∃-Universal termination of logic programs
Salvatore Ruggieri
Dipartimento di Informatica, Università di Pisa, Corso Italia 40, 56125 Pisa, Italy
Received February 1998; revised March 1999
Communicated by G. Levi

Abstract

We introduce the notion of ∃-universal termination of logic programs. A program P and a goal G ∃-universally terminate iff there exists a selection rule $S$ such that every SLD-derivation of $P \cup \{G\}$ via $S$ is finite. We claim that it is an essential concept for declarative programming, where a crucial point is to associate a terminating control strategy to programs and goals. We show that ∃-universal termination and universal termination via fair selection rules coincide. Then we offer a declarative characterization of ∃-universal termination by defining fair-bounded programs and goals. They provide us with a correct and complete method for proving ∃-universal termination. We show other valuable properties of fair-boundedness, including persistency, modularity, ease of use in paper & pencil proofs, automation of proofs. © 2001 Elsevier Science B.V. All rights reserved.

Keywords: Logic programs; Universal termination; Fair selection rules; Fair-bounded programs

1. Introduction

Logic programming is advocated as an ideal support for declarative programming. The milestone work of Kowalski [9] foresees a separation of concerns between the logic and the control components of programs. Logic is demanded to programmers, who write specifications that can be directly used as programs. The generation of a complete control is demanded to the underlying logic programming system. By a complete control, one usually means a selection rule $S$ such that every logical consequence of a program has a refutation via $S$. By the strong completeness theorem of SLD-resolution (see Apt [1]), any selection rule is complete in this sense. However, a stronger form of completeness is usually intended, which takes into account termination as well.

Definition 1.1. By a complete control for a program P and a goal G, we mean any selection rule $S$ such that every SLD-derivation of $P \cup \{G\}$ via $S$ is finite.
The ideal situation where logic and control are kept separated is usually contradicted by practical experience. For efficiency reasons, early systems such as Prolog adopted a fixed control, namely a left-to-right selection rule and a depth-first search strategy. Unfortunately, Prolog's control is not complete in the sense of Definition 1.1. This fact prevents writing programs declaratively, and, in practice, a left-to-right style is customary in any Prolog program. This and other problems with Prolog semantics (including the omission of the occur-check, the use of unsafe negation, the presence of extra-logical predicates, the undisciplined use of cut) motivated the design of more high level, expressive, efficient and practical logic programming systems.

Second generation logic languages adopt more flexible control primitives, which allow for addressing logic and control separately. Program clauses are intended to model the logic of programs, as usual. In addition, programs are augmented by declarations or annotations that (implicitly or explicitly) specify restrictions on the admissible selection rules. In this class of languages, we include NU-Prolog [22], Gödel [8] and the Mercury [19] system, among the others. The trend in logic language design is then to achieve full declarativeness by forbidding the use of extra-logical predicates. Real world application problems are tackled providing facilities including type checking, meta-programming, higher order programming, declarative input/output, libraries and modules. At the same time, efficiency is improved by exploiting the large amount of techniques available for code optimizations, coroutining, parallel executions, deterministic executions, safe omission of the occur-check etc.

In this context, the problem of characterizing classes of terminating programs and goals is essential:

(i) to gain a precise understanding of the class of programs and goals that have a complete control, in the sense of Definition 1.1, or such that a given selection rule is for them a complete control;
(ii) to provide support for paper & pencil verification of termination properties;
(iii) to serve as a theoretical framework on which the design of automatic tools for termination analysis, compiler optimizations and program transformations can be based.

Unfortunately, most existing termination methods (see [6] for a survey) are restricted to reason only on termination via the leftmost selection rule.

In this paper, we introduce and investigate ∃-universal termination of logic programs and goals. We claim that ∃-universal termination is an essential concept for the concerns of separating the development of the logic part of programs from the problem of associating a complete control strategy to them.

**Definition 1.2.** A logic program \( P \) and a goal \( G \) \( ∃ \)-universally terminate iff there exists a selection rule \( S \) such that every SLD-derivation of \( P \cup \{G\} \) via \( S \) is finite.

Since SLD-trees are finitely branching, by König's Lemma, we have that \( P \) and \( G \) \( ∃ \)-universally terminate iff there exists a finite SLD-tree for \( P \cup \{G\} \). If \( P \) and \( G \)
∃-universally terminate then it is possible, at least in principle, to associate to them a complete control, in the sense of Definition 1.1. On the other hand, if $P$ and $G$ does not ∃-universally terminate, then no logic programming system can be complete in the sense of Definition 1.1. Nevertheless, completeness can still be achieved by adopting transformational techniques, or loop checking mechanisms, or, more in general, by modifying the operational interpretation of $P$ and $G$ provided by SLD-resolution.

In this paper, we show that ∃-universal termination coincides with universal termination with respect to fair selection rules. Therefore, fair selection rules are a complete control for every logic program and goal for which a complete control exists, in the sense of Definition 1.1. Then we offer a characterization of ∃-universal termination by means of the notions of fair-bounded programs and goals, that provide us with a correct and complete method for proving ∃-universal termination. The definition of fair-boundedness is purely declarative, in the sense that neither any procedural notion is needed in order to prove a program fair-bounded nor the definition reflects some fixed ordering of the atoms. Proof obligations are modular, i.e. each program clause has to be considered in isolation. Moreover, we argue that the method is simple and intuitive in paper & pencil proofs. Finally, as a by-result of completeness of the method, we observe that any (possibly automatic) method for proving termination via any selection rule is a sufficient condition for proving a program fair-bounded. This allows us to reuse all existing automatic tools and termination proofs to the purpose of showing fair-boundedness.

1.1. Plan of the paper

After some preliminaries, we show in Section 2 that ∃-universal termination and universal termination via fair selection rules coincide. In Section 3, the definitions of fair-bounded logic programs and goals are introduced. In Sections 4 and 5, we show persistency, correctness and completeness of the proof method. In Section 6, the approach is extended to reason on arithmetic built-in’s. In Section 7, well-studied classes of terminating logic programs are related to fair-bounded programs. In Section 8, we discuss the problem of automatically inferring fair-boundedness. Finally, in Section 9 we recall some recent works reasoning about termination via selection rules other than the leftmost one.

1.2. Preliminaries

Throughout the paper we follow the standard notation of Apt [1]. A clause is a construct $A \leftarrow B_1, \ldots, B_n$, with $n \geq 0$, where $A$ and $B_1, \ldots, B_n$ are atoms. A logic program is a finite set of clauses. A goal is a construct $\leftarrow B_1, \ldots, B_n$, where $B_1, \ldots, B_n$ are atoms. $B_P$ the Herbrand base on $L_P$, the language generated by $P$. $U_P$ is the set of ground terms on $L_P$, i.e. the Herbrand universe. A Herbrand interpretation $I$ is a subset of $B_P$. We recall that for ground atoms $A_1, \ldots, A_n$, $I \models A_1, \ldots, A_n$ holds iff $\{A_1, \ldots, A_n\} \subseteq I$. Moreover, for (not necessarily ground) atoms $A_1, \ldots, A_n$, $I \models \exists A_1, \ldots, A_n$ holds iff there exists a ground instance $A'_1, \ldots, A'_n$ of $A_1, \ldots, A_n$ such that $I \models A'_1, \ldots, A'_n$. $\text{ground}(P)$
denotes the set of ground instances of clauses from $P$. Analogously, $\text{ground}(G)$ denotes the set of ground instances of a goal $G$. $[A]$ denotes the set of ground instances of the atom $A$. $T_P$ is the immediate consequence operator defined as follows: $T_P(I) = \{ A \mid A \leftarrow B_1, \ldots, B_n \in \text{ground}(P), I \models B_1, \ldots, B_n \}$. For an ordinal $\alpha$, $T_P \uparrow \alpha$ and $T_P \downarrow \alpha$ are the upward and downward ordinal powers of $T_P$, respectively. A selection rule $\mathcal{F}$ is a function that assigns to an initial fragment of a SLD-derivation whose last goal is $G$ an atom in $G$, called the selected atom. Note that the definition of selection rules is more general than the notion of computation rules of Lloyd [11]. A selection rule $\mathcal{F}$ is fair if for every SLD-derivation $\xi$ via $\mathcal{F}$ either $\xi$ is finite or for every atom $A$ in $\xi$, (some further instantiated version of) $A$ is eventually selected. Finally, $N$ is the set of natural numbers.

2. $\exists$-universal termination and fair selection rules

First, we show that $\exists$-universal termination coincides with universal termination via fair selection rules. Therefore, any fair selection rule is a complete control for any logic program and goal for which a complete control exists, in the sense of Definition 1.1.

**Theorem 2.1.** A logic program $P$ and goal $G$ $\exists$-universally terminate iff every SLD-derivation of $P \cup \{ \leftarrow G \}$ via any fair selection rule is finite.

**Proof.** The if part is immediate. Conversely, suppose that there exists an infinite SLD-derivation $\xi$ via a fair selection rule $\mathcal{F}$. Let $\mathcal{F}$ be any other selection rule. Since $\mathcal{F}$ is fair, by reasoning as in the Switching Lemma [11, Lemma 9.1], we can switch the order of selection of the atoms in $\xi$ accordingly to $\mathcal{F}$, thus obtaining an infinite SLD-derivation of $P \cup \{ \leftarrow G \}$ via $\mathcal{F}$. $\square$

We propose a characterization of $\exists$-universal termination which is indeed a characterization of universal termination with respect to fair selection rules. Correctness and completeness of the method are proved by exploiting well-known properties of fair selection rules. In particular, the following relation between finitely failed fair SLD-trees and downward ordinals of the immediate consequence operator will be useful.

**Lemma 2.2.** Let $P$ be a logic program and $\leftarrow Q$ a goal. Then $T_P \downarrow i \not\models \exists Q$ for some $i \in N$ iff every SLD-derivation of $P \cup \{ \leftarrow Q \}$ via any fair selection rule is failed.

**Proof.** Consider the program $P' = P \cup \{ p \leftarrow Q \}$ where $p$ is a fresh predicate symbol. By definition of $T_P$, $T_P \downarrow i \not\models \exists Q$ for some $i \geq 0$ iff $p \not\in T_{P'} \downarrow i + 1$ for some $i \geq 0$ iff $p \in T_{P'} \downarrow \omega$. We recall (see [1, Theorem 5.6]) that $p \not\in T_{P'} \downarrow \omega$ iff every SLD-derivation of $P' \cup \{ \leftarrow p \}$ via any fair selection rule is failed. Since $p$ is a fresh symbol, we conclude that $T_P \downarrow i \not\models \exists Q$ for some $i \geq 0$ iff every SLD-derivation of $P \cup \{ \leftarrow Q \}$ via any fair selection rule is failed. $\square$
3. Fair-bounded logic programs

In this section we define the class of fair-bounded logic programs and goals. First, we introduce level mappings.

**Definition 3.1.** Given a logic program $P$, a level mapping for $P$ is a function $|| : BP \rightarrow N\infty$ of ground atoms to $N\infty$, where $N\infty = N \cup \{\infty\}$.

Intuitively, level mappings play the role of termination functions. However, differently from more standard definitions (see [6]), we included $\infty$ in the codomain of level mappings, as a means to model non-termination and uninteresting instances of program clauses and goals. We extend the $>$ order on naturals to a relation $\triangleright$ on $N\infty$.

**Definition 3.2.** We define the relation $n \triangleright m$ for $n, m \in N\infty$ as follows:

$$n \triangleright m \iff n = \infty \text{ or } n > m.$$ 

We write $n \trianglerighteq m$ iff $n \triangleright m$ or $n = m$.

Therefore, $\infty \triangleright eq m$ for every $m \in N\infty$. We are now in the position to introduce fair-bounded logic programs.

**Definition 3.3.** Let $P$ be a logic program, $I$ a Herbrand interpretation and $||$ a level mapping for $P$. $P$ is called fair-bounded by $||$ and $I$ if $I$ is a model of $P$ such that for every $A \leftarrow B_1, \ldots, B_n$ in $\text{ground}(P)$:

(a) $I \models B_1, \ldots, B_n$ implies for every $i \in [1, n]$ $|A| \triangleright |B_i|$, and

(b) $I \not\models B_1, \ldots, B_n$ implies there exists $i \in [1, n]$ $I \not\models B_i \land |A| \triangleright |B_i|$.

Let us focus the attention on some valuable properties that follow directly from Definition 3.3. First, we observe that the proof obligations are modular, in the sense that program clauses are taken into consideration separately. Second, the hypothesis of conditions (a) and (b) are mutually exclusive. Third, the notion of fair-boundedness is purely declarative, in the sense that neither any procedural notion is needed in order to prove a program fair-bounded nor the definition reflects some fixed ordering of the atoms. The next definition extends fair-boundedness to goals.

**Definition 3.4.** Let $G$ be goal, $I$ a Herbrand interpretation and $||$ a level mapping. $G$ is called fair-bounded by $||$ and $I$ if there exists $k \in N$ such that for every $A \leftarrow A_1, \ldots, A_n$ in $\text{ground}(G)$:

(a) $I \models A_1, \ldots, A_n$ implies for every $i \in [1, n]$ $|A| \triangleright k |A_i|$, and

(b) $I \not\models A_1, \ldots, A_n$ implies there exists $i \in [1, n]$ $I \not\models A_i \land |A| \triangleright k |A_i|$.

**Example 3.5.** The following program ProdCons abstracts a (concurrent) system composed of a producer and a consumer.
(s) \( \text{system}(N) \leftarrow \text{prod}(Bs), \text{cons}(Bs, N) \).

(p1) \( \text{prod}([s(0) \mid Bs]) \leftarrow \text{prod}(Bs) \).

(p2) \( \text{prod}([s(s(0)) \mid Bs]) \leftarrow \text{prod}(Bs) \).

(c) \( \text{prod}([\]) \).

(c) \( \text{cons}([D \mid Bs], s(N)) \leftarrow \text{cons}(Bs, N), \text{wait}(D) \).

(c) \( \text{cons}([\], 0) \).

(w) \( \text{wait}(s(D)) \leftarrow \text{wait}(D) \).

(w) \( \text{wait}(0) \).

For notational convenience, we identify the term \( s^n(0) \) with the natural number \( n \).
Intuitively, \( \text{prod} \) is the producer of a non-deterministic sequence of 1's and 2's, and \( \text{cons} \) is the consumer of the sequence. The shared variable \( Bs \) in clause (s) plays the role of an unbounded buffer. Moreover, since it is realistic to assume that consumption depends on \( n \), we model consumption by \( \text{wait} \). The overall system is started by the goal \( \leftarrow \text{system}(n) \), where \( n \in \mathbb{N} \), and stops when \( \text{cons} \) has consumed \( n \) messages.

Notice that \( \text{ProdCons} \) and a goal \( \leftarrow \text{system}(n) \) have infinite SLD-derivations via the leftmost and the rightmost selection rules. Actually, viewed as a concurrent system, the program and the goal need the assumption of fairness (i.e., fair selection rules) in order to terminate.

Let us show that \( \text{ProdCons} \) is fair-bounded. First, we define \( \text{size}(t) \) as the number of function symbols (excluding constants) appearing in a ground term \( t \). Moreover, we recall the \( \text{list-length} \) and \( \text{list-max} \) functions, that map ground terms into natural numbers as follows:

\[
\begin{align*}
\text{llen}(f(x_1, \ldots, x_n)) &= 0 \quad \text{if } f \neq [\] \\
\text{llen}([x|x]) &= \text{llen}(x) + 1 \quad \text{otherwise}, \\
\text{lmax}(f(x_1, \ldots, x_n)) &= 0 \quad \text{if } f \neq [\] \\
\text{lmax}([x|x]) &= \max\{\text{lmax}(x), \text{size}(x)\} \quad \text{otherwise}. 
\end{align*}
\]

Note that for a ground list \( xs \), \( \text{llen}(xs) \) is equal to the length of \( xs \) and \( \text{lmax}(xs) \) is equal to the maximum size of an element in \( xs \). Then we define:

\[
I = \{ \text{system}(N) \} \\
\cup \{ \text{prod}(bs) \mid \text{lmax}(bs) \leq 2 \} \\
\cup \{ \text{cons}(bs, n) \mid \text{llen}(bs) = \text{size}(n) \} \\
\cup \{ \text{wait}(X) \},
\]

\[
|\text{system}(n)| = \text{size}(n) + 3 \\
|\text{prod}(bs)| = \text{llen}(bs) \\
|\text{cons}(bs, n)| = \begin{cases} \text{size}(n) + \text{lmax}(bs) & \text{if } \text{cons}(bs, n) \in I \\ \text{size}(n) & \text{if } \text{cons}(bs, n) \notin I \end{cases} \\
|\text{wait}(t)| = \text{size}(t).
\]
Let us show the proof obligations of Definition 3.3. Those for unit clauses are trivial. Consider the recursive clauses (s), (p1), (p2), (c), and (w).

(w) $I$ is obviously a model of (w). Consider now a ground instance:

$$\text{wait}(s(d)) \leftarrow \text{wait}(d)$$

of (w). We observe that:

$$|\text{wait}(s(d))| = \text{size}(d) + 1 \triangleright \text{size}(d) = |\text{wait}(d)|.$$  This implies (a, b).

(c) Consider a ground instance:

$$\text{cons}([d | bs], s(n)) \leftarrow \text{cons}(bs, n), \text{wait}(d)$$

of (c). If $I \models \text{cons}(bs, n), \text{wait}(d)$ then $\text{llen}(bs) = \text{size}(n)$, and then:

$$\text{llen}([d | bs]) = \text{llen}(bs) + 1 = \text{size}(n) + 1 = \text{size}(s(n)),$$

i.e. $I \models \text{cons}([d | bs], s(n))$. Therefore, $I$ is a model of (c). Let us show proof obligations (a, b) of Definition 3.3.

(a) Suppose that $I \models \text{cons}(bs, n), \text{wait}(d)$. We have shown that $I \models \text{cons}([d | bs], s(n))$. We calculate:

$$|\text{cons}([d | bs], s(n))| = \text{size}(n) + 1 + \max\{l\text{max}(bs), \text{size}(d)\}$$

$\triangleright \text{size}(n) + l\text{max}(bs)$

$$= \{ I \models \text{cons}(bs, n) \}$$

$$|\text{cons}(bs, n)|,$$

and

$$|\text{cons}([d | bs], s(n))| = \text{size}(n) + 1 + \max\{l\text{max}(bs), \text{size}(d)\}$$

$\triangleright \text{size}(d)$

$$= |\text{wait}(d)|.$$  These two inequalities show that (a) holds.

(b) If $I \not\models \text{cons}(bs, n), \text{wait}(d)$ then necessarily $I \not\models \text{cons}(bs, n)$. This and

$$|\text{cons}([d | bs], s(n))|$$

$\triangleright \text{size}(n) + 1$

$\triangleright \text{size}(n)$

$$= \{ I \not\models \text{cons}(bs, n) \}$$

$$|\text{cons}(bs, n)|$$

show (b).
I is obviously a model of (p1). Moreover:

\[ |\text{prod}(bs)| = \text{llen}(bs) + 1 \]  

implies (a) and (b).

This case is analogous to the previous one.

Consider a ground instance:

\[
\text{system}(n) \leftarrow \text{prod}(bs), \text{cons}(bs,n)
\]

of (s). Obviously I is a model of (s). Let us show (a,b).

(a) Suppose that \( I \models \text{prod}(bs), \text{cons}(bs,n) \). This implies \( \text{lmax}(bs) \leq 2 \). Moreover, we have that \( \text{llen}(bs) = \text{size}(n) \). We calculate:

\[
|\text{system}(n)| = \text{size}(n) + 3
\]

\[
\begin{align*}
\triangleright & \quad \{ \text{llen}(bs) = \text{size}(n) \} \\
\text{llen}(bs) &= |\text{prod}(bs)|,
\end{align*}
\]

and

\[
|\text{system}(n)| = \text{size}(n) + 3
\]

\[
\begin{align*}
\triangleright & \quad \{ \text{lmax}(bs) \leq 2 \} \\
\text{size}(n) + \text{lmax}(bs) &= |\text{cons}(bs,n)|.
\end{align*}
\]

These two inequalities show (a).

(b) Suppose that \( I \not\models \text{prod}(bs), \text{cons}(bs,n) \). We distinguish two cases. If \( I \not\models \text{cons}(bs,n) \) then:

\[
|\text{system}(n)| = \text{size}(n) + 3 \triangleright \text{size}(n) = |\text{cons}(bs,n)|.
\]

If \( I \models \text{cons}(bs,n) \) and \( I \not\models \text{prod}(bs) \) then:

\[
|\text{system}(n)| = \text{size}(n) + 3
\]

\[
\begin{align*}
\triangleright & \quad \{ I \models \text{cons}(bs,n) \text{ implies llen}(bs) = size(n) \} \\
\text{llen}(bs) &= |\text{prod}(bs)|.
\end{align*}
\]

We conclude this example by noting that for every \( n \in \mathbb{N} \) the goal \( \leftarrow \text{system}(n) \) is fair-bounded by \(| | \) and I.

We claim that proving that a program is fair-bounded is simple and practical in paper & pencil proofs. In fact, proof obligations restrict to consider ground instances of clauses and goals. Substitutions and non-ground terms have not been taken into account,
since the method automatically lifts up to non-ground goals \( G \) by considering every
ground instance of \( G \). Moreover, the various tools used for proving fair-boundedness
have intuitive interpretations.

The level mapping \(| |\) is a decreasing function that ensures termination.

In particular, \( \infty \) allows for excluding from the termination analysis those (instances of)
clauses and goals which cause non-termination or that are uninteresting. In fact, if \(|A| = \infty\)
then conditions (a,b) in Definition 3.3 are trivially satisfied. The need for
reasoning on a subset of goals is motivated by the fact that logic programs are untyped,
and then goals may have atoms that in the intended interpretation of the programmer
are ill-typed. Another reason for introducing \( \infty \) is the fact that a program can terminate
for a strict subset of goals only.

**Example 3.6.** Consider the following simple program:

\[
q(a) \leftarrow q(a).
\]

\[
p(b) \leftarrow p(b).
\]

By defining:

\[
I = \emptyset, \quad |q(a)| = |p(b)| = \infty, \quad |p(a)| = |q(b)| = 0,
\]

it is readily checked that the program and the goal \( \leftarrow p(X), q(X) \) are fair-bounded
by \(||\) and \( I \). Later on, we will show that this implies that they \( \exists \)-universally terminate.
On the contrary, the program and the goal \( \leftarrow q(X) \) have only infinite SLD-derivations.
In particular, we observe that they are not fair-bounded by any \(||\) and \( I \). In fact, by
Definition 3.3 \((a,b), |q(a)| \triangleright |q(a)|\) must hold. Thus \(|q(a)| = \infty\). By Definition 3.4
\((a,b), \leftarrow q(X)\) cannot be fair-bounded by the same \(||\) and \( I \), otherwise there would
exist \( k \in \mathbb{N} \) such that \( k \triangleright \infty \). Summarizing, \( \infty \) allows us to concentrate on interesting
sets of goals by excluding ill-typed and non-terminating ones from the termination
analysis.

Finally, let us discuss in more detail the meaning of proof obligations (a) and (b)
in Definition 3.3. Consider a ground instance of a clause:

\[
A \leftarrow B_1, \ldots, B_n
\]

If the body \( B_1, \ldots, B_n \) is true in the model \( I \), then there might exist a SLD-refutation
for it. (a) is then intended to bound the length of the refutation.

If the body is not true in the model \( I \), then it cannot have a refutation. In this case,
termination actually means that there is an atom in the body that has a finitely failed
SLD-tree. (b) is then intended to bound the depth of the finitely failed SLD-tree.

These intuitions clarify why in Example 3.5 the level mapping for the cons atoms
distinguishes two cases. When cons(\( b \), \( n \)) is in \( I \), we bound the length of a possible
SLD-refutation, while when it is not in \( I \) we bound the depth of a finitely failed SLD-tree.
However, since the proof method is purely declarative, in practice we do not
reason about operational notions such as refutations and SLD-trees, but on models and level mappings.

Finally, the model $I$ is a description of some property of the declarative interpretation of the program, namely the least Herbrand model. In addition, as a consequence of the intended meaning of proof obligation (b) in Definition 3.3, the complement of $I$ is necessarily included in the finite failure set of the program. This intuition will be formally stated later in Lemma 4.2.

4. Termination correctness

In this section, we show that if a program and a goal are fair-bounded then they $\exists$-universally terminate. Therefore, fair-boundedness provides us with a correct method for proving $\exists$-universal termination. First of all, we show that the notion of fair-boundedness is persistent along SLD-derivations.

Lemma 4.1 (Persistency). Let $P$ be a logic program and $\leftarrow Q$ a goal both fair-bounded by $| |$ and $I$. Every SLD-resolvent $\leftarrow Q'$ of $P \cup \{ \leftarrow Q \}$ is fair-bounded by $| |$ and $I$.

Proof. First of all, we observe that for any substitution $\theta$, directly from Definition 3.4, $\leftarrow Q\theta$ is fair-bounded by $| |$ and $I$ by using some bound $\in N$. Let $\theta$ be now the mgu of the selected atom in $Q$ and the input clause head. Assume that $\leftarrow Q\theta = \leftarrow A_1, \ldots, A_n$, and that $c : A_k \leftarrow B_1, \ldots, B_m$ is the instantiation by $\theta$ of the input clause. Then $\leftarrow Q'$ is

$$\leftarrow A_1, \ldots, A_{k-1}, B_1, \ldots, B_m, A_{k+1}, \ldots, A_n.$$ 

Let now $Q_m = A'_1, \ldots, A'_{k-1}, B'_1, \ldots, B'_m, A'_{k+1}, \ldots, A'_n$ be a ground instance of $Q'$. Then there exists $A'_1, \ldots, A'_n$

ground instance of $Q\theta$, with $A'_k \leftarrow B'_1, \ldots, B'_m$ ground instance of $c$.

Let us show the proof obligations of Definition 3.4.

(a) Suppose that $I \models Q_m$. Then $I \models B'_1, \ldots, B'_m$ and, since $I$ is a model of $P$, $I \models A'_1, \ldots, A'_n$. By Definition 3.4(a), bound $\triangleright |A'_i|$ for every $i \in [1, n]$. Moreover, since $I \models B'_1, \ldots, B'_m$, then bound $\triangleright |A'_k| \triangleright |B'_i|$ for $i \in [1, m]$.

(b) Suppose that $I \not\models Q_m$. We distinguish two cases.

(b1) If $I \models A'_1, \ldots, A'_n$ then bound $\triangleright |A'_k|$ by Definition 3.4(a). Moreover, $I \not\models B'_1, \ldots, B'_m$. By Definition 3.3(b) there exists $i \in [1, m]$ such that $|A'_k| \triangleright |B'_i|$ and $I \not\models B'_i$. We conclude that $I \not\models B'_i$ and bound $\triangleright |A'_i| \triangleright |B'_i|$.

(b2) If $I \models A'_1, \ldots, A'_n$ then by Definition 3.4(b), there exists $i \in [1, n]$ such that $I \not\models A'_i$ and bound $\triangleright |A'_i|$. We distinguish two cases.

If $i \neq k$ then the conclusion follows since $A'_i$ is in $Q_m$. 


Suppose, on the contrary, that \( I \not\models A'_k \land \text{bound} \triangleright \langle A'_k \rangle \). Since \( I \) is a model of \( P \), \( I \not\models A'_k \) implies \( I \not\models B'_1, \ldots, B'_n \). By Definition 3.3(b), there exists \( i \in [1, m] \) such that \( \langle A'_i \rangle \triangleright \langle B'_i \rangle \) and \( I \not\models B'_i \). We then conclude that \( I \not\models B'_i \) and \( \text{bound} \triangleright \langle A'_k \rangle \triangleright \langle B'_i \rangle \). □

The following property of fair-bounded programs can be intuitively read as follows. Any ground atom in the complement of the Herbrand interpretation \( I \) used to prove \( P \) fair-bounded has level equal to \( \infty \) or belongs to the finite failure set of \( P \).

**Lemma 4.2.** Let \( P \) be a logic program fair-bounded by \( \| \| \) and \( I \). Then for every ground atom \( A \not\in I \), if \( |A| \neq \infty \) then \( A \not\in T_P \downarrow |A| + 1 \).

**Proof.** Suppose that \( A \not\in I \) and \( |A| \neq \infty \). The proof proceeds by induction on \( |A| \).

\( (|A| = 0) \) We claim that no \( A \leftarrow B_1, \ldots, B_n \) is in \( \text{ground}(P) \). Otherwise, since \( I \) is a model of \( P \) and \( A \not\in I \) then \( I \not\models B_1, \ldots, B_n \). By Definition 3.3(b) there exists \( i \in [1, n] \) such that \( 0 \triangleright \langle B_i \rangle \), which is impossible. By definition of \( T_P \), we conclude that \( A \not\in T_P \downarrow 1 \).

\( (|A| > 0) \) We distinguish two cases.

Suppose that there is no \( A \leftarrow B_1, \ldots, B_n \) in \( \text{ground}(P) \). By definition of \( T_P \), we have that \( A \not\in T_P \downarrow 1 \). By monotonicity of \( T_P \), we conclude \( A \not\in T_P \downarrow |A| + 1 \).

Suppose, on the contrary, that there exists \( A \leftarrow B_1, \ldots, B_n \) in \( \text{ground}(P) \). Since \( I \) is a model of \( P \) and \( A \not\in I \) then \( I \not\models B_1, \ldots, B_n \). By Definition 3.3(b) there exists \( i \in [1, n] \) such that \( I \not\models B_i \) and \( |A| \triangleright \langle B_i \rangle \). By induction hypothesis, we have that \( B_i \not\in T_P \downarrow |B_i| + 1 \). By monotonicity of \( T_P \), we observe that \( B_i \not\in T_P \downarrow |A| \). By definition of \( T_P \), we conclude that \( A \not\in T_P \downarrow |A| + 1 \). □

In what follows we shall use the multiset ordering. A **multiset**, sometimes called **bag**, is an unordered sequence. We denote a multiset consisting of elements \( a_1, \ldots, a_n \) by **bag** \((a_1, \ldots, a_n)\). Given a (non-reflexive) ordering \( > \) on a set \( W \), the **multiset ordering over** \((W, >)\) is an ordering on finite multisets of the set \( W \), and is denoted by \( \succ_k \). It is defined as the transitive closure of the relation \( \succ \) in which \( X \succ Y \) if \( Y \) can be obtained from \( X \) by replacing an element \( a \) of \( X \) by a finite (possibly zero) number of elements \( b \) such that \( a > b \). Finally, we write \( X \succ_k Y \) if \( X \succ \succ_k Y \) or \( X = Y \).

It is well-known (see e.g. Dershowitz [5]) that the multiset ordering over a well-founded ordering is again well-founded. In particular, the multiset ordering over the set of natural numbers with their usual ordering is well-founded. Next we associate a finite multiset over \( N \) to fair-bounded goals.

**Definition 4.3.** Let \( G \leftarrow A_1, \ldots, A_n \) be a goal fair-bounded by \( \| \| \) and \( I \). We define the sets \( |G|^i \) for \( i \in [1, n] \) as follows:

\[
|G|^i = \{ |A^i| \leftarrow A'_1, \ldots, A'_m \in \text{ground}(G) \land I \models A'_1, \ldots, A'_m \}.
\]

We define \( |G|^f \) as the finite multiset

\[
|G|^f = \text{bag}(\max |G|^1, \ldots, \max |G|^n),
\]

if \( I \models \exists (A_1, \ldots, A_n) \), and \( |G|^f = \text{bag}() \) if \( I \not\models \exists (A_1, \ldots, A_n) \).
We observe that the definition is well-formed. By Definition 3.4, the sets $|G_i|$ for $i \in [1,n]$ are finite, and then there exists the maximum (which is 0 in case of empty sets). The following lemma shows a crucial relation between a goal and its SLD-resolvents.

**Lemma 4.4.** Let $P$ be a logic program and $\leftarrow Q$ a goal both fair-bounded by $||$ and $I$. For every SLD-resolvent $\leftarrow Q'$ of $P \cup \{ \leftarrow Q \}$, we have that:

(i) $|\leftarrow Q'|_i \geq_m |\leftarrow Q|_i$, and

(ii) if $I \models \exists Q'$ then $|\leftarrow Q'|_i >_m |\leftarrow Q|_i$.

**Proof.** First of all, we observe that for every substitution $\theta$,

$$|\leftarrow Q|_i \geq_m |\leftarrow Q\theta|_i.$$  \hspace{1cm} (1)

In fact, by Definition 4.3 $|\leftarrow Q|_i \supseteq |\leftarrow Q\theta|_i$ holds for $i \in [1,n]$, where $n$ is the number of atoms in $Q$.

Let $\theta$ be now the mgu of the selected atom in $Q$ and the input clause head. Assume that $\leftarrow Q\theta = \leftarrow A_1, \ldots, A_n$, and that $c : A_k \leftarrow B_1, \ldots, B_m$ is the instantiation by $\theta$ of the input clause. Then $\leftarrow Q'$ is

$$\leftarrow A_1, \ldots, A_{k-1}, B_1, \ldots, B_m, A_{k+1}, \ldots, A_n.$$  

First, suppose that $I \not\models \exists Q'$. Then we have to show only (i), which is immediate by observing that $|\leftarrow Q|_i \geq_m \text{bag}() = |\leftarrow Q'|_i$.

On the other hand, assume now that $I \models \exists Q'$. Then for every $i \in [1,n+m-1]$:

$$|\leftarrow Q'|_i \neq \emptyset.$$  \hspace{1cm} (2)

Let now $Q_m = A'_1, \ldots, A'_{k-1}, B'_1, \ldots, B'_m, A'_{k+1}, \ldots, A'_n$ be a ground instance of $Q'$ such that $I \models Q_m$. Then there exists

$$A'_1, \ldots, A'_n$$

ground instance of $Q\theta$, with $A'_k \leftarrow B'_1, \ldots, B'_m$ ground instance of $c$. As a consequence for every $i \in [1,n]$, $i \neq k$, we have $|A'_i| \in |\leftarrow Q\theta|_i$. This implies:

for $i \in [1,k-1]$ max $|\leftarrow Q'|_{i} \leq \max |\leftarrow Q\theta|_{i}$, \hspace{1cm} (3)

for $i \in [k+1,n]$ max $|\leftarrow Q'|_{i+m-1} \leq \max |\leftarrow Q\theta|_{i}$. \hspace{1cm} (4)

Moreover, since $I \models B'_1, \ldots, B'_m$, by Definition 3.3(a), we have that $|A'_k| > |B'_i|$ for every $i \in [1,m]$. By the assumption that $\leftarrow Q$ is fair-bounded, we have that $|A'_k| \in N$ and then $|A'_k| > |B'_i|$ for every $i \in [1,m]$. Summarizing:

for $i \in [1,m]$ $\forall x \in |\leftarrow Q'|_{k+i-1} \exists y \in |\leftarrow Q\theta|_{k} y > x$.  \hspace{1cm} (5)
In conclusion, we calculate:

\[ |\leftarrow Q|^l \]
\[ \geq_m \{ (1) \} \]
\[ |\leftarrow Q0|^l \]
\[ = \{ \text{Definition 4.3} \} \]
\[ \text{bag}(\max |\leftarrow Q0|^l, \ldots, \max |\leftarrow Q0|^l, \ldots, \max |\leftarrow Q0|^l_n) \]
\[ \geq_m \{ (2–5) \} \]

and the fact that
\[ \forall x \in S \neq \emptyset \exists y \in R, y > x \implies \max R > \max S \}
\[ \text{bag}(\max |\leftarrow Q'|^l, \ldots, \max |\leftarrow Q'|^l_n) \]
\[ = \{ \text{Definition 4.3} \} \]
\[ |\leftarrow Q'|^l \]

which implies (i) and (ii).

The next Theorem shows that no SLD-derivation of fair-bounded programs and goals is infinite via a fair selection rule.

**Theorem 4.5.** Let \( P \) be a logic program and \( \leftarrow Q \) a goal both fair-bounded by \( | | \) and \( I \). Then every SLD-derivation of \( P \cup \{ \leftarrow Q \} \) via a fair selection rule is finite.

**Proof.** Suppose that there exists an infinite SLD-derivation \( \leftarrow Q_1, \ldots, \leftarrow Q_n, \ldots \) of \( P \cup \{ \leftarrow Q \} \) via a fair selection rule. By Lemma 4.1, every \( \leftarrow Q_i \) is fair-bounded by \( | | \) and \( I \). We distinguish two cases depending whether or not for every \( i \geq 1 \), \( I \models \exists Q_i \).

Suppose that \( I \models \exists Q_i \) for some \( i \geq 1 \). By Definition 3.4(b), there exists \( k \in N \) such that for every \( A_1, \ldots, A_n \) ground instance of \( Q_i \), there exists \( j \in [1, n] \) such that \( I \notmodels A_j \) and \( k > |A_j| \). By Lemma 4.2, \( A_j \not\in T_p \upharpoonright |A_j| + 1 \). By monotonicity of \( T_p \), \( A_j \not\in T_p \upharpoonright k \). Summarizing, \( T_p \upharpoonright k \notmodels \exists Q_i \). By Lemma 2.2, every fair SLD-derivation of \( P \cup \{ \leftarrow Q_i \} \) is failed, hence finite. This contradicts the assumption that there exists an infinite fair SLD-derivation.

Suppose now that for every \( i \geq 1 \), \( I \models \exists Q_i \).

By Lemma 4.4(ii), \( |\leftarrow Q|^l \succ_m \cdots \succ_m |\leftarrow Q|^l \succ_m \cdots \) is an infinite descending chain of bags over naturals. This is impossible since the finite multiset ordering over naturals is well-founded.

We are in the position to state correctness of the proposed proof method.

**Theorem 4.6** (Termination correctness). Let \( P \) be a logic program and \( G \) a goal both fair-bounded by \( | | \) and \( I \). Then \( P \) and \( G \) \( \exists \)-universally terminate.

**Proof.** By Theorem 4.5, every SLD-derivation of \( P \cup \{ G \} \) via a fair selection rule is finite. By Theorem 2.1, we conclude that \( P \) and \( G \) \( \exists \)-universally terminate.
Example 4.7. Let us consider Example 3.5. We showed that \texttt{ProdCons} and the goal $\leftarrow \texttt{system}(n)$ are both fair-bounded by $\mid \mid$ and $I$, where $n \in N$. By Theorem 4.5, we conclude that every fair SLD-derivation of $\texttt{ProdCons} \cup \{\leftarrow \texttt{system}(n)\}$ is finite, i.e. they $\exists$-universally terminate.

We conclude this section with a corollary showing that the upward and downward ordinal closures of $T_P$, with $P$ fair-bounded, coincide on the set $\{A \mid A \neq \infty\}$.

Corollary 4.8. Let $P$ be a logic program fair-bounded by $\mid \mid$ and $I$. For every $A \in B_P$ such that $|A| \neq \infty$, we have that

$$A \in T_P \downarrow \omega \iff A \in T_P \uparrow \omega.$$

Proof. Since $|A| \in N$, we have that the goal $\leftarrow A$ is fair-bounded. By Theorem 4.5, every SLD-derivation of $P \cup \{\leftarrow A\}$ via a fair selection rule $\mathcal{S}$ is finite. Then either there exists a SLD-refutation or every SLD-derivation via $\mathcal{S}$ is failed. We recall that there exists a refutation of $P \cup \{\leftarrow A\}$ iff $A \in T_P \uparrow \omega$, and that every SLD-derivation via $\mathcal{S}$ is failed iff $A \notin T_P \downarrow \omega$ (see Lemma 2.2). This implies that $A \in T_P \downarrow \omega$ iff $A \in T_P \uparrow \omega$. \hfill \Box

5. Termination completeness

In this section, we show that if $P$ and $G$ $\exists$-universally terminate then they are fair-bounded by some $\mid \mid$ and $I$. Therefore, fair-boundedness is a correct and complete proof method for $\exists$-universal termination. We start with a simple observation.

Lemma 5.1. For every logic program $P$, $T_P \downarrow \omega$ is a model of $P$.

Proof. Since $T_P \downarrow \omega = \bigcap_{i \in N} T_P \downarrow i$, by monotonicity of $T_P$, $T_P(T_P \downarrow \omega) \subseteq T_P \downarrow i + 1$, for every $i \in N$. Since $T_P \downarrow 0 = B_P$, this implies $T_P(T_P \downarrow \omega) \subseteq \bigcap_{i \in N} T_P \downarrow i + 1 = T_P \downarrow \omega$, i.e. $T_P \downarrow \omega$ is a model of $P$. \hfill \Box

Next, we introduce two further definitions.

Definition 5.2. For a logic program $P$ and a goal $G$, we define $\text{length}^P_{\mathcal{S}}(G)$ as $\infty$ if there exists an infinite SLD-derivation of $P \cup \{G\}$ via the selection rule $\mathcal{S}$, and as the maximum length of a SLD-derivation of $P \cup \{G\}$ via $\mathcal{S}$ if no infinite SLD-derivation via $\mathcal{S}$ exists.

We observe that the definition of $\text{length}^P_{\mathcal{S}}$ is well-formed. In fact, since SLD-trees are finitely branching, by König’s Lemma if there is no infinite SLD-derivation of $P \cup \{G\}$ via $\mathcal{S}$ then there are finitely many SLD-derivations of $P \cup \{G\}$, hence the maximum length exists.
**Definition 5.3.** The *round-robin* selection rule $\mathcal{RR}$ selects atoms in the last goal of an initial fragment $\xi^<$ of SLD-derivations as follows. If $\xi^<$ consists only of the initial goal, then the leftmost atom is selected. Otherwise, if at the previous step in $\xi^<$, the atom $A$ in a goal $G$ was selected, and $A$ was not the rightmost atom in $G$, then the (instantiated version of the) atom following $A$ in $G$ is selected. Finally, if at the previous step of $\xi^<$ the rightmost atom $A$ was selected, then the leftmost atom of the last goal in $\xi^<$ is selected.

Obviously $\mathcal{RR}$ is a fair selection rule. Another useful property of $\text{length}^p_\mathcal{RR}$ is the following.

**Lemma 5.4.** Let $P$ be a logic program, $G$ a goal and $A$ an atom. Then

(i) for every $G'$ SLD-resolvent of $P \cup \{ \leftarrow A \}$

$$\text{length}^p_\mathcal{RR}(\leftarrow A) > \text{length}^p_\mathcal{RR}(G'),$$

(ii) for every $G'$ instance of $G$, $\text{length}^p_\mathcal{RR}(G) \preceq \text{length}^p_\mathcal{RR}(G')$.

**Proof.** (i) is immediate by definition of $\text{length}^p_\mathcal{RR}$. Let us show (ii). Consider a SLD-resolvent $G'_1$ of $P \cup \{ G' \}$ via $\mathcal{RR}$. Since $G'$ is an instance of $G$, the atoms selected by $\mathcal{RR}$ in the two goals occur in the same position. Therefore, there exists a SLD-resolvent $G_1$ of $P \cup \{ G \}$ via $\mathcal{RR}$ such that $G'_1$ is an instance of $G_1$. In general, for every SLD-derivation $G', G'_1, \ldots, G'_n, \ldots$ via $\mathcal{RR}$ there exists:

$$G, G_1, \ldots, G_n, \ldots$$

(initial fragment of a) SLD-derivation via $\mathcal{RR}$ such that $G'_i$ is an instance of $G_i$ for $i \geq 1$. Therefore, $\text{length}^p_\mathcal{RR}(G) \preceq \text{length}^p_\mathcal{RR}(G')$. □

We are now in the position to show that fair-boundedness is a complete termination proof method with respect to fair selection rules.

**Lemma 5.5.** Let $P$ be a logic program and $\mathcal{F}$ a fair selection rule. Then there exist a level mapping $\| \|$ and Herbrand interpretation $I$ such that:

(i) $P$ is fair-bounded by $\| \|$ and $I$, and

(ii) for every $A \in B_P$, $|A| \in N$ iff every SLD-derivation of $P \cup \{ \leftarrow A \}$ via $\mathcal{F}$ is finite.

**Proof.** We define $I = T_P \downarrow \omega$ and $|A| = \text{length}^p_\mathcal{RR}(\leftarrow A)$.

First, consider (ii). As a consequence of Theorem 2.1, every SLD-derivation of $P \cup \{ \leftarrow A \}$ via $\mathcal{F}$ is finite iff every SLD-derivation of $P \cup \{ \leftarrow A \}$ via the (fair) selection rule $\mathcal{RR}$ is finite. By definition of $\text{length}^p_\mathcal{RR}$, we have that $|A| \in N$ iff every SLD-derivation of $P \cup \{ \leftarrow A \}$ via $\mathcal{F}$ is finite.

Let us now consider (i). We show the proof obligations of Definition 3.3. By Lemma 5.1, $I$ is a model of $P$. Consider now $A \leftarrow B_1, \ldots, B_n$ in $\text{ground}(P)$.
(a) Suppose that $I \models B_1, \ldots, B_n$. By definition of $I$, this implies that for every $k \geq 0$, $T_P \downarrow k \models B_1, \ldots, B_n$. By Lemma 2.2, we conclude that there exists at least one non-failed SLD-derivation $\xi$ of $P \cup \{ \leftarrow B_1, \ldots, B_n \}$ via $\mathcal{R}_P$. We claim that for $i \in [1, n]$:

$$\text{length}^P_{\mathcal{R}_P}(\leftarrow B_1, \ldots, B_n) \geq \text{length}^P_{\mathcal{R}_P}(\leftarrow B_i).$$

(6)

In fact, consider a SLD-derivation $\xi'$ of $P \cup \{ \leftarrow B_i \}$ via $\mathcal{R}_P$. Since $\mathcal{R}_P$ is fair and $\xi$ is non-failed, there exists a SLD-derivation of $P \cup \{ \leftarrow B_1, \ldots, B_n \}$ where all the atoms in $\xi'$ are eventually selected, and the other selections are made accordingly to $\xi$. Thus, we obtain a SLD-derivation of $P \cup \{ \leftarrow B_1, \ldots, B_n \}$ via $\mathcal{R}_P$ whose length is greater or equal than the length of $\xi'$. In conclusion, (6) holds. Observing that $\leftarrow B_1, \ldots, B_n$ is an instance of a SLD-resolvent $G$ of $P \cup \{ \leftarrow A \}$, we now calculate for $i \in [1, n]$:

$$|A| = \text{length}^P_{\mathcal{R}_P}(\leftarrow A)
\Downarrow \{ \text{Lemma 5.4(i)} \}
\text{length}^P_{\mathcal{R}_P}(G)
\Downarrow \{ \text{Lemma 5.4(ii)} \}
\text{length}^P_{\mathcal{R}_P}(\leftarrow B_1, \ldots, B_n)
\Downarrow \{ (6) \}
\text{length}^P_{\mathcal{R}_P}(\leftarrow B_i) = |B_i|.$$

(b) Suppose now that $I \not\models B_1, \ldots, B_n$. By definition of $I$, we have that for some $i \in [1, n]$ there exists $k_i \geq 0$ such that $T_P \downarrow k_i \not\models B_i$. Consider now $B_h$ such that $T_P \downarrow k_h \not\models B_h$ and $\text{length}^P_{\mathcal{R}_P}(\leftarrow B_h)$ is minimum.

By Lemma 2.2, we have that every SLD-derivation of $P \cup \{ \leftarrow B_h \}$ via $\mathcal{R}_P$ is failed. By definition of $h$ and Lemma 2.2, we have that for $i \neq h$ there exists a SLD-derivation $\xi_i$ of $P \cup \{ \leftarrow B_i \}$ via $\mathcal{R}_P$ such that $\xi_i$ is successful or has length greater or equal than $\text{length}^P_{\mathcal{R}_P}(\leftarrow B_h)$. We claim that:

$$\text{length}^P_{\mathcal{R}_P}(\leftarrow B_1, \ldots, B_n) \geq \text{length}^P_{\mathcal{R}_P}(\leftarrow B_h).$$

(7)

In fact, let $\xi$ be a SLD-derivation of $P \cup \{ \leftarrow B_h \}$ with length $\text{length}^P_{\mathcal{R}_P}(\leftarrow B_h)$. Consider now a SLD-derivation $\xi'$ of $P \cup \{ \leftarrow B_1, \ldots, B_n \}$ via $\mathcal{R}_P$ where the atoms are selected accordingly to $\xi$ and to $\xi_i$ for $i \neq h$. We observe that the length of $\xi'$ is at least $\text{length}^P_{\mathcal{R}_P}(\leftarrow B_h)$, since $\xi_i$ for $i \neq h$ is successful or longer than $\xi$. In conclusion, (7) holds. Observing that $\leftarrow B_1, \ldots, B_n$ is an instance of a SLD-resolvent $G$ of $P \cup \{ \leftarrow A \}$, we calculate:

$$|A| = \text{length}^P_{\mathcal{R}_P}(\leftarrow A)
\Downarrow \{ \text{Lemma 5.4(i)} \}
\text{length}^P_{\mathcal{R}_P}(G)
\Downarrow \{ \text{Lemma 5.4(ii)} \}$$
\[
\text{length}_{\mathcal{R}, \mathcal{P}}(\leftarrow B_1, \ldots, B_n)
\geq \{ (7) \}
\text{length}_{\mathcal{R}, \mathcal{P}}(\leftarrow B_h) = |B_h|,
\]

and then we conclude that \( I \not\models B_h \) and \(|A| \triangleright |B_h|\).

Finally, we state completeness of the proof method.

**Theorem 5.6** (Termination completeness). Let \( P \) be a logic program and \( G \) a goal that \( \exists \)-universally terminate. Then there exist \( | | \) and \( I \) such that \( P \) and \( G \) are both fair-bounded by \( | | \) and \( I \).

**Proof.** Assume that \( G = \leftarrow Q \). By Theorem 2.1, every SLD-derivation of \( P \cup \{G\} \) via any fair selection rule \( \mathcal{F} \) is finite. Consider the program \( P' = P \cup \{ \text{new} \leftarrow Q \} \), where \( \text{new} \) is a fresh predicate symbol. By Lemma 5.5(i), \( P' \) is fair-bounded by some \( | | \) and \( I \). Since \( \text{new} \) is a fresh symbol, the assumption of the Theorem implies that every SLD-derivation of \( P' \cup \{ \text{new} \} \) via \( \mathcal{F} \) is finite. By Lemma 5.5(ii), we conclude that \( |\text{new}| \in N \). Consider now the restrictions of \( | | \) and \( I \) to \( B_{P'} \), i.e. not including \( \text{new} \). Since the definition of fair-boundedness is modular, \( P \) is fair-bounded by the restrictions of \( | | \) and \( I \). Turning the attention on \( G \), since \( \text{new} \leftarrow Q \) is fair-bounded by \( | | \) and \( I \), we have that for every ground instance \( \leftarrow A_1, \ldots, A_n \) of \( G \):

(a) if \( I \models A_1, \ldots, A_n \) then for \( i \in [1, n] \), \( |\text{new}| \triangleright |A_i| \), and
(b) if \( I \not\models A_1, \ldots, A_n \) then there exists \( i \in [1, n] \) such that \( I \not\models A_i \) and \( |\text{new}| \triangleright |A_i| \).

In conclusion \( G \) is fair-bounded by the restrictions of \( | | \) and \( I \), by fixing \( k = |\text{new}| \) in Definition 3.4.

Summarizing, in the last two sections we showed that the class of fair-bounded logic programs and goals precisely characterize the notion of \( \exists \)-universal termination, i.e. the class of logic programs and goals for which a complete control exists in the sense of Definition 1.1.

6. Arithmetic built-in’s

A program with arithmetic (see [21]) is a logic program in which the predicates:

\(<, \leq, =, \neq, \text{is}, >, \geq, \)

can appear only in clause bodies. These predicates are defined for particular terms, called ground arithmetic expressions (in short, gae’s). The set of gae’s is denoted by \( \text{Gae} \). In this section, we show that fair-boundedness naturally extends to programs containing those predicates.
Accordingly to [21], we extend SLD-resolution assuming that the predicate \( \succ \) is defined by the (infinite) set of unit clauses:

\[
\text{Def}_\succ = \{ n \succ m \mid n, m \in \text{Gae} \land \text{value}(n) \succ \text{value}(m) \},
\]

where \( \text{value}(n) \) is the number denoted by \( n \). \( P \cup \text{Def}_\succ \) is then the (infinite) set of clauses defining \( P \) and the built-in \( \succ \). Analogous definitions can be given for \( \prec, =\prec, =:, =/\sim, \geq \). The definition of \( \equiv \) is the following:

\[
\text{Def}_{\equiv} = \{ \text{value}(m) \equiv m \mid m \in \text{Gae} \}.
\]

Consider now a SLD-derivation for a program with arithmetic and a goal such that the atom \( n \succ m \) is selected. If \( n, m \) are gae’s then, according to the definition of \( \succ \), the SLD-derivation fails if \( \text{value}(n) \) is lower or equal than \( \text{value}(m) \). If the value of \( n \) is greater than the value of \( m \), the resolvent is the rest of the goal. Since the resulting SLD-trees are not finitely branching, a further assumption is necessary.

**Definition 6.1.** We say that a SLD-derivation **ends with a run-time error** if an atom \( n \succ m \) is selected with \( n \) or \( m \) not gae’s.

A similar definition is given for \( \prec, =\prec, =:, =/\sim, \geq \), whereas for \( \equiv \) only the second argument is required to be a gae.

Notice that this is the procedural semantics of \( \succ \) in Prolog. In recent systems, such as Gödel and Mercury, the selection of insufficiently instantiated atoms is delayed until possible. In particular, Mercury statically reorders clause bodies in such a way that its type system can ensure that no insufficiently instantiated atoms will be selected via the leftmost selection rule. Gödel, instead, selects only arithmetic atom that are sufficiently instantiated.

Let us consider now fairness in presence of arithmetic built-in’s. The next Theorem extends the Correctness Theorem 4.6.

**Theorem 6.2.** Let \( P \) be a program with arithmetic, and \( \text{Def} \) be the definition of the arithmetic built-in’s occurring in \( P \). If \( P \cup \text{Def} \) and a goal \( G \) are both fair-bounded by \( || \) and \( I \), then every SLD-derivation of \( P \cup \{ G \} \) via a fair selection rule is finite (possibly ending with a run-time error).

**Proof.** Analogous to the proof of Theorem 4.5, by noting that the only-if part of Lemma 2.2 holds for programs with arithmetic by stating that SLD-derivations via fair selection rules either are failed or end with a run-time error.

**Example 6.3.** Let us consider the Partition program.

\[
\begin{align*}
\text{part}(X, [Y | Ys], [Y | Ls], Bs) & \leftarrow \\
& X \succ Y, \text{part}(X, Ys, Ls, Bs).
\end{align*}
\]
\[ \text{Partition} \cup \text{Def}_\succ \cup \text{Def}_\prec \text{ is fair-bounded by } || \text{ and any model } I \text{ of it, where:} \]
\[ |\text{part}(x, y, l, b)| = \text{llen}(y), \]
\[ |s > t| = 0, \]
\[ |s =< t| = 0. \]

The proof obligations for the unit clauses are trivial. Let us consider now a ground instance of the first clause (for the second clause we reason symmetrically):
\[ \text{part}(x, [y], [y], b) \leftarrow x > y, \text{ part}(x, y, l, b). \]

We have that:
\[ |\text{part}(x, [y], [y], b)| \]
\[ = \text{llen}(y) + 1 \]
\[ > 0 \]
\[ = |x > y|, \]
and
\[ |\text{part}(x, [y], [y], b)| \]
\[ = \text{llen}(y) + 1 \]
\[ > \text{llen}(y) \]
\[ = |\text{part}(x, y, l, b)|. \]

Suppose now that the body of the clause instance is true in \( I \). Then the two inequalities above show condition (a) of Definition 3.3. On the contrary, suppose that the body is not true in \( I \). Then one of the two body atoms in not in \( I \). Since the two inequalities above show the decreasing of level mapping for both, we have that there exists a body atom not in \( I \) for which the level mapping decreases, i.e. condition (b) of Definition 3.3 holds as well.

Summarizing, Partition is fair-bounded by \( || \) and \( I \). Moreover the goal \( \leftarrow \text{part}(X, Y, L, B) \) is fair-bounded by the same \( || \) and \( I \), when \( Y \) is a (not necessarily ground) list and \( X \) any term. Finally, by Theorem 6.2 we conclude that every SLD-derivation of Partition \( \cup \{ \leftarrow \text{part}(X, Y, L, B) \} \) via a fair selection rule is finite (possibly ending with a run-time error).
7. Classes of terminating programs

As a by-result of the Termination Completeness Theorem 5.6, any class of programs and goals that universally terminate via some selection rule is included in the class of fair-bounded programs and goals. In this section, we relate fair-bounded logic program to the well-known classes of recurrent, acceptable and bounded programs.

7.1. Recurrent programs

Recurrent programs were introduced by Bezem [4]. They coincide with the class of programs such that every SLD-derivation for every ground goal via every selection rule is LFFnite.

Definition 7.1. Let \( P \) be a logic program and \( |\| : B_P \rightarrow N \) a level mapping for \( P \). \( P \) is called recurrent by \( |\| \) iff for every \( A \leftarrow B_1, \ldots, B_n \) in \( \text{ground}(P) \),

\[
\text{for } i \in [1,n] : \quad |A| > |B_i|.
\]

Notice that the level mapping is required to have naturals as codomain. The following result states that proving a program recurrent is a sufficient condition for proving the program fair-bounded. For instance, the proof that the Partition program is fair-bounded (see Example 6.3) is in practice a proof that it is recurrent.

Theorem 7.2. If a logic program \( P \) is recurrent by \( |\| \) then \( P \) is fair-bounded by \( |\| \) and any Herbrand model \( I \) of \( P \).

Proof. Immediate from Definitions 3.3 and 7.1. □

7.2. Acceptable programs

Acceptable programs were introduced by Apt and Pedreschi [2]. They coincide with the class of programs such that every SLD-derivation for every ground goal via the leftmost selection rule is finite.

Definition 7.3. Let \( P \) be a logic program, \( I \) a Herbrand interpretation and \( |\| : B_P \rightarrow N \) a level mapping for \( P \). \( P \) is called acceptable by \( |\| \) and \( I \) iff \( I \) is a model of \( P \) and for every \( A \leftarrow B_1, \ldots, B_n \) in \( \text{ground}(P) \),

\[
\text{for } i \in [1,n] : \quad I \models B_1, \ldots, B_{i-1} \quad \text{implies} \quad |A| > |B_i|.
\]

A large number of Prolog programs have been shown to be acceptable [2]. As in the case of recurrent programs, acceptability is a sufficient condition for proving fair-boundedness.

Theorem 7.4. Let \( P \) a logic program acceptable by \( |\| \) and \( I \). Then \( P \) is fair-bounded by \( |\| \) and \( I \).
Proof. Consider \( A \leftarrow B_1, \ldots, B_n \) in \( \text{ground}(P) \):
(a) if \( I \models B_1, \ldots, B_n \) then by Definition 7.3 we conclude that for every \( i \in [1, n] \)
\( |A| > |B_i| \).
(b) if \( I \not\models B_1, \ldots, B_n \) then let \( k \in [1, n] \) such that \( I \models B_1, \ldots, B_{k-1} \) and \( I \not\models B_k \). By Definition 7.3, we conclude \( |A| > |B_k| \). Summarizing, \( I \not\models B_k \) and \( |A| > |B_k| \). \( \Box \)

7.3. Bounded programs

Bounded programs were introduced by Ruggieri [17]. They coincide with the class of programs such that there are finitely many SLD-refutations for every ground goal via any selection rule.

Definition 7.5. Let \( P \) be a logic program, \( I \) a Herbrand interpretation and \( |: B_P \rightarrow N \) a level mapping for \( P \). \( P \) is called bounded by \( | \) and \( I \) iff \( I \) is a model of \( P \) such that for every \( A \leftarrow B_1, \ldots, B_n \) in \( \text{ground}(P) \),
\[
I \models B_1, \ldots, B_n \quad \text{implies} \quad \text{for } i \in [1, n]: \ |A| > |B_i| .
\]

Bounded programs are defined by discarding the proof obligation (b) in Definition 3.3. While recurrent and acceptable programs are included in the class of fair-bounded programs, we have that fair-bounded programs are a special case of bounded programs (apart from the fact that level mappings \( |: B_P \rightarrow N \) into naturals are considered in [17]).

8. Inferring fair-boundedness

On a theoretical level, the problem of deciding whether a program is fair-bounded is undecidable.

Theorem 8.1. It is undecidable whether there exist \( | \) and \( I \) such that a program \( P \) and a goal \( G \) are both fair-bounded by \( | \) and \( I \).

Proof. By the Correctness and Completeness Theorems 4.6 and 5.6, the problem is decidable iff it is decidable whether \( P \) and \( G \exists \)-universally terminate. In [7, Theorem 8], it is shown that it is undecidable whether given a program consisting of one clause of the form
\[
p(T_1, \ldots, T_n) \leftarrow p(S_1, \ldots, S_n)
\]
and a goal \( \leftarrow p(V_1, \ldots, V_n) \), the SLD-resolution stops. The particular form (8) of clauses implies that there is only one SLD-derivation for the program and the goal. As a consequence, it is undecidable whether they \( \exists \)-universally terminate, and \textit{a fortiori} whether they are fair-bounded. \( \Box \)
On a practical level, however, by the Termination Completeness Theorem 5.6 fair-bounded programs and goals include every class of programs and goals that universally terminate via some selection rule. Therefore, any existing automatic tool for proving universal termination of \( P \) and \( G \) is sufficient for proving that \( P \) and \( G \) are fair-bounded. This fact allows us to reuse all existing automatic tools and termination proofs to the purpose of showing fair-boundedness.

Apart from this consideration, we argue that existing tools could be adapted for proving the proof obligations of Definition 3.3 directly. In fact, those proof obligations are variations of proof obligations of well-known methods, such as recurrency and acceptability, for which automatic tools already exist.

9. Related work

A comprehensive survey on termination of logic programs can be found in the paper by De Schreye and Decorte [6]. They classify three types of approaches: techniques that express necessary and sufficient conditions for termination, techniques that provide decidable sufficient conditions, and techniques that prove decidability or undecidability for subclasses of programs and goals. Under this classification, this paper falls in the first type.

Until recently, termination analysis has been focused on fixed selection rules, and in particular Prolog’s leftmost selection rule. In the following, we recall some recent proposals dealing with selection rules other than the leftmost one.

Speirs et al. [20] present the static termination analysis algorithm of Mercury, which infers termination with respect to the leftmost selection rule for a program obtained by permuting the body atoms of the original program. Marchiori and Teusink [12] propose a sufficient termination method for the so-called local selection rules, i.e. rules that resolve completely an atom in a goal before starting resolution of the other atoms. Martin and King [13] present a transformational approach for termination of Gödel programs which relies on an run-time analysis of the length of derivations via semi-local selection rules, a weakening of local selection rules. The basic idea is not to proceed searching along a derivation via any selection rule which is longer than the depth of SLD-trees via semi-local selection rules. Krishna Rao et al. [18] propose a method for proving termination of GHC programs by transforming a GHC program into a term rewriting system, and then applying well-known termination techniques for the paradigm of term rewriting systems. GHC is a concurrent logic language where an atom and a clause can be resolved only if the atom is an instance of the clause’s head. Also, Plümer [16] reasons on termination of GHC programs by relating GHC derivations to the leftmost selection rule, and the using the techniques proposed in [15].

Except for the proposal of Martin and King, the cited approaches impose hypotheses that prevent or restrict full coroutining executions, such as in the producer–consumer program \texttt{ProdCons} of Example 3.5. On the contrary, we observe that fair selection rules allow for full coroutining and parallel executions. We refer the reader to Naish
[14] for a discussion about subtleties and problems of termination via coroutining. He discusses how terminating programs can be composed by disjunction, conjunction and recursion to form terminating programs in presence of coroutining.

We conclude this brief overview by mentioning a few fundamental studies on properties of fair selection rules in SLD-resolution due to Apt and van Emden [3], Lassez and Maher [10] and van Emden and Nait Abdallah [23].

10. Conclusions

In this paper, we have introduced the notion of $\exists$-universal termination of logic programs and goals. We have claimed that it is an essential concept for declarative programming, where a crucial point is to associate a complete control strategy to programs and goals.

We have shown that $\exists$-universal termination and universal termination via fair selection rules coincide. Then we offered a declarative characterization of $\exists$-universal termination by means of the notions of fair-bounded programs and goals, which provide us with a correct and complete method for proving termination. The method is purely declarative, modular, simple and practical in paper & pencil proofs. Moreover, by the Termination Completeness Theorem 5.6 any existing (possibly automatic) method for proving universal termination is a sufficient method for proving fair-boundedness.

Acknowledgements

I am grateful to F. Mesnard and D. Pedreschi for having stimulated the development of this paper. Also, thanks are due to S. Contiero and M. Gabbrielli for comments on a draft.

References


