Contributions to a theory of existential termination for definite logic programs

Giorgio Levi and **Francesca Scozzari**
Dipartimento di Informatica,
Università di Pisa,
Corso Italia 40, 56125 Pisa, Italy
{levi,scozzari}@di.unipi.it

Abstract

We suggest a new formalization of the existential termination problem of logic programs under the PROLOG leftmost selection rule and depth-first computation rule. First of all, we give a characterization of the problem in terms of occurrence sets, by proving that a $\langle program, goal \rangle$ existentially terminates if and only if there exists a finite correct occurrence set. Then we show that in order to study existential termination, we do not need to specify the occurrences of the atoms, since existential termination turns out to be decidable, when instances of atoms are used more than once (up to renaming).

We then reduce the verification of existential termination to the search of a suitable semi occurrence set for the pair $\langle program, goal \rangle$, by providing an algorithm for proving that the proposed semi occurrence set is a correct occurrence set. Finally we propose a simple method (based on abstract interpretation techniques) for generating such semi occurrence sets.

Keywords: existential termination, program analysis.

1 Introduction

The termination of logic programs became, in the last few years, an interesting research topic. The theories and techniques developed for studying the termination in imperative languages and term rewriting systems were used first. New termination techniques specific to logic programs have then been developed. In logic programs, we can make a distinction between universal and existential termination. A program is universally terminating for a given goal if the computation produces all the solutions and then terminates. It is existentially terminating for a goal if it either finitely fails or returns at least one solution.

Much work has been recently devoted to universal termination. For instance [1, 2, 3, 4] suggest new theoretical characterizations. In addition there are many proposals of effective methods, such as those in [12, 14, 15, 16], which infer interargument relations using AND/OR Dataflow Graphs, and those in [6, 7], where the termination is proved for programs enriched with assertions. Up to now, very little attention has been devoted to existential termination. [5] proposes a semantic approach using inductive proof techniques, [11] studies an effective method for function free programs, while [10, 19] just contain the basic definitions. Actually, whoever wants to develop an effective (and general enough) method for proving existential termination, has to deal with the impossibility of making any term abstraction, since existential termination does not enjoy any closure property, neither w.r.t. instantiation nor w.r.t. anti-instantiation. Moreover, the dependence on the computation rule forces us, in most cases, to choose a specific search strategy (usually the depth-first rule of PROLOG), which further complicates the analysis of the control because of the backtracking mechanism.

The development of existential termination techniques becomes essential in order to generalize the analysis of universal termination to normal programs. In fact, the problem of universal termination for negative goals boils down to the problem of existential termination for positive goals. An extension of the techniques for universal termination for dealing with negation is possible only if we have satisfactory results for existential termination.

In this paper we propose a new theoretical characterization of existential termination with the leftmost selection and depth-first computation rules. We also propose an abstract interpretation technique which can be useful to make our results applicable in practice.

In Section 2 we summarize the basic definitions and propose a technique of program indexing which will be used throughout the paper. In Section 3 we introduce occurrence sets, correct occurrence sets and their main properties, including an algebraic characterization in terms of complete lattices. We then introduce the minimal correct occurrence set which will be used in Section 4 to express our main results, which allow us to get rid of the occurrences in the occurrence sets, still preserving correctness and completeness w.r.t. existential termination, leading to the notion of semi occurrence sets of Section 5. Finally, in Section 6 we discuss an application of abstract interpretation which generates the semi occurrence sets and, in Section 7, show one simple example.

2 Preliminaries

2.1 Definitions

Let us first consider the classical definition of existential termination and some of its basic properties [5, 10]. The definition is based on the operational behavior of the program, using the standard concepts of computation and solution.

Definition 2.1 *A program p* **existentially terminates** *for the goal g if the computation either finitely fails or produces at least one solution.*

A basic property of existential termination is its dependence on both the selection rule (as in the case of universal termination) and the computation rule. Moreover, it does not enjoy the instantiation closure properties which are typical of universal termination. Namely the existential termination of a goal does not imply the termination of its instances. It is then necessary to study the existential termination in a goal-dependent way without the possibility of generalizing the results to instances or anti-instances [18].

In the discussion that follows, we shall consider logic programs with the PROLOG selection and search rules. We assume the reader to be familiar with the notions of SLD-resolution and SLD-tree. Moreover, we assume the concept of LD-resolution which uses the leftmost selection rule and the depth-first search rule. An LD-computation either terminates when it has found the first solution or diverges in the first infinite branch. An LD-tree is built by collecting from left to right the branches of the SLD-tree up to and including the first non finitely failed branch. Moreover, if the node n has associated the goal $A, \tilde{B}$, we define the selected atom in n as the atom A, according to the leftmost computation rule. For further definitions we refer to [13, 10].

$\langle p, g\rangle$ denotes a pair consisting of a definite program p and a goal g. Let I be a set of atoms. We denote by $up(I)$ the set of all the instances of atoms in I up to renaming. We denote by $\mathbb{N}^\infty$ the set $\mathbb{N} \setminus \{0\} \cup \{\infty\}$ obtained by extending the natural (positive) numbers and its ordering relation in the obvious way. Moreover, all our relations are defined up to renaming. The (atomic) symbol A will denote an atom or its equivalence class modulo renaming.

Existential termination can be expressed in terms of properties of LD-trees, as shown by the following proposition.

Proposition 2.2 *$\langle p, g\rangle$ existentially terminates if and only if the LD-tree of $\langle p, g\rangle$ is finite.*

2.2 Program indexing

A partial computation can be viewed as a sequence of goals $g_0 \to g_1 \to \ldots \to g_n$, where g_{i+1} is obtained from g_i and a (renamed apart) clause $c \equiv H : -A_1, \ldots, A_n$ by a single derivation step. In the approach we propose, it is necessary to identify the clause used in the resolvent in order to know which atoms in the program cause non-termination. In particular, for a goal $g = \leftarrow B_1, \ldots, B_n$, we want to know which clause the selected atom B_1 comes from. To this aim we propose a simple technique of program indexing. Obviously an atom in the head of a clause can never appear in a partial computation. Therefore only atoms in the body and in the goal need to be indexed.

Given a pair $\langle p, g\rangle$, the indexed program and goal is the pair $\langle p', g'\rangle$ obtained from $\langle p, g\rangle$, by indexing all the atoms in the body of any clause in p and in the goal g, so that all the indices are distinct. For the sake of simplicity, we shall use the natural numbers as index base. We associate then to every pair $\langle p, g\rangle$ a set of atoms $Atom_{\langle p,g\rangle}$ containing all the indexed atoms which are in p and in g.

Example 2.3 *Let p be the following program:*

```
q(a).

q(s) :- q(t),p(u).
```

and $g = \leftarrow q(x), p(y)$ be a goal. We define the program p'

```
q(a).

q(s) :- q₁(t),p₂(u).
```

the goal $g' = \leftarrow q_3(x), p_4(y)$ and $Atom_{\langle p,g\rangle} = \{q_1(t), p_2(u), q_3(x), p_4(y)\}$.

In the following, logic programs and goals will be indexed logic programs and indexed goals. The usual operations on atoms are extended in the obvious way: the mgu definition is not affected by the indexing, while the application of a substitution to an atom preserves the index of the atom itself. This extension aims to guarantee that every atom selected in the computation has an index. Therefore we define:

$$mgu(p_n(\tilde{t}), p_m(\tilde{u})) \stackrel{def}{=} mgu(p(\tilde{t}), p(\tilde{u}))$$

$$(p_n(\tilde{t}))\theta \stackrel{def}{=} p_n(\tilde{t}\theta)$$

We need to compare different atoms obtained as instances of the same atom in the program. Therefore we introduce a new equivalence relation which partitions the set of atoms on the basis of the associated index.

$$A \sim_{ind} B \stackrel{def}{\Leftrightarrow} A = p_n(\tilde{t}) \wedge B = p_n(\tilde{u})$$

Let A be an atom which is an instance of an atom in $Atom_{\langle p,g\rangle}$. We denote by $Atom_{\langle p,g\rangle}(A)$ the only element belonging to $Atom_{\langle p,g\rangle} \cap \{ B \mid B \sim_{ind} A \}$.

3 Occurrence sets and correct occurrence sets

The main idea underlying our construction is that one must be able to recognize the termination of a goal, by comparing an occurrence set for that goal with a specific partial computation. In practice, occurrence sets are used to guarantee the effectiveness of the analysis, allowing to compare the already visited atoms on the basis of the associated index. Therefore, our occurrence sets must contain both the visited atoms and their occurrences (i.e. how many times they occur in the computation).

We start by defining the domain of occurrence sets for a pair $\langle p, g\rangle$ and then stating the basic properties of correct occurrence sets. An occurrence set for $\langle p, g\rangle$ is a function which assigns to every atom A in $Atom_{\langle p,g\rangle}$ a set of pairs of the form $\langle A\theta, n\rangle$, where $n \in \mathbb{N}^\infty$. The second argument is called occurrence of the instance and can assume the value ∞.[1]

Definition 3.1 *We define* **occurrence set** *for* $\langle p, g\rangle$ *a function*

$$I_{\langle p,g\rangle} : Atom_{\langle p,g\rangle} \rightarrow 2^{up(Atom_{\langle p,g\rangle})\times\mathbb{N}^\infty}$$

such that $\forall A \in Atom_{\langle p,g\rangle}\ \{\, B \mid \langle B, n\rangle \in I_{\langle p,g\rangle}(A)\,\} \subseteq up(A)$.

Therefore, an occurrence set can be thought of as a function whose image is a multiset, and likewise we can define the notion of correct occurrence set. The idea behind the concept of correct occurrence set is to collect a superset of the atoms selected in a (leftmost depth-first) derivation, together with an upper approximation of the occurrences.

Definition 3.2 *Let* $I_{\langle p,g\rangle}$ *be an occurrence set for* $\langle p, g\rangle$. *Then* $I_{\langle p,g\rangle}$ *is* **correct** *for* $\langle p, g\rangle$ *if and only if:*

$\forall A \in Atom_{\langle p,g\rangle}\ \forall\theta$ *if* $A\theta$ *is selected* n *times in the LD-tree of* $\langle p, g\rangle$, *then* $\exists m \geq n$ *s.t.* $\langle A\theta, m\rangle \in I_{\langle p,g\rangle}(A)$.[2]

Note that multiple occurrences of predicates in $Atom_{\langle p,g\rangle}$ are not identified. Every atom in the LD-tree comes from a specific atom in $Atom_{\langle p,g\rangle}$ and we can easily identify that atom thanks to the indexing.

Proposition 3.3 *Every pair* $\langle p, g\rangle$ *has a correct occurrence set.*

Proof: The occurrence set
$\forall A \in Atom_{\langle p,g\rangle}\ I(A) = up(Atom_{\langle p,g\rangle}) \times \{\infty\}$
is correct for any program and goal. ■

Among the occurrence sets, we are particularly interested in finite and weakly finite occurrence sets and in completions, in order to capture the concept of existential termination.

Informally, an occurrence set is finite if the set of occurrences of atoms in its image is finite. By weakening the definition, we obtain the characterization of weakly finite occurrence set. Namely we simply count the atoms in the image, by ignoring multiple occurrences (or better by assuming all the occurrences to be unary). On the contrary, the completion is obtained by increasing the size of the occurrence set by setting all the occurrences to infinity.

Definition 3.4 *Let* $I_{\langle p,g\rangle}$ *be an occurrence set. Then it is* **finite** *if the summation*

$$\sum_{A\in Atom_{\langle p,g\rangle}} \ \sum_{\langle B,n\rangle\in I_{\langle p,g\rangle}(A)} n$$

is finite. Likewise, an occurrence set is **weakly finite** *if*

$$\sum_{A\in Atom_{\langle p,g\rangle}} \ \sum_{\langle B,n\rangle\in I_{\langle p,g\rangle}(A)} 1$$

is finite.

---

[1] We prefer to use the notation $\langle A\theta, n\rangle$ rather than $\langle\theta, n\rangle$ in order to simplify the definition of equivalence up to renaming. Consider the atom $A = p(X,Y)$ and the substitutions $\theta = \{X \leftarrow f(Z)\}$, $\sigma = \{X \leftarrow f(W)\}$ and $\gamma = \{X \leftarrow f(Y)\}$. It turns out that $A\theta$ is equivalent to $A\sigma$ up to renaming but it is not equivalent to $A\gamma$, i.e. the renaming equivalence depends on the atom to which we apply the substitution. Therefore it is an equivalence relation which is parametric w.r.t. atoms.

[2] Since the relations are up to renaming, actually the definition should be: $\forall A \in Atom_{\langle p,g\rangle}\ \forall\theta$ if $A\theta$ appears n times (up to renaming) in the LD-tree of $\langle p, g\rangle$, then $\exists m \geq n$ s.t. $\langle B, m\rangle \in I_{\langle p,g\rangle}(A)$ where $A\theta$ is equivalent up to renaming to B.

Definition 3.5 *Let $I_{\langle p,g\rangle}$ be a weakly finite occurrence set. We call* **completion** *the occurrence set $Comp(I_{\langle p,g\rangle})$ defined as follows.*

$$Comp(I_{\langle p,g\rangle})(A) = \{\langle B, \infty\rangle \mid \langle B, n\rangle \in I_{\langle p,g\rangle}(A)\} \ \forall A \in Atom_{\langle p,g\rangle}.$$

Proposition 3.6 *The completion of a weakly finite occurrence set is weakly finite.*

Our characterization of termination is directly obtained by using the above definitions.

Theorem 3.7 *$\langle p, g\rangle$ existentially terminates if and only if there exists a finite correct occurrence set of $\langle p, g\rangle$.*

3.1 The lattice of occurrence sets

We analyze now the structure of the occurrence set $Int_{\langle p,g\rangle}$ for the pair $\langle p, g\rangle$ in order to be able to introduce the notion of minimal correct occurrence set, which will be used to state our main results. Let us start by defining two new operations, i.e. join and meet on occurrence sets.

Definition 3.8 *Let $I, J \in Int_{\langle p,g\rangle}$ be two occurrence sets. We define a new occurrence set $(I \sqcup J)(A)$ for all $A \in Atom_{\langle p,g\rangle}$ as follows.*

$$\begin{aligned}(I \sqcup J)(A) \overset{def}{=} \{ \quad & \langle B, n\rangle \mid \\ & 1.\ \langle B, x\rangle \in I \wedge \langle B, y\rangle \in J \wedge n = max\{x, y\}, \\ & 2.\ \langle B, n\rangle \in I \wedge \forall y\ \nexists \langle B, y\rangle \in J, \\ & 3.\ \langle B, n\rangle \in J \wedge \forall y\ \nexists \langle B, y\rangle \in I\,\}.\end{aligned}$$

Analogously we define the meet $(I \sqcap J)(A)$ for all $A \in Atom_{\langle p,g\rangle}$:

$$(I \sqcap J)(A) \overset{def}{=} \{\,\langle B, n\rangle \mid \langle B, x\rangle \in I \wedge \langle B, y\rangle \in J \wedge n = min\{x, y\}\,\}.$$

$Int_{\langle p,g\rangle}$ can be partially ordered by using the join operation. Let $I, J \in Int_{\langle p,g\rangle}$ be two occurrence sets. We define:

$$I \sqsubseteq J \overset{def}{\Leftrightarrow} I \sqcup J = J.$$

We generalize the join and meet operations to sets of occurrence sets in the obvious way. By using the above ordering, we prove that $\langle Int_{\langle p,g\rangle}, \sqsubseteq, \sqcup, \sqcap\rangle$ is a complete lattice. By exploiting the lattice structure we are able to state the first important result.

Theorem 3.9 *The meet of all the correct occurrence sets for the pair $\langle p, g\rangle$ is a correct occurrence set for $\langle p, g\rangle$. Such a occurrence set is called the minimal correct occurrence set and is unique.*

For the sake of simplicity, we denote the minimal correct occurrence set for a pair $\langle p, g\rangle$ by $I^{min}_{\langle p,g\rangle}$.

Theorem 3.10 *An occurrence set $I_{\langle p,g\rangle}$ is correct for $\langle p, g\rangle$ if and only if $I^{min}_{\langle p,g\rangle} \sqsubseteq I_{\langle p,g\rangle}$.*

Lemma 3.11 *The completion of a weakly finite, correct occurrence set is weakly finite and correct.*

Theorem 3.10 gives a correct occurrence set characterization, by using the ordering on the occurrence sets lattice. However the resulting characterization is obviously not effective because of the two following facts:

1. not all the correct occurrence sets have a finite representation;
2. the computation of $I^{min}_{\langle p,g\rangle}$ is not effective.

In order to solve the first problem, we shall consider weakly finite occurrence sets (and therefore weakly finite, correct occurrence sets) only, since they always have an effective representation. The above constraint automatically solves also the second problem, since the condition $I^{min}_{\langle p,g\rangle} \sqsubseteq I_{\langle p,g\rangle}$ turns out to be decidable, as shown in detail in the next Sections.

4 Towards a decision procedure based on weakly finite occurrence sets

Dealing with occurrence sets is definitely very close to the original model (the LD-tree) and results in a very simple theory. Actually, in order to capture existential termination, we can ignore occurrences and use sets of instances only. This intuition leads to a remarkable simplification in the construction of the minimal correct occurrence set, since we do not need to worry about the derivations which use a given instance infinitely many times. The result which allows us to obtain this simplification is the following. Let A be an atom which is selected at least twice in the LD-tree of $\langle p, g\rangle$. By inspecting the (finite) partial computation, included between the first two selections of the atom A, we are able to decide whether A existentially terminates w.r.t. the program p.

Theorem 4.1 *Let $A \in Atom_{\langle p,g\rangle}$ be such that $\exists\langle A\theta, n\rangle \in I^{min}_{\langle p,g\rangle}(A)$ with $n > 1$.Then it is decidable whether $\langle p, A\theta\rangle$ existentially terminates.*

Proof: By making a depth-first traversal of the LD-tree, let n_1 and n_2 be the first and the second node labeled $A\theta, \tilde{B}$ and $A\theta, \tilde{C}$ respectively (such nodes do exist, because $n \geq 2$ by hypothesis). We take under consideration the branch of the LD-tree starting from the root to the node n_2. We have two cases:

1. the node n_1 belongs to such a branch. This means that the goal $A\theta, \tilde{B}$ is rewritten to $A\theta, \tilde{C}$. We have again two cases:

   i the resolution of the atom $A\theta$ terminates with computed answer solution σ and the computation continues with the new goal $\tilde{B}\sigma$ (and $\tilde{B}\sigma$ is rewritten to $A\theta, \tilde{C}$). In this case $A\theta$ successfully existentially terminates.

   ii the goal $A\theta$ is rewritten to the goal $A\theta, \tilde{D}$ and therefore $A\theta$ diverges.

   Note that it is decidable whether the computation associated to $A\theta$ terminates, since we have to analyze a finite number of steps (i.e. the nodes included between n_1 and n_2).

2. the node n_1 does not belong to such a branch. This means that it necessarily belongs to a previously visited and finitely failed branch. Therefore the goal $A\theta$ either finitely fails or successfully terminates.

Since it is decidable whether n_1 belongs to a finite branch, the existential termination of $A\theta$ is decidable too. ■

By using the previous result we can characterize, in effective way, whether the completion of a weakly finite occurrence set represents a correct occurrence set.

Theorem 4.2 *Let $I_{\langle p,g\rangle}$ be a weakly finite occurrence set. Then it is decidable whether $I_C = Comp(I_{\langle p,g\rangle})$ is a correct occurrence set for $\langle p, g\rangle$, i.e. whether $I^{min}_{\langle p,g\rangle} \sqsubseteq Comp(I_{\langle p,g\rangle})$.*

Proof: The proof is given by the following algorithm (where Visit and Term are two sets of atoms).

let $\langle D, \tilde{E}\rangle$ be the label of the root of the LD-tree of $\langle p, g\rangle$,
  $f(\langle A, \tilde{C}\rangle, Visit, Term) =$
  let $B = Atom_{\langle p,g\rangle}(A)$
  in if $\langle A, \infty\rangle \notin I_C(B)$
    then $\langle I_C$ is not a correct occurrence set,a$\rangle$
    else **if** $A \in Visit \setminus Term$ and A does not existentially terminate
      then $\langle I_C$ is a correct occurrence set,b$\rangle$

else if the depth-first traversal of the LD-tree of $\langle p, g\rangle$
is terminated
then $\langle I_C$ is a correct occurrence set,c$\rangle$
else let $\langle A', \tilde{C'}\rangle$ be the label of the next selected node
in $f(\langle A', \tilde{C'}\rangle, Visit \cup \{A\}, Term \cup (\{A\} \cap Visit))$
in $f(\langle D, \tilde{E}\rangle, \emptyset, \emptyset)$.

In order to prove the correctness of the algorithm we have to show that:

i. all the choices are decidable.

- $\langle A, \infty\rangle \notin I_C(B)$. Since I_C is weakly finite, $I_C(B)$ is finite and therefore the condition turns out to be decidable.
- A does not existentially terminate. We test this condition only after having checked that $A \in Visit \setminus Term$, i.e. that A has already occurred in the visited subtree. Thus, A occurs at least twice in the minimal correct occurrence set of $\langle p, g\rangle$ and therefore, by theorem 4.1, it is decidable whether A existentially terminates.

ii. the algorithm always terminates. In the algorithm we check whether the selected atom belongs to the set $I_C(B)$. By definition of weakly finite occurrence set, the set $\{ B \mid \exists A \in Atom_{\langle p,g\rangle} \langle B, n\rangle \in I_C(A) \}$ is finite. It follows that the algorithm considers finitely many distinct atoms. We have only to prove that no atom is selected infinitely many times. We assume, by contradiction, that there exists a set of atoms which are selected infinitely many times. This implies that at least one of these atoms recursively calls itself and therefore the atom does not existentially terminate. But this is a contradiction, since the algorithm should terminate with label b.

iii. it returns correct answers. It is worth noting that the sets $Visit$ and $Term$ contain, respectively, the already selected atoms and, among those which have been selected at least twice, those which existentially terminate. If the algorithm terminates with label a, I_C cannot be a correct occurrence set, because it does not contain the selected atom. If it terminates with label c, the complete LD-tree has been visited and all the atoms belong to I_C, and therefore we can assert that it is a correct occurrence set. The interesting case is the exit with label b. In this case there exists an atom A which has been visited at least twice and does not existentially terminate. Theorem 4.1 (part 1.ii.) states that A is rewritten to itself, and therefore the computation goes into an infinite loop where the selected atoms of the visited goals are all and only the atoms included between the atom A and its next invocation. Thus I_C is a correct occurrence set since it contains all these atoms with occurrence ∞.

■

Some remarks about the above theorems are necessary. The above results state that, for a weakly finite occurrence set, we can decide whether its completion is correct. Furthermore, we have proved that if a weakly finite occurrence set is correct, then, by theorem 3.10, also the completion is correct, but the vice versa is not valid. This means that, when moving from an occurrence set to its completion, the chances to obtain a correct occurrence set do increase. We can thus get rid of weakly finite occurrence sets and take into consideration their completions only, since we know that we have more chances to find a correct occurrence set in this class of occurrence sets. In practice we can neglect the occurrences and consider weakly finite occurrence sets only, without caring about atoms with multiple occurrences. This is captured by the notion of semi occurrence set.

5 Semi occurrence sets

Definition 5.1 *A* **semi occurrence set** *for $\langle p, g\rangle$ is a function*

$$I_{\langle p,g\rangle} : Atom_{\langle p,g\rangle} \rightarrow 2^{up(Atom_{\langle p,g\rangle})},$$

with the constraint that $\forall A \in Atom_{\langle p,g\rangle}\ I_{\langle p,g\rangle}(A) \subseteq up(A)$ and finite.

Note that the set of semi occurrence sets and the set of completions are isomorphic. Therefore we can embed the structure of semi occurrence sets in the lattice of occurrence sets. To this aim we define the extension of a semi occurrence set as the occurrence set obtained by setting all the occurrences to infinity.

Definition 5.2 *Let I be a semi occurrence set for $\langle p, g\rangle$. The occurrence set $Ext(I)$ is defined as:*

$$\forall A \in Atom_{\langle p,g\rangle}\ Ext(I)(A) \stackrel{def}{=} \{\, \langle B, \infty\rangle \mid B \in I(A)\,\}.$$

By exploiting the concept of semi occurrence set, theorem 4.2 becomes:

Theorem 5.3 *Let $I_{\langle p,g\rangle}$ be a semi occurrence set. Then it is decidable whether $Ext(I_{\langle p,g\rangle})$ is a correct occurrence set for $\langle p, g\rangle$, i.e. whether $I^{min}_{\langle p,g\rangle} \sqsubseteq Ext(I_{\langle p,g\rangle})$.*

Our task is therefore to provide a semi occurrence set and, by using the above theorems and algorithms, to check whether the correct occurrence set is finite and therefore to infer whether $\langle p, g\rangle$ existentially terminates.

In order to achieve our aim, we only have to check whether the finiteness of weakly finite, correct occurrence set is decidable. The characterization we obtain is still stronger and allows us to decide, for a weakly finite, correct occurrence set, whether the minimal correct occurrence set of $\langle p, g\rangle$ is finite, and therefore if there exists a finite correct occurrence set. The next theorem guarantees the effectiveness of such a check by using the previously specified algorithm.

Theorem 5.4 *Let $I_{\langle p,g\rangle}$ be a semi occurrence set such that $Ext(I_{\langle p,g\rangle})$ is a correct occurrence set for $\langle p, g\rangle$. Then it is decidable whether $I^{min}_{\langle p,g\rangle}$ is finite.*

Proof: The proof is based on the algorithm given in theorem 4.2. We use such an algorithm and the occurrence set $Ext(I_{\langle p,g\rangle})$. Since the occurrence set is correct, it turns out that the algorithm terminates either with label b or c. If the algorithm terminates with label b, we may conclude that $I^{min}_{\langle p,g\rangle}$ can not be finite since it contains at least an atom which does not existentially terminate. If the algorithm terminates with label c, the minimal correct occurrence set is finite since the LD-tree of $\langle p, g\rangle$ is finite. ■

6 Generation of the semi occurrence set

As a conclusion, we suggest a method to automatically generate the semi occurrence sets. The basic idea is to generate an approximation of the atoms selected in the LD-tree, by a refinement of the depth-k abstraction ([17]) used for program analysis based on abstract interpretation ([8, 9]). Since existential termination does not enjoy any closure property, we are forced to compute an abstract denotation depending on both the program and the goal. To this end, we shall use a (leftmost) top-down construction based on the depth-first search rule. We start defining some useful functions and the abstract domain used by the fixpoint operator.

Let us consider the function $|\ldots| : Term \rightarrow \mathbb{N}$, such that

$$|t| = \begin{cases} 1 & \text{if } t \text{ is a constant or a variable} \\ max\{|t_1|, \ldots, |t_n|\} + 1 & \text{if } t = f(t_1, \ldots, t_n) \end{cases}$$

and a given positive integer k. The associated equivalence relation is $\sim_k$, such that $t_1 \sim_k t_2$ iff $\alpha_k(t_1) = \alpha_k(t_2)$, where $\alpha_k(t)$ represents the term which can be obtained from the concrete one by substituting with a fresh variable each subterm t' in t, such that $|t| - |t'| = k$.

We can define the abstract universe as the image of α_k, that is $Term^a = \alpha_k(Term) = \{[t]_{\sim_k} : t \in Term\}$ and the abstract base $Atom^a = \{p(t_1^a, \ldots, t_n^a) : p \in \Pi, t_i^a \in Term^a\}$. The abstract semantics is defined as a set of pairs of the form $\langle g, h \rangle$, where g is the current goal and h is the sequence of renamed apart clauses used in a partial (leftmost depth-first) derivation starting from g. We shall refer to h as the history of the (partial) derivation. It follows that the abstract interpretation domain is simply $\mathcal{P}((Atom^a)^* \times Clause^*)$.

The next step concerns the abstract domain ordering. First of all, we define the structure of the program and the ordering on sequences of clauses. A program p is defined as a sequence of clauses $p = c_1 :: \ldots :: c_n$. The ordering on the clauses reflects the position in the program, i.e. $\forall i, j \; c_i < c_j \Leftrightarrow i < j$. We extend the ordering on sequences of clauses. Given a clause $c \in Clause$ and $s, s', s'' \in Clause^*$ we define

$$\begin{array}{rl} c :: s < s' \Leftrightarrow & 1)\; s' = \lambda \text{ or} \\ & 2)\; s' = d :: s'' \text{ and } c < d \\ & 3)\; s' = d :: s'' \text{ and } c = d \text{ and } s < s''. \end{array}$$

Note that the empty sequence λ is the greatest element. We order the pairs $\langle g, h \rangle$ according to the ordering on the sequences, i.e. by comparing only the histories of the derivation: $\langle g, h \rangle < \langle g', h' \rangle \Leftrightarrow h < h'$. This allows us to implement exactly the depth-first search strategy.

The abstraction function α_k and the equivalence relation $\sim_k$ are extended on $Atom$ and $Atom^*$ in the obvious way, that is $\alpha_k(p(t_1, \ldots, t_n)) = p(\alpha_k(t_1), \ldots, \alpha_k(t_n))$ and $A_1, \ldots, A_n \sim_k B_1, \ldots, B_n \Leftrightarrow \forall i = 1 \ldots n \; \alpha_k(A_i) \sim_k \alpha_k(B_i)$.

We define the equivalence relation $\sim_k$ on pairs, in such a way that two pairs are in $\sim_k$-relation if and only if the selected atoms in the goals are in the same relation.

$$\langle g, h \rangle \sim_k \langle g', h' \rangle \Leftrightarrow (g = A, \tilde{A} \text{ and } g' = B, \tilde{B} \text{ and } A \sim_k B) \text{ or } g = g' = \lambda.$$

It is straightforward to extend α_k on the abstract domain and to define a corresponding Galois insertion $\langle \alpha_k, \gamma_k \rangle$ such that $\gamma_k(I) = \{A \in Atom : \alpha_k(A) \in I\}$.[3]

The next step concerns the abstract fixpoint operator. First of all, we define a function

$$SLD_{Left} : \mathcal{P}((Atom)^* \times Clause^*) \rightarrow \mathcal{P}((Atom)^* \times Clause^*)$$

which computes all the pairs reachable by one step of SLD-resolution via the leftmost selection rule.

$$\begin{array}{rl} SLD_{Left}(I) = \{\langle (\tilde{B}, \tilde{A})\theta; h :: c \rangle \mid & \langle A, \tilde{A}; h \rangle \in I \\ & c \equiv H :- \tilde{B} \in p \\ & \theta = mgu(A, H)\}. \end{array}$$

In order to recognize successful computations, we define the function Id_λ which returns the pairs with empty goal: $Id_\lambda(I) = \{\langle \lambda; h \rangle \mid \langle \lambda; h \rangle \in I\}$.

Finally, the abstract semantics is obtained by computing, at each step, the set $SLD_{Left}(I)$ minus the pairs already computed in the previous steps, (possibly) augmented with a success pair. The least element of this set, when it exists, is exactly the result obtained by one step of SLD-resolution via the leftmost selection and depth-first search rules. The definition of the $T_{p,g}$ become:

[3] Note that we do not need to prove that the abstract semantics is correct w.r.t. a suitable concrete one, just because our semi occurrence sets are not necessarily correct occurrence sets. However, one can easily prove that our abstract semantics is a Galois insertion of the more concrete one, which can be obtained as a limit with $k = \infty$.

$$T_{p,g}(I) = I \cup min((SLD_{Left}(I) \setminus I) \cup Id_\lambda(I))$$

where $min(\emptyset) = \emptyset$[4]. Using a top-down construction, we have to start with the goal g and history λ, just because in the sequence ordering the empty history represents the top element. The least fixpoint is then computed in the following way:

$$T_{p,g} \uparrow_k 0 = \{\langle g, \lambda\rangle\}$$

$$T_{p,g} \uparrow_k (n+1) = \begin{cases} \alpha_k(T_{p,g} \uparrow_k n) & \text{if } T_{p,g}(T_{p,g} \uparrow_k n) \sim_k T_{p,g} \uparrow_k n \\ T_{p,g}(T_{p,g} \uparrow_k n) & \text{otherwise,} \end{cases}$$

where $\alpha_k(I) = \{\langle \alpha_k(g), h\rangle : \langle g, h\rangle \in I\}$ and $I \sim_k J \Leftrightarrow \forall p \in I\ \exists q \in J\ p \sim_k q\ and\ \forall p \in J\ \exists q \in I\ p \sim_k q$.

Note that the $T_{p,g} \uparrow_k$ operator always terminates just because the partition induced by the $\sim_k$-relation on the abstract domain is finite.

Given the above abstract semantics, we extract the semi occurrence set $I_{\langle p,g\rangle}$, by simply collecting all the selected atoms and getting rid of the history of the derivation, i.e.

$$I_{\langle p,g\rangle} = \{A \mid \langle A, \tilde{A}; h\rangle \in T_{p,g} \uparrow_k \omega\}.$$

7 Example

Let p be the following program:

```
visit([1|X]).

visit([0|Y]) ← append₁(Y,[0],Z),visit₂(Z).

append([],S,S).

append([T|U],V,[T|W]) ← append₃(U,V,W).
```

The behavior of the program w.r.t. a given goal $visit(t)$ is the following:

1. if $t = [t_1, \ldots, t_n \mid X]$ then $visit(t)$:

   i infinitely fails $\Leftrightarrow \forall 1 \leq i \leq n\ t_i = 0$;

   ii finitely fails $\Leftrightarrow \exists 1 \leq j \leq n\ \forall 1 \leq i < j\ t_i = 0$ and $t_j \neq 0, 1$ and t_j is not a variable;

   ii succeeds $\Leftrightarrow \exists 1 \leq j \leq n\ \forall 1 \leq i < j\ t_i = 0$ and ($t_j = 1$ or t_j is a variable);

2. $visit(t)$ finitely fails otherwise.

We choose $k = min\{i \mid p \sim_k \alpha_k(p)$ and $g \sim_k \alpha_k(g)\}$. The semi occurrence set extracted from $T_{p,g} \uparrow_k \omega$ is powerful enough to allow the algorithm to state the (non-)termination for every goal[5].

[4] Note that $<$ is a total order on $\mathcal{P}((Atom^a)^* \times Clause^*)$ and therefore min is always defined.

[5] It is worth noting that, in some cases, the technique is able to decide about the non-termination of a goal. In fact, when the algorithm terminates with label b, we can infer that the minimal correct occurrence set is infinite and therefore that the goal does not existentially terminate.

8 Conclusions

Our theory for capturing the notion of existential termination applies to positive logic programs, under the leftmost and depth-first rules and, according to the well-known properties of termination, can be applied to study the universal termination of negative goals in normal logic programs.

An interesting extension is related to the possibility of using different selection rules. In fact, the choice of the selection rule affects only the notion of LD-tree, while all our results are still valid. On the contrary, the choice of a different search rule, would affect the validity of theorem 4.1, on which our technique is based. Nevertheless, we believe that our result can be generalized to any search rule.

As a final remark, it is worth noting that our results can be applied in a "reverse" mode. Namely, in order to prove that $\langle p, A\rangle$ existentially terminates, we can look for a goal g such that the minimal correct occurrence set of $\langle p, g\rangle$ contains the atom A with occurrence ≥ 2. In such a way, we could reduce the problem of existential termination of A to the search of a goal with given properties.

References

[1] K. R. Apt and D. Pedreschi. Studies in Pure Prolog: Termination. In J. W. Lloyd, editor, *Computational Logic*, pages 150–176. Springer-Verlag, Berlin, 1990.

[2] K. R. Apt and D. Pedreschi. Proving Termination of General Prolog Programs. In T. Ito and A.R. Meyer, editors, *Proc. of Int. Conf. on Theoretical Aspects of Computer Software*, volume 526 of *LNCS*, pages 265–289. Springer-Verlag, Berlin, 1991.

[3] K.R. Apt and D. Pedreschi. Reasoning about Termination of Prolog Programs. Technical Report TR-14/91, Dip. di Informatica, Univ. di Pisa, 1991.

[4] K.R. Apt and D. Pedreschi. Modular Termination Proofs for Logic and Pure Prolog Programs. In G. Levi, editor, *Advances in Logic Programming Theory*, pages 183–229. Clarendon Press - Oxford, 1994.

[5] M. Baudinet. Proving Termination Properties of Prolog Programs: A Semantic Approach. *Journal of Logic Programming*, 14:1–29, 1992.

[6] A. Bossi, N. Cocco, and M. Fabris. Proving Termination of Logic Programs by Exploiting Term Properties. In S. Abramsky and T.S.E. Maibaum, editors, *Proc. TAPSOFT'91*, volume 494 of *LNCS*, pages 153–180. Springer-Verlag, Berlin, 1991.

[7] A. Bossi, N. Cocco, and M. Fabris. Norms on Terms and Their Use in Proving Universal Termination of a Logic Programs. *TCS*, 124(2):297–328, 1994.

[8] P. Cousot and R. Cousot. Abstract Interpretation: A Unified Lattice Model for Static Analysis of Programs by Construction or Approximation of Fixpoints. In *Proc. Fourth ACM Symp. Principles of Programming Languages*, pages 238–252, 1977.

[9] P. Cousot and R. Cousot. Systematic Design of Program Analysis Frameworks. In *Proc. Sixth ACM Symp. Principles of Programming Languages*, pages 269–282, 1979.

[10] D. De Schreye and S. Decorte. Termination of Logic Programs: the Never-Ending Story. *Journal of Logic Programming*, 19/20:199–260, 1994.

[11] N. Francez, O. Grumberg, S. Katz, and A. Pnueli. Proving Termination of Prolog Programs. In R. Parikh, editor, *Logics of Programs*, volume 193 of *LNCS*, pages 89–105. Springer-Verlag, 1985.

[12] G. Gröger and L. Plümer. Handling of Mutual Recursion in Automatic Termination Proofs for Logic Programs. Unpublished draft.

[13] J. W. Lloyd. *Foundations of Logic Programming.* Springer-Verlag, Berlin, 1987.

[14] L. Plümer. *Termination Proofs for Logic Programs*, volume 446 of *Lecture Notes in Artificial Intelligence.* Springer-Verlag, Berlin, 1990.

[15] L. Plümer. Termination Proofs for Logic Programs based on Predicate Inequalities. In D. H. D. Warren and P. Szeredi, editors, *Proc. Seventh ICLP*, pages 634–648. The MIT Press, 1990.

[16] L. Plümer. Automatic Termination Proofs for Prolog Programs Operating on Nonground Terms. In K. Furukawa, editor, *Proc. Eighth Int'l Conf. on Logic Programming*, pages 503–517. The MIT Press, 1991.

[17] T. Sato and H. Tamaki. Enumeration of Success Patterns in Logic Programs. *Theoretical Computer Science*, 34:227–240, 1984.

[18] F. Scozzari. Analisi di terminazione mediante interpretazione astratta. Master's thesis, Dipartimento di Informatica, Università di Pisa, 1994. in italian.

[19] T. Vasak and J. Potter. Characterization of Terminating Logic Programs. In *Proc. Third IEEE ILPS*, pages 140–147. IEEE Comp. Soc. Press, 1986.