ap}_2/3$  alone. Therefore, we construct a following program  $P'$ :

$$\begin{array}{l} \text{ap}_2([], [t_1|t_2], [t_1|t_2]), \text{ap}_1([], t_2, t_2). \\ \text{ap}_2([t_6|t_1], [t_5|t_2], [t_6|t_3]), \text{ap}_1([t_6|t_1], t_2, [t_6|t_4]) \leftarrow \\ \text{ap}_2(t_1, [t_5|t_2], t_3), \text{ap}_1(t_1, t_2, t_4). \end{array}$$

Now, we need to verify that  $S = \{\text{ap}_2(a_1, a_2, a_3), \text{ap}_1(b_1, b_2, b_3) \mid a_3 > b_3\}$  satisfies  $T_{P'}(S) \subseteq S$ . Using the 2 clauses, this is reduced to “[ $t_1|t_2$ ]  $>$   $t_2$ ” and “ $t_3 > t_4$  implies [ $t_6|t_3$ ]  $>$  [ $t_6|t_4$ ]”, imposing monotonicity and subterm properties on  $>$ . The proof is completed analogously to the previous example.

### 4.3 Further examples

*Example 6.* The following example is the famous *quicksort* program [18, 1].

$$\begin{array}{ll}
\text{quicksort}([X|Xs], Ys) \leftarrow & \text{partition}([X|Xs], Y, [X|Ls], Bs) \leftarrow \\
\text{partition}(Xs, X, Littles, Bigs), & X \leq Y, \text{partition}(Xs, Y, Ls, Bs). \\
\text{quicksort}(Littles, Ls), & \text{partition}([X|Xs], Y, Ls, [X|Bs]) \leftarrow \\
\text{quicksort}(Bigs, Bs), & X > Y, \text{partition}(Xs, Y, Ls, Bs). \\
\text{append}(Ls, [X|Bs], Ys). & \text{partition}([], Y, [], []). \\
\text{quicksort}([], []). & \\
\text{append}([X|Xs], Ys, [X|Zs]) \leftarrow & \text{append}([], Xs, Xs). \\
\text{append}(Xs, Ys, Zs). &
\end{array}$$

with the following moding information:

$$\text{quicksort}(i, o), \text{partition}(i, i, o, o), \text{append}(i, i, o)$$

As before, the analysis begins with constructing the  $>$ -decreases, imposed in the term-acceptability. The following ones are obtained:

$$\text{append}([X|Xs], Ys)\theta > \text{append}(Xs, Ys)\theta \quad (3)$$

$$I \models (X \leq Y)\theta \quad (4)$$

$$\text{implies } \text{partition}([X|Xs], Y)\theta > \text{partition}(Xs, Y)\theta$$

$$I \models (X > Y)\theta \quad (5)$$

$$\text{implies } \text{partition}([X|Xs], Y)\theta > \text{partition}(Xs, Y)\theta$$

$$I \models \text{partition}(Xs, X, Littles, Bigs)\theta \quad (6)$$

$$\text{implies } \text{quicksort}([X|Xs])\theta > \text{quicksort}(Littles)\theta$$

$$I \models \text{partition}(Xs, X, Littles, Bigs)\theta \wedge \text{quicksort}(Littles, Ls)\theta \quad (7)$$

$$\text{implies } \text{quicksort}([X|Xs])\theta > \text{quicksort}(Bigs)\theta$$

Conditions (3-5) can be satisfied if monotonicity and subterm properties on  $>$  are imposed. The decrease  $\text{quicksort}([X|Xs])\theta > \text{quicksort}(Littles)\theta$  in condition (6) cannot be directly achieved by imposing some constraint on  $>$ . Thus, we impose that the underlying decrease  $[X|Xs]\theta > Littles\theta$  should hold for the intermediate body atoms ( $\text{partition}(Xs, X, Littles, Bigs)\theta$ ) that are in the model  $I$ . In other words, the constraint is that  $I$  should be such that for all  $\text{partition}(t_1, t_2, t_3, t_4) \in I: [t_2|t_1] > t_3$ . We define  $S = \{\text{partition}(t_1, t_2, t_3, t_4) \mid [t_2|t_1] > t_3\}$  and, as above, impose a new constraint that  $I$  should include  $S$ , that is  $T_P(S) \subseteq S$  should hold. The later inclusion reduces to three constraints:

$$[t_2] > [] \quad (8)$$

$$[t_2|t_1] > t_3 \text{ implies } [t_2, t|t_1] > [t|t_3] \quad (9)$$

$$[t_2|t_1] > t_3 \text{ implies } [t_2, t|t_1] > t_3 \quad (10)$$

Conditions (8) and (10) are fed up into Option 1) step, imposing monotonicity and subterm properties on  $>$ . However, the fulfilment of the condition (9) does

not necessary follow from these properties of  $>$ . There are two ways to cope with this kind of problem. Either to *restrict an order*, i.e., to chose a more specific order that allows to prove the desired decrease, or to *narrow down the set  $S$* , i.e., to find an additional constraint  $Q$ , such that  $S' = \{partition(t_1, t_2, t_3, t_4) \mid [t_2|t_1] > t_3 \ \& \ Q(t_1, t_2, t_3, t_4)\}$  and  $T_P(S') \subseteq S'$  can be easily proved.

Restricting to a specific order. One may choose, for example, an order that is based on the term-size norm for the elements of  $U_P$  and further is extended by level-mappings, measuring egalitarianly input positions, and only them. One can verify, that this order has subterm and monotonicity properties. More precise, for  $t_1, t_2 \in U_P$  we define that  $t_1 > t_2$  if  $\|t_1\| > \|t_2\|$ , where  $\|\cdot\|$  is the term-size norm. For this particular  $>$ , if  $[t_2|t_1] > t_3$ , i.e.,  $1 + \|t_2\| + \|t_1\| > \|t_3\|$ , then  $1 + \|t_2\| + (1 + \|t\| + \|t_1\|) > 1 + \|t\| + \|t_3\|$ , that is  $\|[t_2, t|t_1]\| > \|[t|t_3]\|$ , and by definition of  $>$ ,  $[t_2, t|t_1] > [t|t_3]$ . This shows that our way of reasoning is a conservative extension of the traditional reasoning based on the acceptability condition and using only monotonicity and equality norm-based interargument relationships. Note, however, that from a term-rewriting perspective an ordering based on term-size is not in the usual collection of applied term-orders. This motivates the alternative approach below.

Narrowing down the set. The first step is to determine the additional constraint  $Q$ , that should hold for  $partition/3$ . In our case observe, that the following should hold:

$$[t_2|t_1] > t_3 \ \& \ Q(t_1, t_2, t_3, t_4) \text{ implies } [t_2, t|t_1] > [t|t_3] \ \& \ Q([t|t_1], t_2, [t|t_3], t_4)$$

That is  $[t_2|t_1] > t_3 \ \& \ Q(t_1, t_2, t_3, t_4)$  implies  $[t_2, t|t_1] > [t|t_3]$ , or  $[t_2|t_1] > t_3 \ \& \ \neg([t_2, t|t_1] > [t|t_3])$  implies  $\neg Q(t_1, t_2, t_3, t_4)$ . Now we replace a negation  $\neg(a > b)$  by a disjunction  $(a \leq b) \vee (a \| b)$  (recall that  $\|$  denotes incomparable) and separate the cases.

1.  $[t_2|t_1] > t_3 \ \& \ [t|t_3] \geq [t_2, t|t_1]$  implies  $\neg Q(t_1, t_2, t_3, t_4)$ . Then,  $[t_2, t|t_1] > [t|t_1]$  by the subterm property and thus,  $[t|t_3] > [t|t_1]$ . Monotonicity implies that either  $t_3 > t_1$  or  $t_3 \| t_1$ .
2.  $[t_2|t_1] > t_3 \ \& \ [t_2, t|t_1] \| [t|t_3]$  implies  $\neg Q(t_1, t_2, t_3, t_4)$ . Then,  $[t_2, t|t_1] > [t|t_1]$  by the subterm property. If  $[t|t_1] \geq [t|t_3]$ , one gets a contradiction with  $[t_2, t|t_1] \| [t|t_3]$ . Thus, either  $[t|t_3] > [t|t_1]$  or  $[t|t_3] \| [t|t_1]$ . Both cases stem to either  $t_3 > t_1$  or  $t_3 \| t_1$ .

Thus,  $\neg Q(t_1, t_2, t_3, t_4)$  is a conjunction, and one of its conjuncts is  $((t_3 > t_1) \vee (t_3 \| t_1))$ . Thus,  $Q(t_1, t_2, t_3, t_4)$  is a disjunction and one of its disjuncts is  $((t_1 > t_3) \vee (t_1 = t_3))$ . Thus, define  $S' = \{partition(t_1, t_2, t_3, t_4) \mid [t_2|t_1] > t_3 \ \& \ t_1 \geq t_3\}$ .

The second step is to prove that  $T_P(S') \subseteq S'$ . The proof can be done by the technique presented above. To complete the proof, observe that the condition (7) can be analysed in the way similar to the example from [16].

*Example 7.* Our last example is the motivating example from the Introduction, i.e., computing the derivative of a linear function in one variable  $t$ :

$$d(der(t), 1).$$

$$\begin{aligned}
d(\text{der}(A), 0) &\leftarrow \text{number}(A). \\
d(\text{der}(X + Y), DX + DY) &\leftarrow d(\text{der}(X), DX), d(\text{der}(Y), DY). \\
d(\text{der}(X * Y), X * DY + Y * DX) &\leftarrow d(\text{der}(X), DX), d(\text{der}(Y), DY). \\
d(\text{der}(\text{der}(X)), DD X) &\leftarrow d(\text{der}(X), DX), d(\text{der}(DX), DD X).
\end{aligned}$$

We assume that the moding information  $d(i, o)$  is given. We start by constructing the  $>$ -decreases that follow from the term-acceptability.

$$d(\text{der}(X + Y))\theta > d(\text{der}(X))\theta \quad (11)$$

$$I \models d(\text{der}(X), DX)\theta \text{ implies } d(\text{der}(X + Y))\theta > d(\text{der}(Y))\theta \quad (12)$$

$$d(\text{der}(X * Y))\theta > d(\text{der}(X))\theta \quad (13)$$

$$I \models d(\text{der}(X), DX)\theta \text{ implies } d(\text{der}(X * Y))\theta > d(\text{der}(Y))\theta \quad (14)$$

$$d(\text{der}(\text{der}(X)))\theta > d(\text{der}(X))\theta \quad (15)$$

$$I \models d(\text{der}(X), DX)\theta \text{ implies } d(\text{der}(\text{der}(X)))\theta > d(\text{der}(DX))\theta \quad (16)$$

Conditions (11)-(15) impose monotonicity and subset properties to hold on  $>$ . In order to satisfy condition (16), it is sufficient to prove that in  $I$ , the first argument of  $d$  is greater than the second one, that is that if  $S = \{d(t_1, t_2) \mid t_1 > t_2\}$  then  $T_P(S) \subseteq S$  and  $S \subseteq I$ . This may be reduced to the following conditions:

$$\text{der}(t) > 1 \quad (17)$$

$$I \models \text{number}(t_1) \text{ implies } \text{der}(t_1) > 0 \quad (18)$$

$$\text{der}(t_1) > t_2 \ \& \ \text{der}(t_3) > t_4 \text{ implies } \text{der}(t_1 + t_3) > t_2 + t_4 \quad (19)$$

$$\text{der}(t_1) > t_2 \ \& \ \text{der}(t_3) > t_4 \text{ implies } \text{der}(t_1 * t_3) > t_1 * t_4 + t_2 * t_3 \quad (20)$$

$$\text{der}(t_1) > t_2 \ \& \ \text{der}(t_2) > t_3 \text{ implies } \text{der}(\text{der}(t_1)) > t_3 \quad (21)$$

Condition (21) follows from monotonicity and transitivity of  $>$ . However, (18)-(20) are not satisfied by general properties of  $>$  and we choose, to specify the order. The order that meets these conditions is the recursive path ordering [11] with  $\text{der}$  having the highest priority.

## 5 Discussion and conclusions

We have presented a non-transformational approach to termination analysis of logic programs, based on general term-orderings. The approach gets its power from integrating the traditional notion of acceptability [3] with the wide class of term-orderings that have been studied in the context of the term-rewriting systems. In theory, such an integration is unnecessary: acceptability (based on level mappings only) is already equivalent to LD-termination. In practice, the required level mappings may sometimes be very complex, and automatic systems for proving termination are unable to generate them. In such cases, generating an appropriate term-ordering, replacing the level mapping, may often be much easier, especially since we can reuse the impressive machinery on term-orders developed for term-rewrite systems.

Unlike transformational approaches, that establish the termination results for logic programs by the reasoning on termination of term-rewriting systems, we apply the term-orderings directly to the logic programs, thus, avoiding transformations. This could both be regarded as an advantage and as a drawback of our approach. It may be considered as a drawback, because reasoning on successful instances of intermediate body-atoms introduces an additional complication in our approach, for which there is no counterpart in transformational methods (except for in the transformation step itself). On the other hand, we consider it as an advantage, because it is precisely this reasoning on intermediate body atoms that gives more insight in the property of *logic program termination* (as opposed to *term-rewrite system termination*).

So, in a sense our approach provides the best of both worlds: a means to incorporate into ‘direct’ approaches the generality of general term-orderings.

However, in its current version, the proposed technique is not a fully conservative extension of standard acceptability. For the moment, our approach is limited to well-moded programs and goals only. In particular, termination proofs are only obtained for goals with ground input. In standard acceptability, the termination proof is valid for all goals that are bounded under the considered level mapping.

Recall, that the goal is *bounded* [6] if the level mapping is bounded on the set of all ground instances of the goal. This notion ensures that the level mapping of a (non-ground) goal can only increase up to some finite bound when the goal becomes more instantiated. Observe that every ground goal is trivially bounded.

In ongoing work, we are extending our technique to non-ground input. A main complication is that generalising the notion of boundedness to general term-orderings is not straightforward. One particular possible generalisation of boundedness to term-orderings, which is useful for maintaining most of our results, is:

An atom  $A$  is *bounded* with respect to an ordering  $<$ , if there exists an atom  $C$  such that for all ground instances  $A\theta$  of  $A$ :  $A\theta < C$ , and  $\{B \in B_P^E \mid B < C\}$  is finite.

Such a definition imposes constraints which are very similar to the ones imposed by standard boundedness in the context of level mappings. However, one thing we loose is that it is no longer generalisation of of groundness. Consider an atom  $p(a)$  and assume that our language contains a functor  $f/1$  and a constant  $b$ . Then one particular well-founded ordering is

$$p(a) > \dots > p(f(f(b))) > p(f(b)) > p(b).$$

So,  $p(a)$  is not bounded with respect to this ordering.

Of course, this is not really a problem. Our formulation of the approach for non-ground (input) goals and general (not well-moded) programs does not necessary have to capture ground goals. To illustrate this, consider again the Example 2.

$$\begin{aligned} p(a) &\leftarrow q(X) \\ q(f(X)) &\leftarrow q(X) \end{aligned}$$

Consider also the well-founded ordering

$$p(a) > \dots > q(f(f(X))) > q(f(X)) > q(X)$$

Note that there is a strict decrease in the term-ordering, from head to body, for both clauses. Also, note that the goal  $p(a)$  is not LD-terminating. So, if we would have insisted on a definition of boundedness with respect to  $>$  which generalises groundness (so that  $p(a)$  would be bounded in this example), then any reasonable extension of term-acceptability would not imply LD-termination for all bounded goals (in particular:  $p(a)$  in the example).

Our next priority for future work is a full implementation of the approach. Although we already tested very many examples manually, an implementation will allow us to conduct a much more extensive experimentation, comparing the technique also in term of efficiency with other systems. Since we apply a demand-driven approach, systematically reducing required conditions to more simple constraints on the ordering and the model, we expect that the method can lead to very efficient verification.

Another line of future work is to develop an extension to term-orderings for the notion of “acceptability with respect to a set of atoms” of [8, 9]. We expect that this extension may be much easier in the non-ground case, because “acceptability with respect to a set” does not require any notion of boundedness.

## 6 Acknowledgement

We thank Sofie Verbaeten and Maurice Bruynooghe for comments on this work. Alexander Serebrenik is supported by GOA: “ $LP^+$ : a second generation logic programming language”. Danny De Schreye is a senior research associate of FWO Flanders, Belgium.

## References

1. K. R. Apt. *From Logic Programming to Prolog*. Prentice-Hall International Series in Computer Science. Prentice Hall, 1997.
2. K. R. Apt and S. Etalle. On the unification free Prolog programs. In A. M. Borzyszkowski and S. Sokolowski, editors, *18th International Symposium on Mathematical Foundations of Computer Science*, pages 1–19. Springer Verlag, 1993. Lecture Notes in Computer Science, volume 711.
3. K. R. Apt and D. Pedreschi. Studies in Pure Prolog: Termination. In J. W. Lloyd, editor, *Proceedings Esprit Symposium on Computational Logic*, pages 150–176. Springer Verlag, 1990.
4. T. Arts. *Automatically proving termination and innermost normalisation of term rewriting systems*. PhD thesis, Faculteit Wiskunde & Informatica, Universiteit Utrecht, 1997.
5. T. Arts and H. Zantema. Termination of logic programs using semantic unification. In M. Proietti, editor, *Logic Programming Synthesis and Transformation, 5th International Workshop. LOPSTR'95*, pages 219–233. Springer Verlag, 1995. Lecture Notes in Computer Science, volume 1048.

6. M. Bezem. Characterizing termination of logic programs with level mappings. In E. L. Lusk and R. A. Overbeek, editors, *Logic Programming, Proceedings of the North American Conference 1989*, pages 69–80. MIT Press, 1989.
7. D. De Schreye and S. Decorte. Termination of logic programs: The never-ending story. *The Journal of Logic Programming*, 19/20:199–260, May/July 1994.
8. D. De Schreye, K. Verschaetse, and M. Bruynooghe. A framework for analyzing the termination of definite logic programs with respect to call patterns. In I. Staff, editor, *Fifth Generation Computer Systems '92: Proceedings of the International Conference on Fifth Generation Computer Systems.*, pages 481–488. IOS Press, 1992.
9. S. Decorte and D. De Schreye. Termination analysis: some practical properties of the norm and level mapping space. In J. Jaffar, editor, *Proceedings of the 1998 Joint International Conference and Symposium on Logic Programming*, pages 235–249. MIT Press, June 1998.
10. S. Decorte, D. De Schreye, and H. Vandecasteele. Constraint-based termination analysis of logic programs. *ACM Transactions on Programming Languages and Systems (TOPLAS)*, 21(6):1137–1195, November 1999.
11. N. Dershowitz. Termination. In C. Kirchner, editor, *Proceedings of the First International Conference on Rewriting Techniques and Applications*, pages 180–224. Springer Verlag, 1985. Lecture Notes in Computer Science, volume 202.
12. N. Dershowitz and C. Hoot. Topics in termination. In C. Kirchner, editor, *Rewriting Techniques and Applications, 5th International Conference*, pages 198–212. Springer Verlag, 1993. Lecture Notes in Computer Science, volume 690.
13. N. Dershowitz and Z. Manna. Proving termination with multiset orderings. *Communications of the ACM (CACM)*, 22(8):465–476, August 1979.
14. M. C. Ferreira and H. Zantema. Well-foundedness of term orderings. In N. Dershowitz, editor, *Conditional Term Rewriting Systems, proceedings fourth international workshop CTRS-94*, pages 106–123. Springer Verlag, 1995. Lecture Notes in Computer Science, volume 968.
15. J. Hsiang. Rewrite method for theorem proving in first order theory with equality. *Journal of Symbolic Computation*, 8:133–151, 1987.
16. M. Krishna Rao, D. Kapur, and R. Shyamasundar. Transformational methodology for proving termination of logic programs. *The Journal of Logic Programming*, 34:1–41, 1998.
17. L. Plümer. *Termination Proofs for Logic Programs*. Lecture Notes in Artificial Intelligence, volume 446. Springer Verlag, 1990.
18. L. Sterling and E. Shapiro. *The Art of Prolog*. The MIT Press, 1994.
19. K. Verschaetse. *Static termination analysis for definite Horn clause programs*. PhD thesis, Departement Computerwetenschappen, K.U.Leuven, 1992.