Is there an axiomatic semantics for standard pure Prolog?

François Denis and Jean-Paul Delahaye

Abstract


Many studies [1, 7, 20, 21, 26, 28] have shown the soundness and completeness of SLD-resolution and of the finite failure rule for a definite program. This semantics corresponds to the operational semantics of an (ideal) Prolog system, i.e. which only generates fair SLD-trees and employs a breadth-first search rule. Unfortunately, the Prolog systems currently used may generate non-fair SLD-trees and employ a depth-first-left-right strategy which is no longer complete. For these standard Prolog implementations, the operational semantics of a program depends not only on its logic content but also on the way it is written. In this work, we introduce two systems of axioms associated to a definite program $P$:

- the finite standard translation $A_{xf}$ which provides a logic characterization of the finite operational semantics of a definite program $P$ computed by a Standard Prolog System and does not depend on the ordering given to the clauses in the program $P$.
- the finite invariant translation $A_{xi}$ which gives a logic characterization of the success and the failures of $P$, all SLD-trees of which are finite. This translation does not depend on the ordering given to the clauses in the program and to the atoms in each clause.

1. Introduction

Many studies [1, 7, 20, 21, 26, 28] have shown that SLD-resolution and the negation as failure rule are sound and complete with Clark’s completion of a definite program $P$ (denoted in [28] by $\text{comp}(P)$). We use in this paper an extension of the language of the original program $P$ by adding a new predicate symbol $q'$ for every
predicate symbol $q$ of $P$ [5, 9, 11, 18] and we denote by $\Lambda x_{\text{diff}}$ a new axiomatic extension of $P$ such that, for every ground positive literal $A$,
\[
\text{comp}(P) \vdash A \iff \Lambda x_{\text{diff}} \vdash A,
\]
\[
\text{comp}(P) \vdash \neg A \iff \Lambda x_{\text{diff}} \vdash A'.
\]

The previous classical results can be stated then in the following way: let $Q$ be a positive atomic query,
- if $Q$ succeeds under some computation rule with answer $\theta$ then $\Lambda x_{\text{diff}} \vdash \forall Q\theta$,
- if $\Lambda x_{\text{diff}} \vdash \forall Q\theta$ then $Q$ succeeds under all computation rules with an answer including $\theta$,
- if $Q$ finitely fails under some computation rule then $\Lambda x_{\text{diff}} \vdash \forall Q'$,
- if $\Lambda x_{\text{diff}} \vdash \forall Q'$ then $Q$ finitely fails under all fair computation rules.

Thus an Ideal Prolog System which generates only fair SLD-trees and employs a breadth-first search rule is complete wrt $\Lambda x_{\text{diff}}$. The ideal of logic programming, i.e. only specify the logic component of a problem and let the procedural aspects of the program's execution be handled by the system [28], seems to be reached for problems which can be expressed by definite programs.

But this implementation is very inefficient and the Prolog systems currently used may generate non-fair SLD-trees and employ a depth-first-left-right strategy which is no longer complete. For these standard Prolog implementations, the operational semantics of a program depends not only on its logic content but also on the way it is written. This fact has often been taken into account for normal programs (i.e. with negative literals in the body of the clauses) [3, 25, 28, 29, 31, 34] and it seems a little strange that, for definite programs, we are satisfied with the description of an ideal situation which is practically never encountered.

Operational semantics of Standard Prolog Systems has already been described in [2, 4, 6, 8, 17, 22, 32] but not in an axiomatic way. A first axiomatic translation of a definite program has been proposed in [9] with the aim of giving an axiomatic description of what is really computed by a Standard Prolog System. This axiomatic is complete only for some special classes of programs which have been studied in [11]. But these classes of programs are rather restrictive and this is why in this paper we try to go further. This leads us to introduce two new axiomatic translations of a definite program $P$ aiming to grasp the operational semantics of pure standard Prolog:
- the finite standard translation $Ax_{\text{fst}}$ which provides a logic characterization of the finite operational semantics of a definite program $P$ computed by a Standard Prolog System, i.e. $\{A: A$ is a ground atom, $\leftarrow A$ succeeds and has a finite standard SLD tree$\} \cup \{A': A$ is a ground atom and $\leftarrow A$ has a finitely failed standard SLD-tree$\}$. This translation does not depend on the ordering given to the clauses in the program $P$.
- the finite invariant translation $Ax_f$. More restrictive but also more homogeneous than $Ax_{\text{fst}}$, it gives a logic characterization of the success and the failures of $P$, all
SLD-trees of which are finite [30]. It describes what a user may expect from a
definite program $P$ when he has neither information on the computation rule used
by the Prolog System nor on the strategy employed to search SLD-trees generated
by the computation rule. This translation does not depend on the ordering given to
the clauses in the program and to the atoms in each clause.

By logic characterization, we mean least Herbrand model in the best case and an
$\omega$-iteration $T^{\uparrow}\omega$, for the operator $T$ associated in the usual way with the system of
axioms, in the general case. $T^{\uparrow}\omega$ seems to be a "computable version" of the least
Herbrand model. We prove that for all the axiomatic translations considered in this
paper, $T^{\uparrow}\omega$ is the least Herbrand model for program without local variables.

For "good" programs (i.e. for which every computation is finite, whatever rule
and strategy are used), all these semantics are logically equivalent. In the general
case, the finite operational semantics of a program (the only ones that have a
practical meaning) are provided by $Ax_{fst}$ and $Ax_r$.

Thus if the intended logic meaning of a definite program $P$ is provided by $Ax_{diff}$,
and natural operational meanings of $P$ are described by $Ax_{fst}$ or $Ax_r$, we can say
that the work of a (pure) Prolog programmer is to write programs such that $Ax_{diff}$
and $Ax_{fst}$ (or $Ax_r$) are logically equivalent.

This paper is a modified and completed version of [10].

2. Preliminaries

We use in this paper a "predicates symbol doubling technique" which has already
been used in [5, 9, 18] in logic programming works but also in [16, 19] in the study
of semantical and mathematical paradoxes. This technique, as trivalued logic, allows
us to express negation as failure more closely than classical logic does.

Let $P$ be a Prolog program.
- $l(P)$ or $l$ denotes the first order language associated with $P$.
- $l'(P)$ or $l'$ denotes the first order language associated with $P$ in which $p'$ has been
  substituted for every predicate symbol $p$.
- $h(P)$ or $h$ (resp. $h'(P)$ or $h'$) denotes the Herbrand base associated with $l$ (i.e.
  the set of all ground atoms of $l$) (resp. $l'$).
- If $E$ is a subset of $h(P)$, let $E' = \{A' \in h'(P) | A \in E\}$.

In the following, we consider axioms $Ax$ based on $l$ or $l \cup l'$ associated with a
definite Prolog program.

The Herbrand interpretation of $l$ (resp. $l \cup l'$) is a subset of $h$ (resp. $h \cup h'$) with
constants assigned to themselves and every $n$-ary function symbol assigned to the
mapping $(t_1, \ldots, t_n) \rightarrow f(t_1, \ldots, t_n)$. It is a free interpretation with equality assigned
to identity [34]. Truth value of a formula wrt an interpretation and notion of model
for a formula are defined as usual [28]. The standard SLD tree for $P \cup \{G\}$ is the
SLD tree for $P \cup \{G\}$ coming from the computation rule which always selects the
leftmost atom. The standard depth of a goal $G$ (wrt a definite program $P$) is the
length of the longest branch of the standard SLD tree for \( P \cup \{ G \} \). The *invariant depth* of a goal \( G \) (wrt a definite program \( P \)) is the length of the longest branch of every SLD tree for \( P \cup \{ G \} \). We can show, using Koenig's lemma, that if all the SLD trees of \( P \cup \{ G \} \) are finite then the invariant depth of \( G \) is finite. A SLD tree is *fair* if for every infinite branch on the tree, every atom \( A \) of the branch (or some further instantiated version of \( A \)) is selected within a finite number of steps [26, 28]. An *Ideal Prolog System* only generates fair SLD trees and employs a breadth-first search rule. A *Standard Prolog System* generates standard SLD trees (which are not fair in general) and employs a depth-first search rule.

**Definition 2.1** (Apt and Van Emde [1], Delahaye [9], Denis [11], Lloyd [28]). We think that there are at least four natural operational semantics related to a definite program.

(a) *The operational semantics of an Ideal Prolog*: \( SS \cup FF' \)

- \( SS = \{ A \in h(P); P \cup \{ - A \} \text{ has an SLD-refutation} \} \). It is the success set of an Ideal Prolog System.
- \( FF = \{ A \in h(P); \text{there exists a finitely failed SLD tree for } P \cup \{ - A \} \} \). It is the finite failure set of an Ideal Prolog System.

The operational semantics of the program \( P \) computed by an Ideal Prolog System may be represented by \( SS \cup FF' \). This is the operational semantics which is usually studied.

(b) *The operational semantics of a Standard Prolog*: \( SS_{st} \cup FF'_{st} \)

- \( SS_{st} = \{ A \in SS; \text{the standard SLD tree for } P \cup \{ - A \} \text{ has a success branch on the left of which every branch is finite} \} \). It is the success set of a Standard Prolog System.
- \( FF_{st} = \{ A \in FF; \text{the standard SLD tree for } P \cup \{ - A \} \text{ is finite} \} \). It is the finite failure set of a Standard Prolog System.

The operational semantics of the program \( P \) computed by a Standard Prolog System may be represented by \( SS_{st} \cup FF'_{st} \). This semantics has been investigated in [9, 11].

(c) *The finite operational semantics of a Standard Prolog*: \( SS_{fst} \cup FF'_{fst} \)

- \( SS_{fst} = \{ A \in SS_{st}; \text{the standard SLD tree for } P \cup \{ - A \} \text{ is finite} \} \). It is the finite *success set* of a Standard Prolog System.

\( SS_{fst} \cup FF'_{fst} \) is the *finite standard operational semantics* of \( P \). It does not depend on the ordering given to the clauses in the program.

(d) *The finite invariant operational semantics of Prolog*: \( SS_{f} \cup FF'_{f} \)

- \( SS_{f} = \{ A \in SS; \text{all SLD trees for } P \cup \{ - A \} \text{ are finite} \} \). An atom \( A \in SS_{f} \) iff \( A \) is in the success set of all Prolog Systems.
- \( FF_{f} \) is the set of all \( A \in FF \) such that all SLD trees for \( P \cup \{ - A \} \) are finite. An atom \( A \in FF_{f} \) iff \( A \) is in the finite failure set of all Prolog Systems.

\( SS_{f} \cup FF'_{f} \) is the *finite invariant operational semantics* of \( P \). It does not depend on the ordering given to the clauses in the program and to the atoms in each clause. It represents what a user may expect from a definite program \( P \) when he wants his
program to work in a satisfactory manner whatever computation rule is used by the Prolog System, and whatever strategy is employed to search SLD trees.

The last two semantics are very natural and the main aim of this paper is to obtain axiomatic definitions for them.

- \( A \in \text{FF}_{st}^n \) (resp. \( \text{SS}_{fst}^n \)) iff \( A \in \text{FF}_{st} \) (resp. \( \text{SS}_{fst} \)) and standard depth of \( \leftarrow A \) is \( \leq n \).
- \( A \in \text{FF}_r^? \) (resp. \( \text{SS}_{r}^? \)) iff \( A \in \text{FF}_r \) (resp. \( \text{SS}_r \)) and invariant depth of \( \leftarrow A \) is \( \leq n \).

The next example shows that all the previous defined sets may be different.

**Example 2.2**

\[
\begin{align*}
A \leftarrow & \quad \text{SS}_r = \{B(b), D(b)\}, \\
A \leftarrow A & \quad \text{SS}_{fst} = \{B(b), D(b), C\}, \\
B(a) \leftarrow B(a) & \quad \text{SS}_{st} = \{B(b), D(b), C, A\}, \\
B(x) \leftarrow & \quad \text{SS} = \{B(b), D(b), C, A, B(a)\}, \\
C \leftarrow D(x), B(x) & \quad \text{FF}_r = \{D(a), G\}, \\
D(b) \leftarrow & \quad \text{FF}_{st} = \{D(a), G, F\}, \\
E \leftarrow A, G & \quad \text{FF} = \{D(a), G, F, E\}.
\end{align*}
\]

We obtain immediately from the previous definitions that for every definite program \( P \),

\[
\text{SS}_r \subseteq \text{SS}_{fst} \subseteq \text{SS}_{st} \subseteq \text{SS} \subseteq h(P) \setminus \text{FF} \subseteq h(P) \setminus \text{FF}_{st} \subseteq h(P) \setminus \text{FF}_r.
\]

\( \text{SS}_{st} \) is generally different from \( \text{SS} \) because, with a depth-first search rule, a success branch on the standard SLD tree may not be found. \( \text{SS}_{fst} \) is different from \( \text{SS}_{st} \) if \( \text{SS}_{st} \) contains an atom standard SLD tree of which is infinite. \( \text{SS}_r \) is different from \( \text{SS}_{fst} \) if \( \text{SS}_{fst} \) contains an atom some SLD tree of which is infinite.

\( \text{FF}_{st} \) is generally different from \( \text{FF} \) because a standard SLD tree may not be fair. \( \text{FF}_r \) is different from \( \text{FF}_{st} \) if \( \text{FF}_{st} \) contains an atom some SLD tree of which is not fair.

**Remark 2.3.** (i) Practically, these finite semantics are not very restrictive for a (pure) Prolog programmer trying to write definite programs which are supposed to verify \( \text{SS} = \text{SS}_{fst} \) and \( \text{FF} = \text{FF}_r \) (and better still, \( \text{SS} = \text{SS}_r \)). This property insures that the classical intended meaning of the program corresponds to what is really computed by standard implementations of Prolog. Hierarchical programs, and in a general way, all programs for which it has been proved that all computations are finite, have this property.

(ii) Finite standard semantics does not depend on the ordering given to the clauses in the program, but the converse is false. That is, there exist a program \( P \) and a ground atom \( A \) such that \( A \in \text{SS}_{st} \) whatever ordering is given to the clauses and \( A \notin \text{SS}_{fst} \) [12].
2.1. Operator associated with a set of axioms

Let Ax be a conjunction of formulas of a language I of the form

\[ p(x_1, \ldots, x_n) \iff F(x_1, \ldots, x_n) \]

where \( p \) is a predicate symbol of \( I \) and \( F \) is a formula of \( I \), with free variables among \( x_1, \ldots, x_n \).

The operator \( T_{Ax} \) associated with Ax is defined as follows:

\[ T_{Ax} : 2^{h(P)} \rightarrow 2^{h(P)} \]

\[ I \rightarrow \{ A\theta \in h(P) : A \iff F \text{ is an axiom of Ax, } \theta \text{ is a substitution such that } (A \iff F)\theta \text{ is ground, } F\theta \text{ is true in } I \} \]

2.2. Axiomatic translations from a program

Let us consider an example. Let \( P \) be the program

\[
\begin{align*}
A & \iff A \\
A & \iff A \\
B(a) & \iff \\
C(x) & \iff A, B(s(y))
\end{align*}
\]

IF translation [1]: \( Ax_{IF} \) is the following system of axioms:

\[
\begin{align*}
A & \iff \text{True} \lor A \\
B(x) & \iff x = a \\
C(x) & \iff \exists y (A \land B(s(y)))
\end{align*}
\]

Remark 2.4. We make a distinction in this paper between "\( \iff \)" which denotes the Prolog symbol separating the head and the body of a clause and "\( \iff \)" which denotes the classical implication symbol.

Let \( T_{IF} \) be the operator associated with \( Ax_{IF} \). The least Herbrand model for \( Ax_{IF} \) (which is also the least fixpoint of \( T_{IF} \)) is equal to SS [1, 28, 9]. Thus, this translation provides an axiomatic and denotational characterization of the success of an Ideal Prolog System.

IFF translation: \( Ax_{IFF} \) (Clark's completion [7] denoted by \( Ax_{y} \) in [9], comp(\( P \)) in [28] and CDB in [34]) is the following system of axioms:

\[
\begin{align*}
A & \iff \text{True} \lor A \\
B(x) & \iff x = a \\
C(x) & \iff \exists y (A \land B(s(y)))
\end{align*}
\]

The least Herbrand model for \( Ax_{IFF} \) is SS and its greatest Herbrand model is included in \( h(P) \backslash FF \). This translation allows us to take finite failure into account [1, 7, 21, 28].
Double IFF translation: \( \text{Ax}_{\text{diff}} \) is the conjunction of the following axioms:

\[
\begin{align*}
A &\iff \text{True} \lor A \\
A' &\iff \text{False} \land A' \\
B(x) &\iff x = a \\
B'(x) &\iff x \neq a \\
C(x) &\iff \exists y (A \land B(s(y))) \\
C'(x) &\iff \forall y (A' \lor B'(s(y)))
\end{align*}
\]

If, in this translation, \( \neg p \) is substituted for every predicate symbol \( p' \), a system of axioms equivalent to \( \text{Ax}_{\text{diff}} \) is obtained.

Introduced by Delahaye [9] (where it is denoted by \( \text{Ax}_d \)), this translation is an attempt to express Negation as Failure in classical logic. It has been generalized for programs with negation in [5, 18]. This translation permits us to characterize \( \text{SS} \cup \text{FF}' \) as an ordinal power of \( \text{T}_{\text{diff}} \).

Let \( \text{T}_{\text{diff}} \) be the associated operator, it has been proved in [9] that

\[
\text{SS} \cup \text{FF}' = \text{T}_{\text{diff}} \uparrow \omega.
\]

Another axiomatic translation \( \text{Ax}_{\text{st}} \) has been proposed in [9] (where it is denoted by \( \text{Ax}_d \)) in order to characterize the operational semantics of a Standard Prolog. We have proved in [11] that this characterization is obtained only for programs which verify some strong hypothesis.

One of the purposes of this paper is to propose new systems of axioms in order to characterize more closely the operational semantics of a Standard Prolog System.

3. Finite standard axiomatic

Let us suppose now that all goals are computed by a Standard Prolog System. The word depth refers to standard depth in this section.

Consider a Prolog program without variable, in which clauses having \( A \) as head predicate symbol are (in this order)

\[
\begin{align*}
A &\leftarrow B, C \\
A &\leftarrow D
\end{align*}
\]

\[
\begin{align*}
[\leftarrow A \text{ succeeds}] &\iff [\leftarrow B, C \text{ succeeds or } (\leftarrow B, C \text{ finitely fails and } \leftarrow D \text{ succeeds})]. \\
[\leftarrow A \text{ finitely fails}] &\iff [\leftarrow B, C \text{ finitely fails and } \leftarrow D \text{ finitely fails}. \\
[\leftarrow B, C \text{ succeeds}] &\iff [\leftarrow B \text{ succeeds and } \leftarrow C \text{ succeeds}. \\
[\leftarrow B, C \text{ finitely fails}] &\iff [\leftarrow B \text{ finitely fails or } (\leftarrow B \text{ finitely succeeds and } \leftarrow C \text{ finitely fails})].
\end{align*}
\]
The notion of \textit{finite standard success} appears naturally. The following recursive definition can be given for it (in this example):

\[
\text{[\textasciitilde} A \text{~finately succeeds]} \text{iff } \{(\text{\textasciitilde} B, C \text{~finately succeeds and} \text{\textasciitilde} D \text{~is finite}) \text{ or } (\text{\textasciitilde} B, C \text{~finately fails and} \text{\textasciitilde} D \text{~finately succeeds})\}
\]

i.e. in an equivalent way

\[
\text{[\textasciitilde} A \text{~finately succeeds]} \text{iff } \{(\text{\textasciitilde} B, C \text{~and} \text{\textasciitilde} D \text{~are finite}) \text{ and } (\text{\textasciitilde} B, C \text{~finately succeeds or} \text{\textasciitilde} D \text{~finately succeeds})\].
\]

\[\text{[\textasciitilde} B, C \text{~finately succeeds]} \text{iff } [\text{\textasciitilde} B \text{ and} \text{\textasciitilde} C \text{~finately succeed}].\]

The notion of \textit{finite goal} is characterized by

\[\text{[\textasciitilde} A \text{~is finite]} \text{iff } \{(\text{\textasciitilde} B \text{~finately fails or} (\text{\textasciitilde} B \text{~finately succeeds and} \text{\textasciitilde} C \text{~is finite}) \text{ and } (\text{\textasciitilde} D \text{~is finite})\}\]
A system of axioms \( \text{Ax}_{\text{fist}} \), defined on the language \( l \cup l' \), is associated with a program \( P \) in the following way: let \( C_1, C_2, \ldots, C_p \) be the clauses of \( P \) with head predicate symbol \( p \). Consider the following two formulas:

\[
\begin{align*}
\text{p}(x) & \iff (C_1^F \land \cdots \land C_p^F) \land (C_1^S \lor \cdots \lor C_p^S), \\
\text{p}'(x) & \iff (C_1^F \land \cdots \land C_p^F).
\end{align*}
\]

\( \text{Ax}_{\text{fist}} \) is the conjunction of these formulas, for every predicate symbol \( p \) of the program.

**Example 3.3.**

\[
P = \begin{cases}
A \\
A \leftarrow A \\
B(s(x)) \\
C \leftarrow A, B(a)
\end{cases}
\]

\( \text{Ax}_{\text{fist}}(P) \) (after simplifications)

\[
\begin{align*}
B(x) & \iff \exists y \ (x = s(y)), \\
B'(x) & \iff \forall y \ (x \neq s(y)), \\
C & \iff A \land B(a), \\
C' & \iff A' \lor (A \land B'(a)).
\end{align*}
\]

Let \( T_{\text{fist}} \) be the operator associated with this system of axioms, following the general definition given previously.

**Example 3.4.** \( T_{\text{fist}}[1] = \{B'(a), B(s^n(a)) \}, n \geq 1 \) = least fixpoint of \( T_{\text{fist}} = SS_{\text{fist}} \cup FF'_{\text{fist}} \).

**Proposition 3.5.**

(a) \( T_{\text{fist}} \) is monoton.

(b) for every ordinal \( \alpha \) and ground atom \( A, A' \) cannot be simultaneously in \( T_{\text{fist}} \uparrow \alpha \).

**Proof.** (a) See [9] for a proof in a general case.

(b) Straightforward using an induction on \( \alpha \).

**Lemma 3.6.** Consider the goal \( \leftarrow L \) of depth \( \leq n \). Suppose that \( SS_{\text{fist}}^n \cup (FF_{\text{fist}}^n)' \subset T_{\text{fist}} \uparrow n \). Then,

(a) if \( \leftarrow L \) finitely succeeds, then \( \exists[L]_{\text{fist}}^F \) is true in \( T_{\text{fist}} \uparrow n \),

(b) if \( \leftarrow L \) finitely fails, then \( \forall[L]_{\text{fist}}^F \) is true in \( T_{\text{fist}} \uparrow n \),

(c) \( \forall[L]_{\text{fist}}^F \) is true in \( T_{\text{fist}} \uparrow n \),

where \( \exists F \) and \( \forall F \) denote, respectively, the existential and universal closure of the formula \( F \).

**Proof.** (by induction on \( q \)). This is obvious if \( q = 0 \). Suppose that the result is true for \( q - 1 \) \( (q \geq 1) \), and let \( L \) (resp. \( L' \)) be the list \( A_1, A_2, \ldots, A_q \) (resp. \( A_2, \ldots, A_q \)) and \( G \) be the goal \( \leftarrow L \).
(a) There is a ground instance $G\sigma$ of $G$ which succeeds. Depth of $G\sigma$ is $\leq n$. $A_1\sigma \in \mathbb{SS}_{\text{fst}}^n$ then $A_1\sigma \in T_{\text{fst}}\uparrow n$ and since $\leftarrow L'\sigma$ succeeds and has a depth $\leq n$, then from the induction hypothesis, $[L'\sigma]^{\text{FS}}_{\text{fst}}$ is true in $T_{\text{fst}}\uparrow n$ and, by definition, $[L\sigma]^{\text{FS}}_{\text{fst}}$ and therefore $\exists [L]^{\text{FS}}_{\text{fst}}$ are true in $T_{\text{fst}}\uparrow n$.

(b) Every ground instance $G\sigma$ of $G$ fails and has a depth $\leq n$.

- If $A_1\sigma$ fails, then $A_1\sigma \in T_{\text{fst}}\uparrow n$ and, by definition, $[L\sigma]^{\text{FS}}_{\text{fst}}$ is true in $T_{\text{fst}}\uparrow n$.
- If $A_1\sigma$ succeeds, then $A_1\sigma \in T_{\text{fst}}\uparrow n$. The goal $\leftarrow L'\sigma$ fails and $[L'\sigma]^{\text{FS}}_{\text{fst}}$ is true in $T_{\text{fst}}\uparrow n$ from the induction hypothesis. Then, $A_1\sigma \land [L'\sigma]^T_{\text{fst}}$ and therefore $[L\sigma]^T_{\text{fst}}$ are true in $T_{\text{fst}}\uparrow n$.

This shows that $\forall [L]^T_{\text{fst}}$ is true in $T_{\text{fst}}\uparrow n$.

(c) Similar proof. \(\square\)

Lemma 3.7.

\[ \mathbb{SS}_{\text{fst}}^n \cup (\mathbb{FF}_{\text{fst}})^n \subseteq T_{\text{fst}}\uparrow n. \] (1)

Proof (by induction on $n$). Obvious if $n = 0$. Prove for example that $\mathbb{SS}_{\text{fst}}^{n+1} \subseteq T_{\text{fst}}\uparrow (n+1)$ if relation (1) is supposed for $\alpha$. Let $p(t) \in \mathbb{SS}_{\text{fst}}^{n+1}$ and $C = p(s) \leftarrow L$ be a clause of $P$.

- If $p(t)$ and $p(s)$ are not unifiable, then $C^F(t)$ is true (in $T_{\text{fst}}\uparrow n$).
- If $p(t)$ and $p(s)$ are unifiable, and $\sigma = \text{mgu}(p(t), p(s))$, then

  - $\leftarrow L\sigma$ has a depth $\leq n$ and then, by the previous lemma, $\forall [L\sigma]^T_{\text{fst}}$ is true in $T_{\text{fst}}\uparrow n$. $C^S(t)$ is therefore true in $T_{\text{fst}}\uparrow n$.
  - if $\leftarrow L\sigma$ finitely succeeds (and this is the case for at least one clause of $P$), then $\exists [L\sigma]^T_{\text{fst}}$ is true in $T_{\text{fst}}\uparrow n$ and $C^S(t)$ is true in $T_{\text{fst}}\uparrow n$. \(\square\)

In conclusion, if $C_1, C_2, \ldots, C_p$ are the clauses of $P$ with head predicate symbol $p$, the formula $(C_1^F \land \cdots \land C_p^F) \land (C_1^S \lor \cdots \lor C_p^S)$ with $t$ as argument is true in $T_{\text{fst}}\uparrow n$ and $A(t) \in T_{\text{fst}}\uparrow (n+1)$.

Theorem 3.8. (soundness of standard resolution wrt finite standard axiomatic)

\[ \mathbb{SS}_{\text{fst}} \cup \mathbb{FF}_{\text{fst}} \subseteq T_{\text{fst}}\uparrow \omega. \]

Proof. Follows from previous lemmas. \(\square\)

Lemma 3.9. Suppose that for every integer $n$ there is an integer $d_n$ such that for every ground atom $A$, $A \in T_{\text{fst}}\uparrow n$ or $A' \in T_{\text{fst}}\uparrow n$ implies that depth($\leftarrow A$) $\leq d_n$. Then,

(a) For any list $L = A_1, A_2, \ldots, A_q$ of ground atoms, $[L]^F_{\text{fst}}$ is true in $T_{\text{fst}}\uparrow n$ implies that depth($\leftarrow L$) $\leq qd_q$.

(b) Suppose that there exists an integer $N$ such that the depth of all ground instances of $\leftarrow L\sigma$ is $\leq N$. Then depth($\leftarrow L$) $\leq N$.

(c) Let $A$ be a ground atom. If for every clause $B \leftarrow L$ in $P$ such that $A$ and $B$ are unifiable with $\sigma$ as mgu, $\forall [L\sigma]^T_{\text{fst}}$ is true in $\text{fst}\uparrow n$, then depth($\leftarrow A$) $\leq Md_n + 1$ where $M$ is the maximal number of atoms occurring in the body of a clause of $P$. 


Proof. (a) Prove it by induction on \( q \). This is obvious if \( q = 1 \). Let \( q > 2 \). \([L]^{\text{F}}_{\text{fst}}\) is true in \( T_{\text{fs}t} \uparrow n \) implies that \( A \in T_{\text{fs}t} \uparrow n \) or \( A' \in T_{\text{fs}t} \uparrow n \) and then depth(\( \leftarrow A \)) \( < d_n \). If \( \leftarrow A \) fails, then depth(\( \leftarrow A \)) = depth(\( \leftarrow A \)) < d_n \). If \( \leftarrow A \) succeeds then \( A \in T_{\text{fs}t} \uparrow n \) (by Lemma 3.7), \( A' \in T_{\text{fs}t} \uparrow n \) (by Proposition 3.5), and then \([A_2, \ldots, A_q]^{\text{F}}_{\text{fst}}\) is true in \( T_{\text{fs}t} \uparrow n \) (by definition of \([L]^{\text{F}}_{\text{fst}}\)) then depth(\( \leftarrow A_2, \ldots, A_q \)) \( \leq (q-1)d_n \) from the induction hypothesis. In conclusion, depth(\( \leftarrow A \)) \( \leq d_nq \).

(b) Suppose the conclusion does not hold. It is sufficient to instantiate a branch of length \( > N \) to obtain a contradiction.

(c) Let \( A \) be a ground atom such that for every clause \( B \leftarrow L \) in \( P \) such that \( A \) and \( B \) are unifiable with \( \sigma \) as mgu, \( \forall[L\sigma]^{\text{F}}_{\text{fst}} \) is true in \( T_{\text{fs}t} \uparrow n \). Previous lemmas prove that depth(\( \leftarrow L\sigma \)) \( \leq d_nq \leq d_nM \) and then, depth(\( \leftarrow A \)) \( \leq Md_n + 1 \).

Theorem 3.10. For every program \( P \), \( SS_{\text{fst}} \cup FF_{\text{fst}} = T_{\text{fst}} \uparrow \omega \).

Proof. More precisely, we prove that if \( d_n \) is the sequence defined by the induction relation \( d_{n+1} = Md_n + 1 \) where \( M \) is the maximal number of atoms occurring in the body of a clause of \( P \) and \( d_1 = 1 \) (an explicit definition of this sequence is \( d_n = (M^n - 1)/(M - 1) \) if \( M \neq 1 \), \( d_n = n \) if \( M = 1 \)), then \( A \in T_{\text{fs}t} \uparrow n \) or \( A' \in T_{\text{fs}t} \uparrow n \) implies that depth(\( \leftarrow A \)) \( \leq d_n \).

Let \( n = 1 \). The only clauses that can apply to \( A \) have an empty body. Thus, the result is true in this case. Suppose the proposition is true for an integer \( n \geq 1 \). Let \( A \) be a ground atom such that \( A \in T_{\text{fs}t} \uparrow (n + 1) \) or \( A' \in T_{\text{fs}t} \uparrow (n + 1) \). For every clause \( B \leftarrow L \) in \( P \) such that \( A \) and \( B \) are unifiable with \( \sigma \) as mgu, \( \forall[L\sigma]^{\text{F}}_{\text{fst}} \) or \( \forall[L\sigma]^{\text{FF}}_{\text{fst}} \) is true in \( T_{\text{fs}t} \uparrow n \). Remark 3.2 shows that \( \forall[L\sigma]^{\text{F}}_{\text{fst}} \) is true in \( T_{\text{fs}t} \uparrow n \) in both cases and then depth(\( \leftarrow A \)) \( \leq Md_n + 1 = d_{n+1} \) by the previous lemma.

Theorem 3.10 provides a sound and complete denotational characterization of the finite standard semantics of a definite program.

4. Finite invariant axiomatic

The Finite Standard Semantics of a definite program \( P \) does not depend on the ordering of the clause in \( P \) but depends on the ordering of the atoms in each clause of \( P \). It is well adapted to a Prolog system which uses a left most atom computation rule and a depth first search rule. The Finite Invariant Operational Semantics of a definite program is more homogeneous. It is adapted to any Prolog system using a depth first strategy, whatever computation rule it employs.

In this section we give an axiomatic characterization of this semantics. All proofs are omitted here since they consist of straightforward adaptations of previous ones, the notion of standard depth being replaced by invariant depth.

Let \( I = A_1, A_2, \ldots, A_n \) be a list of atoms, define:

\[
[L]^{\text{F}} = A_1 \land A_2 \land \cdots \land A_n \text{ and } [L]^{\text{FS}} = \text{True}.
\]
\( [L]^F = (A_1 \lor A_1) \land \cdots \land (A_n \lor A_n) \) and \( [L]^F = \text{True} \).

\( [L]^F = \text{true} \land (A_1 \lor \cdots \lor A_n) \) and \( [L]^F = \text{false} \).

**Remark 4.1.** Let \( X \subset h \cup h' \) and \( L \) be a list of ground atoms:

- \( [L]^F \) true in \( X \) iff \( [L]^F \) or \( [L]^F \) is true in \( X \)
- \( [L]^{SF} \) is true in \( X \) implies that \( [L]^{SF} \) is true in \( X \)
- \( [L]^{SF} \) is true in \( X \) implies that \( [L]^{SF} \) is true in \( X \)

As in the previous section, we associate with a definite clause \( C = A(t) \leftarrow L \) the following formulas:

\[
C^F(x) = [\forall y_1 \ldots y_m (x \neq t \lor [L]^F)]
\]
\[
C^T(x) = [\forall y_1 \ldots y_m (x \neq t \lor [L]^F)]
\]
\[
C^S(x) = [\exists y_1 \ldots y_m (x = t \land [L]^F)]
\]

in which \( y_i \) are variables occurring in the terms \( t_i \) and atoms \( A_i \) and \( x_i \) are new variables.

A system of axioms \( Ax_r \), defined on the language \( l \cup l' \), is associated with a program \( P \) in the following way. Let \( C_1, C_2, \ldots, C_p \) be the clauses of \( P \) with head predicate symbol \( p \). Write the conjunction of the following formulas:

\[
p(x) \leftarrow (C^F_1 \land \cdots \land C^F_p) \land (C^S_1 \lor \cdots \lor C^S_p)
\]
\[
p'(x) \leftarrow (C^{SF}_1 \land \cdots \land C^{SF}_p)
\]

\( Ax_r \) is the conjunction of these formulas for every predicate symbol \( p \) of the program.

**Example 4.2.**

\[
\begin{align*}
P &= \begin{cases}
A, \\
A \leftarrow A, \\
B(s(x)), \\
C \leftarrow B(a), A.
\end{cases}
\end{align*}
\]

\( Ax_r(P) \) (after simplifications)

\[
\begin{align*}
B(x) &\iff \exists y \ (x = s(y)), \\
B'(x) &\iff \forall y \ (x \neq s(y)), \\
C &\iff A \land B(a), \\
C' &\iff A' \lor B'(a).
\end{align*}
\]

Let \( T_r \) be the operator associated with this system of axioms.
Example 4.3. \( T_{f^1} = \{ R'(a), R(s''(a))n \geq 1 \} \) = least fixpoint of \( T_f = SS_f \cup FF_f \).

Proposition 4.4. (a) \( T_f \) is monotonic.
(b) For every ordinal \( \alpha \) and ground atom \( A \), \( A \) and \( A' \) cannot be simultaneously in \( T_f \rightarrow \alpha \).

Lemma 4.5. Consider the goal \( \leftarrow L \) of depth \( \leq n \). Suppose that \( SS_f \cup (FF_f)' \subseteq T_f \rightarrow n \). Then,
(a) \( \forall[L]^F \) is true in \( T_f \rightarrow n \),
(b) if \( \leftarrow L \) finitely succeeds, then \( \exists[L]^G \) is true in \( T_f \rightarrow n \),
(c) if \( \leftarrow L \) finitely fails, then \( \forall[L]^F \) is true in \( T_f \rightarrow n \).

Lemma 4.6.
\[ SS_f^n \cup (FF_f)' \subseteq T_f \rightarrow n. \quad (1) \]

Theorem 4.7. (soundness of standard resolution wrt finite invariant axiomatic)
\( SS_f \cup FF_f \subseteq T_f \rightarrow \omega \).

Lemma 4.8. Suppose that for every integer \( n \) there is an integer \( d_n \) such that for every ground atom \( A: A \in T_f \rightarrow n \) or \( A' \in T_f \rightarrow n \) implies that \( \text{depth}(\leftarrow A) \leq d_n \). Then,
(a) For any list \( L \) of ground atoms \( A_1, A_2, \ldots, A_q \), \( [L]^F \) is true in \( T_f \rightarrow n \) implies that \( \text{depth}(\leftarrow L) \leq qd_n \).
(b) Suppose that there exists an integer \( N \) such that the depth of all ground instances of \( \leftarrow L \) is \( \leq N \). Then \( \text{depth}(\leftarrow L) \leq N \).
(c) Let \( A \) be a ground atom. If for every clause \( B \leftarrow L \) in \( P \) such that \( A \) and \( B \) are unifiable with \( \sigma \) as mgu, \( \forall[L\sigma]^F \) is true in \( T_f \rightarrow n \), then \( \text{depth}(\leftarrow A) \leq Md_n + 1 \) where \( M \) is the maximal number of atoms occurring in the body of a clause of \( P \).

Theorem 4.9. For every program \( P \), \( SS_f \cup FF_f = T_f \rightarrow \omega \).

Theorem 4.9 provides a sound and complete denotational characterization of the finite semantics of a definite program.

Remark 4.10. Unfortunately, \( T_{f^1} \) and \( T_f \) are not always continuous. This is due to the presence of universal quantifiers in the body of the axioms of \( \text{Ax}_{f^1} \) and \( \text{Ax}_f \). Therefore, \( T_{f^1} \rightarrow \omega \) (resp. \( T_f \rightarrow \omega \)) is neither the least fixpoint of \( T_{f^1} \) (resp. \( T_f \)) nor the least Herbrand model of \( \text{Ax}_{f^1} \) (resp. \( \text{Ax}_f \)). Thus, \( T_{f^1} \rightarrow \omega \) (resp. \( T_f \rightarrow \omega \)) appears as a "computable version" of the least Herbrand model of \( \text{Ax}_{f^1} \) (resp. \( \text{Ax}_f \)).
Example 4.11.

\[
P(x) \iff \exists y \ (x = s(y) \land P(y)) \land \forall y \ (x \neq s(y) \lor P(y) \lor P'(y)),
\]

\[
P'(x) \iff \forall y \ (x \neq s(y) \lor P'(y)),
\]

\[
Q(x) \iff \exists y \ (x = a \land P(y)) \land \forall y \ (x \neq a \lor P(y) \lor P'(y)),
\]

\[
Q'(x) \iff \forall y \ (x \neq a \lor P'(y)).
\]

\[
T_{fs} \uparrow m = T_{f} \uparrow m = \left\{ P'(s^n(a)), Q'(s^n(a)) \right\}
\]

for every integer \( m \),

\[
T_{fs} \uparrow \omega = T_{f} \uparrow \omega = \left\{ P'(s^n(a)), Q'(s^n(a)) \right\},
\]

\[
T_{fs} \uparrow (\omega + 1) = T_{f} \uparrow (\omega + 1) = \left\{ P'(s^n(a)), Q'(s^n(a)) \right\} \neq T_{fs} \uparrow \omega.
\]

But, if \( P \) is a program such that no clause has local variables, i.e. every variable of every clause of \( P \) occurs in the head of the clause, then \( T_{fs} \) (resp. \( T_{f} \)) is continuous \([11]\). In this case, \( T_{fs} \uparrow \omega \) (resp. \( T_{f} \uparrow \omega \)) is the least fixpoint of \( T_{fs} \) (resp. \( T_{f} \)) and the least Herbrand model of \( Ax_{fa} \) (resp. \( Ax_{f} \)) which is therefore, a sound and complete axiomatic characterization of \( SS_{fa} \cup FF_{fa} \) (resp. \( SS_{f} \cup FF_{f} \)).

5. Conclusion

Finite standard operational semantics (resp. finite invariant operational semantics) is a natural description of the computations of a Standard Prolog System (resp. of any Prolog system based on SLD resolution). These semantics have a sound and complete axiomatic characterization if we only consider Prolog programs which have no local variable (and purely propositional programs, as a particular case).

In general, for any program, these semantics are described by an \( \omega \)-iteration \( T \uparrow \omega \) (where \( T \) is the classical operator associated with the corresponding system of axioms \( Ax \)) which appears as the "computable version" of the least Herbrand model for \( Ax \). This restriction is of the same kind as the one encountered in the study of the classic Finite Failure set \( FF \) of a definite program: \( FF = II \setminus T \downarrow \omega \) and \( T \downarrow \omega \) is not always a fixpoint of \( T \).

In conclusion, we bring the following answer to the question asked in the title: we obtained a fairly good axiomatic approximation of the computations of a Standard Prolog System (and even, any Prolog system). These results may be viewed as a more realistic version (i.e. concerning real Prolog systems, not ideal ones) of the classical results of Apt, Hill, Kowalski and Van Emden \([1, 20, 35]\). The Herbrand Model Theoretic framework seems to be too restrictive in order to obtain entirely satisfying results but the completeness results of \([21, 24]\) obtained in Clark's
Equational Theory framework leave the hope of obtaining more precise axiomatic characterization.

References