DECLARATIVE MODELING OF THE OPERATIONAL BEHAVIOR OF LOGIC LANGUAGES

M. FALASCHI, G. LEVI and C. PALAMIDESSI

Dipartimento di Informatica, Università di Pisa, Corso Italia 40, 56100 Pisa, Italy

M. MARTELLI

CNUCE—C.N.R., Via S. Maria 36, 56100 Pisa, Italy

Abstract. The paper defines a new declarative semantics for logic programs, which is based on interpretations containing (possibly) non-ground atoms. Two different interpretations are introduced and the corresponding models are defined and compared. The classical results on the Herbrand model semantics of logic programs are shown to hold in the new models too (i.e. existence of a minimal model, fixpoint characterization, etc.). With the new models, we have a stronger soundness and completeness result for SLD-resolution. In particular, one of the two models allows the set of computed answer substitutions to be characterized precisely.

Contents

1. Introduction

One of the nice features of logic programming has always been the correspondence between the declarative (model theoretic and fixpoint) semantics and the operational

semantics. In the case of ground atoms, this correspondence is complete. In fact, the minimal Herbrand model M of a program W is equal to its success set [6]. However, the notion of success set

$$SS = \{A \mid A \text{ is a ground atom and } \leftarrow A \text{ has a refutation}\}$$

is not completely adequate as operational semantics, since it hides one of the fundamental aspects of a logic program: the ability to compute substitutions. A more adequate definition should be

$$SS' = \{(A, \vartheta) \mid A \text{ is an atom and } \leftarrow A \text{ has a refutation with computed answer substitution } \vartheta\}.$$

Unfortunately, the correspondence of M with SS' does not hold any more. Even the stronger result of Clark [3], given in terms of general models, does not fully characterize this set declaratively.

We think it is important to fill this gap. In this paper we propose a semantic construction for logic programming, which is more adequate to describe the relations between the different kinds of semantics and is also able to deal with the meaning of universally quantified formulas. As we will show, one of our model-theoretic semantics characterizes exactly the set of computed answer substitutions and is therefore "equivalent" to SS'. The basic idea is to allow variables in the Herbrand Universe. We want to point out here that the elimination of the variables in the standard semantics was due to the need to eliminate existentially quantified variables, while the presence of variables in the Herbrand Universe allows universally quantified variables to be modeled. Essentially we want to have a syntactic way to capture the meaning of universally quantified formulas and then to be able to talk about validity not only for standard Herbrand models but for any other kind of models.

In the paper we will present the standard concepts in Section 2, and then the new Herbrand structures in Section 3. Section 4 is devoted to Herbrand interpretations and models. C-models and S-models are both based on Herbrand Universes with variables, but they are different in the ability to capture the operational behavior of the programs. In Sections 5 and 6 we will present the new model theoretic and fixpoint semantics. Finally Section 7 contains the main results of soundness and completeness for this semantic characterization, and in Section 8 some examples are discussed. Throughout the presentation we will give the relations between our semantics and the standard one [1, 6, 17].

2. The language: standard concepts

Let us recall the main definitions of Horn Clause Logic (HCL). Any concept not formally defined in the paper refers to [17].

The language alphabet is $\langle D, V, P\rangle$. D is a family indexed on N (non-negative integers), where D_n is a (possibly empty) set of n-adic operators $a, b, c, \ldots$ (data

constructors). If $d \in D_n$ then n is the *arity* of d. Constants are 0-adic constructors. V is a (infinite) denumerable set of variable symbols $x, y, z, \ldots$. P is a family indexed on N, where P_n is a (possibly empty) set of n-adic predicate symbols $p, q, r, \ldots$.

The *free D-algebra* on V, $T_{D(V)}$, is inductively defined as the least family such that:

- $\forall c \in D_0.\ c \in T_{D(V)}$ (D_0 contains always at least an element);
- $\forall v \in V.\ v \in T_{D(V)}$;
- $\forall t_1 \in T_{D(V)} \ldots \forall t_n \in T_{D(V)}.\ \forall d \in D_n.\ d(t_1, \ldots, t_n) \in T_{D(V)}$.

The elements of $T_{D(V)}$ are called *terms*. $T_{D(\emptyset)}$ is the *word algebra*, or the *Standard Herbrand Universe* (U). Its elements are called *ground terms*. The *Standard Herbrand Base* (B) is the set of all predicate symbols applied to ground terms.

The HCL basic construct is the *atomic formula* $p(t_1, \ldots, t_m)$, where $p \in P_m$, and the t_is are elements of $T_{D(V)}$. A *definite clause* is a construct of the form $A \leftarrow B_1, \ldots, B_n$ ($n \geq 0$), where A and the B_is are atomic formulas, "$\leftarrow$" and "," denote logic implication and conjunction, respectively, and all variables are universally quantified. A is the *head* of the clause and $B_1, \ldots, B_n$ is the *body*. If the body is empty the clause is a *unit clause* (denoted by A). A HCL *program* is a finite set of definite clauses $W = \{C_1, \ldots, C_n\}$. A *goal statement* is a construct of the form $\leftarrow A_1, \ldots, A_m$, where each A_i is an atomic formula.

Let E be an expression (term or formula). We denote by $Var(E)$ the set of variables occurring in E.

A *substitution* is a mapping $\vartheta: V \to T_{D(V)}$ such that $Dom(\vartheta)$ is finite, where $Dom(\vartheta)$ is the set $\{x \in V \mid \vartheta(x) \neq x\}$. Let $Im(\vartheta)$ denote the set $\{t \in T_{D(V)} \mid \exists x \in Dom(\vartheta), \vartheta(x) = t\}$. ϑ is a *valuation* if it is ground, i.e. $Im(\vartheta) \subseteq U$. ϑ is a *variable-pure* substitution if $Im(\vartheta) \subseteq V$ (i.e. the image of ϑ contains only variables). Given a set X of variables, $\vartheta|_X$ (ϑ restricted to X) is the substitution whose domain is $D = X \cap Dom(\vartheta)$ and such that for all variables in D it is equal to ϑ. For an expression E, $\vartheta|_{Var(E)}$ is abbreviated by $\vartheta|_E$. The composition of substitutions is defined in the obvious way, and induces a *preorder* on substitutions:

$$\vartheta_1 \leq \vartheta_2 \text{ iff } \exists \gamma.\ \vartheta_1 \gamma = \vartheta_2 \quad (\vartheta_1 \text{ is more general than } \vartheta_2).$$

In the following we will use the symbol $\circ$ for the composition of substitutions ($\vartheta \circ j = \vartheta j$) for readability. The *application* of a substitution ϑ to an expression (term, formula or n-tuple of expressions) E, denoted by $E\vartheta$, is defined as the simultaneous replacement of every variable x in E with $\vartheta(x)$. The application of substitutions induces a preorder on expressions:

$$E_1 \leq E_2 \text{ iff } \exists \vartheta.\ E_1 \vartheta = E_2.$$

In the following we will sometimes use $E_2 \geq E_1$ to denote $E_1 \leq E_2$.

The intersection of this preorder with its inverse is an equivalence relation (on expressions) called *variance* ($\approx$). In other words, two expressions E_1, E_2 are variants ($E_1 \approx E_2$) iff there exist two substitutions ϑ and γ such that $E_1\vartheta = E_2$ and $E_2\gamma = E_1$. In this case ϑ (resp. γ) is called *variable renaming* with respect to E_1 (resp. E_2). A different definition of variable renaming, equivalent to the above one, is the following: ϑ is a variable renaming for E_1 iff ϑ is a variable-pure substitution, ϑ is injective and $(Var(E_1) - Dom(\vartheta)) \cap Im(\vartheta) = \emptyset$.

Note that, in general, $E_1 \approx E_1'$ and $E_2 \approx E_2'$ do not imply $(E_1, E_2) \approx (E_1', E_2')$, while the converse is always true. As an example $f(x) \approx f(y)$ and $g(x) \approx g(z)$, but $(f(x), g(x))$ is not a variant of $(f(y), g(z))$.

Two expressions E_1 and E_2 are *unifiable* iff $\exists\vartheta.\ E_1\vartheta = E_2\vartheta$. If ϑ is a minimal substitution that makes E_1 and E_2 syntactically equal, then it is called the *mgu* of E_1 and E_2 ($mgu(E_1, E_2)$).

The operational semantics of HCL programs is based on the notion of refutation. Let G be the goal $\leftarrow A_1, \ldots, A_m$ and let $C \equiv A \leftarrow B_1, \ldots, B_n$ be a variant of a clause of W. Assume that A and A_i are unifiable, and let ϑ be their *mgu*. Then the goal

$$G' \equiv \leftarrow (A_1, \ldots, A_{i-1}, B_1, \ldots, B_n, A_{i+1}, \ldots, A_m)\vartheta$$

is *derivable* from G, by using C with substitution ϑ. Briefly, $G \mapsto_C^{\vartheta} G'$. By repeated applications of this step we obtain a *derivation*:

$$G \underset{C_1}{\overset{\vartheta_1}{\longmapsto}} G_1 \underset{C_2}{\overset{\vartheta_2}{\longmapsto}} G_2 \cdots \underset{C_n}{\overset{\vartheta_n}{\longmapsto}} G_n,$$

briefly $G \mapsto_{C_1 \cdots C_n}^{\vartheta^*} G_n$, where $\vartheta = \vartheta_1 \circ \cdots \circ \vartheta_n$. If G_n is empty (*null clause*, denoted by $\square$), then G is *refutable* in W and $\vartheta|_G$ is the *computed answer substitution* (c.a.s.) (note that in the literature the terminology SLD-resolution is used for this type of refutation together with a selection rule that chooses, for every goal, the atom to resolve).

The operational meaning of a program W is (more formally) defined as

$$SS = \{p(t_1, \ldots, t_n) \mid p(t_1, \ldots, t_n) \in B \text{ and } \exists\vartheta \text{ such that } \leftarrow p(t_1, \ldots, t_n) \overset{\vartheta}{\longmapsto}^* \square\}$$

The other standard semantics (model-theoretic and fixpoint), defined in [6], characterizes a HCL program W from a declarative point of view. Both of them are based on (standard) Herbrand interpretations (subsets of the Herbrand base). The model-theoretic semantics has to do with the notion of (standard) Herbrand model. A Herbrand model is a Herbrand interpretation which satisfies (à la Tarski) the program. The meaning of a program W is defined as the minimal Herbrand model M of W (i.e. the set of the ground atoms that are logical consequences of W). The second semantics is given as the least fixpoint (*lfp*) of a transformation T_W on Herbrand interpretations, defined as

$$T_W(I) = \{A \mid \exists A' \leftarrow B_1, \ldots, B_n \text{ in } W, \exists\vartheta \text{ such that } B_1\vartheta, \ldots, B_n\vartheta \in I \text{ and } A'\vartheta = A\}.$$

In [6] the equivalence of model-theoretic, fixpoint and operational semantics is proved ($M = lfp(T_W) = SS$). This result gives the soundness and completeness of SLD-resolution for HCL. However, it is worth noting that the operational semantics definition given above does not reflect entirely all the features of the language. In fact, the above set characterizes only the ground atoms which are refutable in (and which are logical consequences of) the program.

A more adequate operational semantics should also consider the refutability of non-ground goals together with the notion of computed answer substitution

$$SS' = \{(p(t_1, \ldots, t_n), \vartheta) \mid t_1, \ldots, t_n \in T_{D(V)}, \exists \vartheta \text{ such that } \leftarrow p(t_1, \ldots, t_n) \overset{\vartheta'}{\longmapsto}{}^* \square, \text{ and } \vartheta = \vartheta'|_{V(t_1,\ldots,t_n)}\}.$$

It is easy to see that $SS = \{A\vartheta\rho \mid (A, \vartheta) \in SS', \rho \text{ is ground}\}$, and SS' strictly contains more information than SS (SS' cannot be defined in terms of SS). Therefore M and $lfp(T_W)$ do not correspond any more to the new operational semantics. As a matter of fact, the full soundness and completeness results for SLD-resolution are given in terms of general models (i.e. models defined on any kind of domain) [17].

(*Soundness*) if $\leftarrow A_1, \ldots, A_m$ has a refutation in W with a computed answer substitution ϑ, then $(A_1, \ldots, A_m)\vartheta$ is a logical consequence of W (true in every model of W).

(*Completeness*) if $(A_1, \ldots, A_m)\vartheta$ is a logical consequence of W then there exists a substitution γ, more general than ϑ, such that $\leftarrow A_1, \ldots, A_m$ has a refutation in W with γ as computed answer substitution.

In Section 4 we give a new definition of Herbrand interpretations and models which will allow us to fully characterize the program's behavior from a declarative point of view.

Another kind of operational semantics, defined in [11], is based on the set of substitutions on clause heads obtained by unit resolution (a kind of bottom-up resolution, starting from unit clauses). In [19] this set is denotationally characterized by the least fixpoint of a transformation on substitutions. Our work can also be seen as a characterization of the relations between this kind of semantics and the standard operational semantics of logic programs (based on top-down resolution, starting from goals).

Appendix A contains some technical properties of the substitutions and unification which will be useful in the following.

3. Herbrand structures

The basic idea for the definition of our notion of Herbrand interpretations is to allow variables in the elements of the domain. The new domain syntactically characterizes a larger class of possible domains. In fact a term containing variables (e.g. $f(g(x, y))$) represents a set of elements whose *structure* is only partially determined. In the example, the elements represented by $f(g(x, y))$ are the ones obtainable

by the application of functions corresponding to g and f to *any* pair of elements. Hence x and y are *syntactical notations* standing for generic elements. The difference is that in the standard notion of Herbrand universe, the elements represented by x and y could be only those made by Herbrand constants and constructors, and in this way the full representativeness of general interpretations is lost.

The obvious solution could seem to consider directly $T_{D(V)}$ as the new Herbrand universe. This is not an elegant approach because in $T_{D(V)}$ there are different terms that represent the same set. For example $T_{D(V)}$ contains both $f(g(x, y))$ and $f(g(w, z))$. A more adequate definition is to consider $T_{D(V)}$ modulo variable renamings.

3.1. Definition. The new *Herbrand universe* U_V (for a given program) is defined as $T_{D(V)/\approx}$, i.e. the set of equivalence classes (*quotient set*) of $T_{D(V)}$ with respect to the *variance* equivalence relation $\approx$.

It will be useful to extend this definition to uniquely represent tuples of terms of U_V which differ only because of some variable renaming. We define, for $n \geq 1$, $U_V^n = T_{D(V)/\approx}^n$, where $T_{D(V)}^n$ is the nth cartesian product of $T_{D(V)}$.

It is well known that the preorder $\leq$ on $T_{D(V)}$ induces an order relation on $T_{D(V)/\approx}$ (and therefore on U_V). For the sake of simplicity, the elements of U_V will have the same representation as the elements of $T_{D(V)}$. For example, the intended meaning of $f(x, g(y)) \in U_V$, is that the equivalence class of $f(x, g(y))$ belongs to U_V. Analogously, the order on U_V will still be denoted by $\leq$. In the same way the preorder $\leq$ on $T_{D(V)}^n$ induces an ordering (still denoted by $\leq$) on U_V^n.

In the following, the elements of U_V will be called terms, and they will be denoted by choosing a representative whose variables are renamed, whenever it is needed, to avoid confusion with other variables. This also holds for any other structure that we will define (base, interpretations, etc.).

Operations such as the *mgu* are intended to be performed on arbitrary representatives (with the required renaming) of the equivalence classes. The new Herbrand base is defined as follows.

3.2. Definition. The new *Herbrand base* B_V (for a given program) is the set of all formulas $p(t_1, \ldots, t_n)$, where $p \in P$, p has arity n, and $(t_1, \ldots, t_n) \in U_V^n$.

The ordering on the U_V^ns induces an ordering on B_V, that is, if $(t_1, \ldots, t_n) \leq (t_1', \ldots, t_n')$ then $p(t_1, \ldots, t_n) \leq p(t_1', \ldots, t_n')$.

If I is a subset of B_V, let us denote with $ground(I)$ the set of ground formulas in I. Note that $B = ground(B_V)$.

4. Herbrand interpretations and models

Usually, Herbrand interpretations are subsets of the Herbrand base, and the notion of truth coincides with the one of being a member of. With the above proposed

base, and this notion of truth, the simple definition of interpretations as subsets is not adequate because of the presence of variables. In fact, as noted above, a variable stands for any possible element and it is not reasonable to have an interpretation containing (for example) an element $p(x)$ and not $p(a)$. Therefore, we must consider *upward-closed* subsets. Let us recall that a subset S of an ordered set is upward-closed if $s \in S$ and $s \preceq s'$ imply $s' \in S$.

This approach is not new; it was first proposed in [3] and later in [13, 10]. In these papers the idea is used to prove specific results. A complete formal treatment of this approach has been proposed and used in [4, 8, 9]. It is based on the notion of *termal model* and on the formal description of the operational semantics via proof trees. This allows some interesting results in the treatment of negation.

However, this is not the only possible approach. We could also keep the notion of Herbrand interpretations as subsets (not necessarily upward-closed) of B_V and consider a different notion of truth. This is not merely a difference in the use of formal tools, but corresponds to a different way of looking at the problem. Consider, for instance, the programs $W = \{p(x)., p(a)., q(a).\}$ and $W' = \{p(x)., q(a).\}$. In the first approach, W and W' have the same Herbrand interpretations (and models). From an operational point of view, W and W' are different. In fact, the goal $\leftarrow p(x)$ has a refutation with answer $\{x/a\}$ in W, while this is not the case in W'. In other words, in W, p is able to *produce* the data a, while, in W', p is only able to *consume* it (for a similar approach and a more detailed discussion of the producer-consumer relationship see [15]). We want a notion of interpretation which enables us not only to characterize the set of atomic logical consequences of a program, but also to capture the behavioral difference between programs like W and W'.

In the following we describe both of the approaches sketched above. We call them C-*approach* (upward-*Closed* interpretations) and S-*approach* (*Subset* interpretations), respectively. Consistently, we talk about C-*interpretations* and S-*interpretations*, C-*models* and S-*models*, etc. We show that the S-approach is richer and has a stronger relation with the operational semantics. Moreover, we show that the C-approach (and the corresponding results) can be derived from the S-approach, by relaxing the notion of interpretation structure (i.e. the pair Herbrand interpretation, notion of truth).

4.1. S-*interpretations and* S-*models*

4.1. Definition. An S-*Herbrand interpretation* I is any subset of B_V.

The notion of S-interpretation goes together with an appropriate notion of truth (S-*truth*) which defines the meaning of formulae.

4.2. Definition (S-*truth*). Let I be an S-interpretation. Then

- a unit clause $A\cdot$ is S-true in I iff A belongs to I,
- a definite clause $A \leftarrow B_1, \ldots, B_n$ is S-true in I iff for every $B'_1, \ldots, B'_n$ belonging to I, if there exists $\vartheta = mgu((B'_1, \ldots, B'_n), (B_1, \ldots, B_n))$, then $A\vartheta$ belongs to I,

- an atom A (possibly not ground) is S-true in I iff $\exists A'$, such that (the equivalence class of) A' belongs to I and $A' \leq A$.

Unit clauses are singled out for clarity. Their case is already contained in the definition for definite clauses. Note that the standard notion of truth can be consistently extended in our Herbrand interpretations by considering a ground atom true in I if it is an instance of an element of I, and by deriving, as usual, the notion of truth for more complex formulae (in terms of their components).

It is easy to show that the new notion of truth implies the standard one, but it is not equivalent. For instance, if I is an interpretation containing only $p(a)$ and $p(b)$, where a and b are the only ground elements of the Herbrand universe, then $\forall x.\, p(x)$ is true in I from the classical point of view, but it is not true with respect to our notion.

The following definition formally establishes the notion of S-model.

4.3. Definition. Let I be an S-Herbrand interpretation (of a program W). I is an S-*Herbrand model* of W iff every clause of W is S-true in I.

Note that atoms and unit clauses are treated differently in the notion of S-truth. This corresponds to the idea that the programs W and W' at the beginning of Section 4 have different S-models (even if they have the same models both in the classical approach and in the C-approach developed below).

4.2. C-interpretations and C-models

4.4. Definition. A C-*Herbrand interpretation* I is any upward-closed subset of B_V.

4.5. Definition (C-*truth*). Let I be a C-interpretation. Then

- a unit clause $A\cdot$ is C-true in I iff for every substitution ϑ, $A\vartheta$ is C-true in I,
- a definite clause $A \leftarrow B_1, \ldots, B_n$ is C-true in I iff for every substitution ϑ, if $B_1\vartheta, \ldots, B_n\vartheta$ are C-true in I, then $A\vartheta$ is C-true in I,
- an atom A (possibly not ground) is C-true in I iff (the equivalence class of) A belongs to I.

4.6. Definition. Let I be a C-Herbrand interpretation (of a program W). I is a C-*Herbrand model* of W iff every clause of W is C-true in I.

The C-approach differs from the standard one for reasons similar to those expressed for the S-approach. As a matter of fact, it can be considered a particular case of the S-approach. Some immediate relations between the two approaches are shown by the following results.

4.7. Proposition. *Let I be an S-interpretation. Let us denote by $up(I)$ the upward closure of I, which obviously is a C-interpretation. Let F be an atom or a definite clause.*

(a) *If F is S-true in I then it is C-true in $up(I)$.*

(b) *If F is C-true in $up(I)$ then it is S-true in $up(I)$.*

Proof. (a) If F is an atom and F is S-true in I then $\exists A \in I$, $\exists \vartheta$ such that $A\vartheta = F$ and therefore $F \in up(I)$. Now assume F is a definite clause $A \leftarrow B_1, \ldots, B_n$. If $B_1\vartheta, \ldots, B_n\vartheta \in up(I)$, then $\exists B'_1, \ldots, B'_n \in I$ such that $B'_1 \leqslant B_1\vartheta, \ldots, B'_n \leqslant B_n\vartheta$. It is possible to choose $B'_1, \ldots, B'_n$ in such a way that they do not share variables, either among them, or with $B_1\vartheta, \ldots, B_n\vartheta$. By Proposition A.2, $(B'_1, \ldots, B'_n) \leqslant (B_1, \ldots, B_n)\vartheta$. Then, by Proposition A.3, there exists $\vartheta' = mgu((B'_1, \ldots, B'_n), (B_1, \ldots, B_n))$, therefore $A\vartheta' \in I$ and then $A\vartheta \in up(I)$, since $A\vartheta' \leqslant A\vartheta$.

(b) If F is an atom the proof is obvious. If F is a definite clause $A \leftarrow B_1, \ldots, B_n$ and $\exists B'_1, \ldots, B'_n \in up(I)$ and $\vartheta = mgu((B'_1, \ldots, B'_n), (B_1, \ldots, B_n))$, then $B'_1\vartheta, \ldots, B'_n\vartheta \in up(I)$, and therefore $A\vartheta \in up(I)$. □

4.8. Corollary. *Let I be a subset of B_V. If I is an* S*-model then $up(I)$ is a* C*-model.*

Proof. It follows immediately from Proposition 4.7(a). □

4.9. Corollary. *If I is an upward closed subset of B_V, then*

(a) *a formula F (atom or definite clause) is* S*-true in I iff it is* C*-true in I.*

(b) *I is an* S*-model (of a program W) iff I is a* C*-model of W.*

Proof. It follows immediately from Proposition 4.7(b) and Corollary 4.8, since, whenever I is upward closed, $I = up(I)$. □

4.3. Relation with standard semantics

In the following we discuss the relation between our semantics and the standard semantics. Let us define some useful transformations.

4.10. Definition. Let W be a set of definite clauses.

(a) Given a (standard, not necessarily Herbrand) interpretation I of W, we define the corresponding C-interpretation (and S-interpretation) $H(I) = \{A \mid A \in B_V$ and $\forall A$ is true in $I\}$, where $\forall A$ is the universal closure of A.

(b) Given an S-interpretation (a C-interpretation) I for W, given an arbitrary pre-interpretation J (i.e. a domain D, and a mapping Ψ which maps every n-adic constructor c into a function $\Psi(c): D^n \to D$, see [17]), we define the standard interpretation $G_J(I)$ on J such that for every predicate p, for each $a_1, \ldots, a_n \in D$

(i) $p(a_1, \ldots, a_n)$ is verified in $G_J(I)$ iff $\exists t_1, \ldots, t_n \in U_V$ such that $p(t_1, \ldots, t_n)$ is S-true (C-true) in I and there exists an assignment ρ from the variables of $t_1, \ldots, t_n$ into elements of D such that $a_1 = \Psi_\rho(t_1), \ldots, a_n = \Psi_\rho(t_n)$, where $\Psi_\rho(t)$ represents the element of D obtained by applying Ψ and ρ to the components of t.

(ii) $p(a_1, \ldots, a_n)$ is false otherwise.

Note that

(a) the first definition is correct, i.e. $H(I)$ is upward-closed; in fact, if A is true in I, then, for each ϑ, $A\vartheta$ is true in I;

(b) if I contains ground atoms only, then the predicates can be verified in $G_J(I)$ only on the subset of D which corresponds to the Herbrand universe.

The following proposition shows that the above defined mapping H maps models in C- (and S-)models.

4.11. Proposition. *Let W be a set of definite clauses. If I is a model then $H(I)$ is a* C*-model (and therefore an* S*-model).*

Proof. Consider a clause $A \leftarrow B_1, \ldots, B_n$ true in I. Let ϑ be a substitution, and assume $B_1\vartheta, \ldots, B_n\vartheta$ belong to $H(I)$. Then $\forall(B_1, \ldots, B_n)\vartheta$ is true in I and thus $\forall A\vartheta$ is true in I. Therefore $A\vartheta$ belongs to $H(I)$. □

Proposition 4.11 allows the basic result in the standard Herbrand approach (that is, the existence of a Herbrand model for consistent sets of clauses) to be extended to our approaches.

4.12. Corollary. *Let W be a set of definite clauses. If W has a model then it has an* S*-model and a* C*-model.*

Proof. Immediate from Proposition 4.11. □

We can prove the counterpart of Proposition 4.11, i.e. G_J maps S-models (C-models) into models.

4.13. Proposition. *Let W be a set of definite clauses. If I is an* S*-model then for every pre-interpretation J, $G_J(I)$ is a model.*

Proof. Consider a clause $A \leftarrow B_1, \ldots, B_n$ true in I. Let ρ be an assignment, and assume $B_1, \ldots, B_n$ true in $G_J(I)$ under ρ. Then there exist $B'_1, \ldots, B'_n$ true in I, and there exists ϑ such that $B_1\vartheta = B'_1, \ldots, B_n\vartheta = B'_n$. $\forall i = 1, nB'_i(\psi\rho'(t^i_1), \ldots, \psi_{\rho'}(t^i_{k_i})) = B_i(\psi_\rho(t^i_1), \ldots, \psi_\rho(t^i_{k_i}))$, where $\rho'\vartheta = \rho$. Therefore $A\vartheta$ is true in I, then $A\vartheta$ is true in $G_J(I)$ under ρ' and thus A is true in $G_J(I)$ under ρ. □

4.14. Corollary. *Let W be a set of definite clauses. If W has an* S*-model (a* C*-model) then it has a model.*

Proof. Immediate from Proposition 4.13. □

4.15. Corollary. *If I is an* S*-model or a* C*-model for a set of definite clauses W then $ground(up(I))$ is a standard Herbrand model of W.*

Proof. Immediate from Proposition 4.13. In fact, if the pre-interpretation J has domain U, and the mapping Ψ is the standard mapping for constructors in the Herbrand interpretations, $G_J(I)$ is equal to $ground(up(I))$. □

5. Model-theoretic semantics

In this section we define an ordering relation on the classes of S-interpretations and C-interpretations, and we show that both of them are complete lattices. Furthermore, we show that for the two notions of Herbrand models the standard properties still hold. In particular, the Herbrand base is an S-model (a C-model) and the greatest lower bound of all S-models (C-models) is an S-model (a C-model) (see [14] for a similar proof). Therefore there exists the minimal S-model (C-model), which we define as the S-model-theoretic (C-model-theoretic) semantics of the program.

5.1. The lattice of S-*interpretations and the* S-*model-theoretic semantics*

5.1. Definition (*Ordering on* S-*interpretations*). Let I, I' be S-interpretations. $I \leqslant_S I'$ iff I is included (in the set-theoretic sense) in I'.

This definition of ordering reflects the idea that if an interpretation I is less than I' then I gives value true to less (or the same) atoms, by means of a proper subset of elements.

5.2. Proposition. *If* $I \leqslant_S I'$ *then, for every atom* A S-*true in* I, A *is* S-*true in* I'.

Proof. Immediate. □

5.3. Proposition. *The class of* S-*interpretations is a complete lattice with respect to* $\leqslant_S$, *i.e. every set of* S-*interpretations has a greatest lower bound and a least upper bound.*

Proof. Immediate. If L is a set of S-interpretations then $glb(L) = \bigcap L$ and $lub(L) = \bigcup L$. (Where $\bigcap$ and $\bigcup$ denote the set-theoretic intersection and union respectively.) □

Note that $\emptyset$ and (the Herbrand base) B_V are respectively the bottom and the top element of the lattice.

5.4. Proposition (Model intersection property). *If* L *is a non-empty set of* S-*models of a program* W, *then* $\bigcap L$ *is an* S-*model of* W.

Proof. Consider the definite clause $A \leftarrow B_1, \ldots, B_n$ of W. Assume $B'_1, \ldots, B'_n$ belong to $\bigcap L$, and there exists $\vartheta = mgu((B'_1, \ldots, B'_n), (B_1, \ldots, B_n))$. Then, for each I in L, $B'_1, \ldots, B'_n$ belong to I, and thus $A\vartheta$ belongs to I. Therefore $A\vartheta$ belongs to $\bigcap L$. □

5.5. Corollary. *The class of* S-*models is a complete lattice.*

Proof. Immediate by Proposition 5.4. In fact, it is sufficient to show that for any set L of models $lub(L)$ exists. Thus, let L' be the set of the upper bounds of L. If

L' is empty then $lub(L) = B_V$. Otherwise it is immediate to see that $lub(L) = inf L' = \bigcap L'$. □

5.6. Corollary (Existence of the minimal model). *For every program W there exists a minimal* S*-model* M_S.

Proof. The proof follows from Proposition 5.4, since B_V is an S-model of W. □

As usual, the minimal S-model M_S (whose existence is guaranteed by Corollary 5.6) can be defined to denote the S-model-theoretic semantics of a HCL program W. The following theorem shows that M_S represents the set of the atomic (possibly non-ground) logical consequences of W. Therefore M_S is more meaningful than the standard minimal Herbrand model, which represents the ground logical consequences only.

5.7. Theorem. *Let W be a program, and* M_S *be its minimal* S*-model. For every atom* $A \in B_V$, *A is* S*-true in* M_S (*A is an instance of an element of* M_S) *iff* $\forall A$ *is true in every model of W* ($\forall A$ *is a logical consequence of W*).

Proof. ($\Rightarrow$) (by contradiction): Consider an atom A S-true in M_S and assume there exists a model I in which $\forall A$ is not true. Therefore for each $A' \leq A$, $\forall A'$ is not true in I. Hence $H(I)$ (see Definition 4.10(a)) does not contain any atom $A' \leq A$. Since $H(I)$ is an S-model (by Proposition 4.11), M_S cannot contain any $A' \leq A$, which contradicts the hypothesis of A being S-true in M_S.

($\Leftarrow$) (by contradiction): Consider an atom A such that $\forall A$ is true in every model. Assume that A is not S-true in M_S. Let $x_1, \ldots, x_n$ be the variables occurring in A. Consider a pre-interpretation J on the domain of ground terms built on the constructors of W augmented with n new constants $a_1, \ldots, a_n$. $G_J(M_S)$ (see Definition 4.10(b) does not satisfy A under the assignment which instantiates each x_i with a_i, and then $\forall A$ is not true in $G_J(M_S)$. Since $G_J(M_S)$ is a model (by Proposition 4.13), this contradicts the hypothesis. □

5.2. The lattice of C*-interpretations and the* C*-model-theoretic semantics*

The definitions and propositions of the C-approach are developed along the lines of those given for the S-approach in the previous section.

5.8. Definition (*Ordering on* C*-interpretations*). Let I, I' be C-interpretations. $I \leq_C I'$ iff I is included in I'.

In this case the definition of ordering as set inclusion simply reflects the idea that an interpretation is less than another one iff it gives value true to less atoms.

5.9. Proposition. *$I \leqslant_C I'$ iff, for every atom A* C*-true in I, A is* C*-true in I'.*

Proof. Immediate. □

Note that this result is stronger (it is an iff relation) than the corresponding one (Proposition 5.2) about the relation between S-ordering and S-truth. This is due to the more constrained nature of C-interpretations (being upward-closed).

5.10. Proposition. *The class of* C*-interpretations is a complete lattice with respect to $\leqslant_C$, i.e. every set of* C*-interpretations has a greatest lower bound and a least upper bound.*

Proof. If L is a set of C-interpretations then $\bigcap L$ and $\bigcup L$ are upward-closed, therefore $glb(L) = \bigcap L$ and $lub(L) = \bigcup L$. □

Also in this case $\emptyset$ and (the Herbrand base) B_V are respectively the bottom and the top element of the lattice.

5.11. Proposition (Model intersection property). *If L is a non-empty set of* C*-models of a program W, then $\bigcap L$ is a* C*-model of W.*

Proof. From Corollary 4.9(b) and Proposition 5.4, being $\bigcap L$ upward-closed. □

5.12. Corollary. *The class of* C*-models is a complete lattice.*

Proof. Analogous to Corollary 5.5. □

5.13. Corollary (Existence of the minimal model). *For every program W there exists a minimal* C*-model M_C.*

Proof. The proof follows from Proposition 5.11, since B_V is a C-model of W. □

It can be shown that there is a strong relation between the minimal models in the two approaches.

5.14. Proposition. *For every program W, $M_C = up(M_S)$.*

Proof. ($\subseteq$): By Corollary 4.8 $up(M_S)$ is a C-model, then $M_C \subseteq up(M_S)$.

($\supseteq$): By Corollary 4.9(b) M_C is an S-model, then $M_S \subseteq M_C$. Therefore $up(M_S) \subseteq up(M_C) = M_C$. □

The minimal C-model M_C can be defined to denote the C-model-theoretic semantics of a HCL program W. The following theorem corresponds to Theorem 5.7.

5.15. Theorem. *Let W be a program, and M_C its minimal* C*-model. For every atom $A \in B_V$, A is* C*-true in M_C (A is an element of M_C) iff $\forall A$ is true in every model of W.*

Proof. Since an atom $A \in B_V$ is S-true in I iff it is S-true in $up(I)$, the proof follows immediately from Theorem 5.7 and Proposition 5.14. □

This result allows us to prove as a corollary the theorem [6, 17] that relates the minimal Herbrand model M of W and the set of the ground atoms that are logical consequences of W.

5.16. Corollary. *Let W be a program, and consider its minimal (standard) Herbrand-model M. Then $M = \{A \in B \mid A$ is a logical consequence of $W\} = ground(M_C)$.*

Proof. Let $GC = \{A \in B \mid A$ is a logical consequence of $W\}$ and $NGC = \{A \in B_V \mid \forall A$ is a logical consequence of $W\}$. Clearly $GC = ground(NGC)$. By Theorem 5.15 $ground(NGC) = ground(M_C)$. Finally, we have to prove that $M = ground(M_C)$.

($\subseteq$): Immediate by Corollary 4.15.

($\supseteq$): $M_C \subseteq H(M)$. Therefore $ground(M_C) \subseteq ground(H(M)) = M$ (see Definition 4.10). □

Theorem 5.7 and Theorem 5.15 show that our approach is richer than the standard one. In fact, it allows the set of correct answer substitutions to be characterized for a given goal (Theorem 6.6 of [17]) not only in the ground case.

5.17. Corollary. *Let W be a program and G a goal $\leftarrow A_1, \ldots, A_n$. Then the following are equivalent:*

(a) *ϑ is a correct answer substitution for $W \cup G$ (i.e. $\forall((A_1, \ldots, A_n)\vartheta)$ is a logical consequence of W);*

(b) *$(A_1, \ldots, A_n)\vartheta$ is true in every* S*-model (*C*-model);*

(c) *$(A_1, \ldots, A_n)\vartheta$ is true in the minimal* S*-model (*C*-model).*

Proof. The equivalence of (a) and (c) is an immediate extension of Theorem 5.7 (Theorem 5.15), and the equivalence of (b) and (c) is obvious. □

5.18. Example. Let us consider the following program W:

(1) $p(x) \leftarrow q(x)$.

(2) $p(a)$.

(3) $p(b)$.

(4) $q(x)$.

The standard (ground) model must contain at least all ground instances of the unit clauses, i.e. the atoms $p(a)$, $p(b)$, and $q(a)$, $q(b)$. The set $\{p(a), q(a), q(b), p(b)\}$ is a standard Herbrand model for W, and therefore it is the minimal one.

By our definition of S-model, any S-model must contain at least (variants of) the unit clauses, i.e. the atoms $p(a)$, $p(b)$, and $q(x)$. Moreover, by clause (1) $p(x)$ must also belong to any S-model.

We can note now that $\{p(a), p(b), q(x), p(x)\}$ is an S-model and therefore it is the minimal S-model.

The case of C-models is analogous. It can be easily shown that the minimal C-model is $\{p(a), q(a), q(b), p(b), p(x), q(x)\}$.

6. Fixpoint semantics

The denotational characterization of a program is usually given in terms of the least fixpoint of a continuous transformation associated to it. In the case of logic programs this transformation can be seen as an inference operator, and its least fixpoint is also used to prove the effectiveness of the minimal model and its relation with the operational semantics. In this section we define two transformations, one on S-interpretations and one on C-interpretations, for the S-approach and the C-approach, respectively. We prove that they are continuous, and we show the relation with the corresponding model-theoretic semantics defined in the previous section.

6.1. S-*transformation and* S-*fixpoint semantics*

6.1. Definition. Let W be a program. The mapping T_S on the set of S-Herbrand interpretations, associated to W, is defined as follows

$$T_S(I) = \{A' \in B_V \mid \exists A \leftarrow B_1, \ldots, B_n \text{ in } W, \exists B'_1, \ldots, B'_n \in I,$$
$$\exists \vartheta = mgu((B'_1, \ldots, B'_n), (B_1, \ldots, B_n)), \text{ and } A' = A\vartheta\}.$$

6.2. Proposition. *T_S is monotonic and continuous.*

Proof. Monotonicity is straightforward. Let us show that T_S is continuous. Let K be a chain (i.e. a totally ordered set) of S-interpretations. We have to prove that $T_S(\bigcup K) = \bigcup T_S(K)$.

($\supseteq$): Immediate by monotonicity.

($\subseteq$): Let $A' \in T_S(\bigcup K)$. Then $\exists A \leftarrow B_1, \ldots, B_n$ in W, $\exists B'_1, \ldots, B'_n \in \bigcup K$, $\exists \vartheta = mgu((B'_1, \ldots, B'_n), (B_1, \ldots, B_n))$, and $A' = A\vartheta$. Then there exist $I_1, \ldots, I_n \in K$ such that $B'_1 \in I_1, \ldots, B'_n \in I_n$. Let $I = \max\{I_1, \ldots, I_n\}$. Then $A' \in T_S(I) \subseteq \bigcup T_S(K)$. □

6.3. Corollary. *There exists the least fixpoint of T_S, $lfp(T_S)$ and $lfp(T_S) = \bigcup_{n \in \omega} T_S^n(\emptyset)$.*

Proof. Standard. □

6.4. Lemma. *An* S*-interpretation* I *is an* S*-model iff* $T_S(I) \subseteq I$ ($T_S(I) \leqslant_S I$).

Proof. I is an S-model
iff $(\forall A \leftarrow B_1, \ldots, B_n \in W$, if $\exists B'_1, \ldots, B'_n \in I$, such that
$\exists \vartheta = mgu((B'_1, \ldots, B'_n), (B_1, \ldots, B_n))$, then $A\vartheta \in I)$
iff $(\forall A'$ such that $\exists A \leftarrow B_1, \ldots, B_n \in W$, $\exists B'_1, \ldots, B'_n \in I$, and
$\exists \vartheta = mgu((B'_1, \ldots, B'_n), (B_1, \ldots, B_n))$, and $A' = A\vartheta$, then $A' \in I)$
iff $T_S(I) \subseteq I$. □

6.5. Theorem. *For every program* W, $M_S = lfp(T_S) = \bigcup_{n\in\omega} T_S^n(\emptyset) (= T_S \uparrow \omega$, *see* [17]).

Proof. It follows from Corollary 6.3 and Lemma 6.4, since, by monotonicity, $\min\{I \mid T_S(I) = I\} = \min\{I \mid T_S(I) \leqslant_S I\}$. □

6.2. C*-transformation and* C*-fixpoint semantics*

6.6. Definition. Let W be a program. The mapping T_C on the set of C-Herbrand interpretations, associated to W, is defined as follows

$$T_C(I) = \{A' \in B_V \mid \exists A \leftarrow B_1, \ldots, B_n \text{ in } W, \exists \vartheta \text{ such that } B_1\vartheta, \ldots, B_n\vartheta \in I, \text{ and } A' = A\vartheta\}.$$

It is easy to show that this definition is correct, since $T_C(I)$ is upward-closed.

6.7. Proposition. *For every* S*-interpretation* I, $T_C(up(I)) = up(T_S(I))$.

Proof. ($\supseteq$): If $A \in up(T_S(I))$ then $\exists A' \in T_S(I)$ such that $A' \leqslant A$. Therefore $\exists A'' \leftarrow B_1, \ldots, B_n$ in W, $\exists B'_1, \ldots, B'_n \in I$, $\exists \vartheta = mgu((B'_1, \ldots, B'_n), (B_1, \ldots, B_n))$, and $A' = A''\vartheta$. Hence $A''\vartheta \in T_C(up(I))$ and therefore $A \in up(T_C(up(I))) = T_C(up(I))$.

($\subseteq$): Let $A \in T_C(up(I))$. Then $\exists A' \leftarrow B_1, \ldots, B_n$ in W, $\exists \vartheta$ such that $B_1\vartheta, \ldots, B_n\vartheta \in up(I)$, and $A = A'\vartheta$. Therefore, $\exists B'_1, \ldots, B'_n \in I$, sharing no variables mutually and with $B_1, \ldots, B_n$, such that $B'_1 \leqslant B_1\vartheta, \ldots, B'_n \leqslant B_n\vartheta$. By Proposition A.2, $(B'_1, \ldots, B'_n) \leqslant (B_1, \ldots, B_n)\vartheta$. Therefore, by Proposition A.3, there exists $\vartheta'' = mgu((B_1, \ldots, B_n), (B'_1, \ldots, B'_n))$, and $\vartheta''|_{B_1,\ldots,B_n} \leqslant \vartheta$. Hence $A = A'\vartheta \geqslant A'\vartheta'' \in T_S(I)$. □

6.8. Proposition. T_C *is continuous* (*and, therefore, monotonic*).

Proof. This proposition can easily be proved as a consequence of the corresponding Proposition 6.2. Let K be a chain of C-interpretations. Then $T_C(\bigcup K) =$ (by Proposition 6.7) $up(T_S(\bigcup K)) =$ (by Proposition 6.2) $up(\bigcup T_S(K)) = \bigcup up(T_S(K)) =$ (by Proposition 6.7) $\bigcup T_C(K)$. □

6.9. Corollary. *There exists the least fixpoint of* T_C, $lfp(T_C)$ and $lfp(T_C) = \bigcup_{n\in\omega} T_C^n(\emptyset)$.

Proof. Standard. □

6.10. Lemma. *A C-interpretation I is a C-model iff $T_C(I) \subseteq I$ ($T_C(I) \leq_C I$).*

Proof. It can easily be proved by the corresponding Lemma 6.4. Let I be a C-interpretation. I is a C-model iff (by Corollary 4.9(b)) I is an S-model iff (by Lemma 6.4) $T_S(I) \subseteq I$ iff (since I is upward-closed) $up(T_S(I)) \subseteq I$ iff (by Proposition 6.7) $T_C(I) \subseteq I$. □

6.11. Theorem. *For every program W, $M_C = lfp(T_C) = \bigcup_{n\in\omega} T_C^n(\emptyset) (= T_C \uparrow \omega)$.*

Proof. It follows from Corollary 6.9 and Lemma 6.10, since by monotonicity, $\min\{I \mid T_C(I) = I\} = \min\{I \mid T_C(I) \leq_C I\}$. □

Let us now relate our transformations with T_W (the standard transformation on ground Herbrand interpretations).

6.12. Proposition. (a) *For every C-interpretation I, $ground(T_C(I)) = T_W(ground(I))$.*
(b) *For every S-interpretation I, $ground(up(T_S(I))) = T_W(ground(up(I)))$.*

Proof. Immediate by the definitions of T_C and T_W and Proposition 6.7. □

6.13. Corollary. *For every $n \in \omega$,*
(a) $ground(T_C^n(\emptyset)) = T_W^n(\emptyset)$.
(b) $ground(up(T_S^n(\emptyset))) = T_W^n(\emptyset)$.

Proof. (a) (by induction): ($n = 0$) $ground(T_C^0(\emptyset)) = \emptyset = T_W^0(\emptyset)$;
($n > 0$) $ground(T_C^n(\emptyset)) = ground(T_C(T_C^{n-1}(\emptyset)))$
$= T_W(ground(T_C^{n-1}(\emptyset)))$ (by Proposition 6.12)
$= T_W(T_W^{n-1}(\emptyset))$ (by inductive hypothesis)
$= T_W^n(\emptyset)$.
(b): We only need to show that $up(T_S^n(\emptyset)) = T_C^n(\emptyset)$. The proof follows from Proposition 6.7 with an inductive argument similar to case (a). □

Theorem 6.11 allows us to prove as a consequence the theorem [6, 17] about the equivalence between the minimal Herbrand model M of a program W and the least fixpoint of T_W.

6.14. Corollary. *For every program W, $M = lfp(T_W) = \bigcup_{n\in\omega} T_W^n(\emptyset) (= T_W \uparrow \omega)$.*

Proof.

$$M = ground(M_C) \quad \text{(by Corollary 5.16)}$$
$$= ground(\bigcup_{n\in\omega} T_C^n(\emptyset)) \quad \text{(by Theorem 6.11)}$$
$$= \bigcup_{n\in\omega} T_W^n(\emptyset) \quad \text{(by Corollary 6.13).} \quad \square$$

6.15. Example. Let us consider the same program W of Example 5.18, and the construction of its minimal S-model by our fixpoint transformation, starting from the empty set. The case of the C-model is analogous. Thus we consider the S-case only.

$T_S^0(\emptyset) = \emptyset$,

$T_S^1(\emptyset) = \{p(a), p(b), q(x)\}$, applying clauses (2), (3) and (4),

$T_S^2(\emptyset) = T_S(\{p(a), p(b), q(x)\}) = \{p(a), p(b), q(x), p(x)\}$

where $p(x)$ is obtained from $q(x)$ and clause (1),

$T_S^3(\emptyset) = T_S(\{p(a), p(b), q(x), p(x)\}) = \{p(a), p(b), q(x), p(x)\} = T_S^2(\emptyset)$,

thus $T_S^\omega(\emptyset) = \{p(a), p(b), q(x), p(x)\} = M_S$.

7. Relation between the declarative and the operational semantics

Let us now give two important results of our semantic construction, i.e. a soundness and a completeness theorem which fully characterize the correspondence between our model theoretic semantics and the operational semantics.

7.1. Theorem (Strong soundness). *Let W be a program, let G be a goal $\leftarrow A_1, \ldots, A_n$ and assume $G \mapsto^{\vartheta *} \square$. Then $\exists A'_1, \ldots, A'_n \in M_S$ and $\exists \vartheta' = mgu((A_1, \ldots, A_n), (A'_1, \ldots, A'_n))$ such that $\vartheta'|_G = \vartheta|_G$.*

Proof (*by induction on the length k of the refutation*). Assume (without loss of generality) that A_1 is the first atom selected for the derivation.

($k = 1$): In this case $n = 1$, $\exists A'_1 \cdot$ (unit clause) in W and $\exists \vartheta = mgu(A_1, A'_1)$. Then $A'_1 \in M_S$ and $\vartheta' = \vartheta$ satisfies the required properties.

($k > 1$): In this case

- $\exists C \equiv H \leftarrow B_1, \ldots, B_m$ which is a variant (sharing no variables with G) of a clause in W,
- $\exists \delta = mgu(A_1, H) = \delta|_{A_1} \cup \delta|_H$,
- $G' \mapsto^{*} \square$ (in $k-1$ steps), where $G' \equiv \leftarrow (B_1, \ldots, B_m, A_2, \ldots, A_n)\delta$, and
- $\vartheta = \delta \cdot \psi$ and $\vartheta|_G = (\delta|_{A_1} \circ \psi|_{G'})|_G$.

Then, by inductive hypothesis,

- $\exists B'_1, \ldots, B'_m, A''_2, \ldots, A''_n$ variants (sharing no variables mutually and with C, G and G') of atoms in M_S,
- $\exists \psi' = mgu((B_1, \ldots, B_m, A_2, \ldots, A_n)\delta, (B'_1, \ldots, B'_m, A''_2, \ldots, A''_n)) = \psi'|_{G'} \cup \psi'|_V$ where $V = Var(B'_1, \ldots, B'_m, A''_2, \ldots, A''_n)$.
- $\psi'|_{G'} = \psi|_{G'}$.

We have to prove that

(a) $\exists A'_1, \ldots, A'_n \in M_S$

(b) $\exists \vartheta' = mgu((A_1, \ldots, A_n), (A'_1, \ldots, A'_n))$

(c) $\vartheta'|_G = (\delta \circ \psi)|_G$.

We note that

- by Proposition A.3, $(\delta \circ \psi'|_{G'}) \cup \psi'|_V$ is a unifier of $(B_1, \ldots, B_m, A_2, \ldots, A_n)$ and $(B'_1, \ldots, B'_m, A''_2, \ldots, A''_n)$; then
- there exists $\gamma = mgu((B_1, \ldots, B_m), (B'_1, \ldots, B'_m)) = \gamma_1 \cup \gamma_2$, where $\gamma_1 = \gamma|_{B_1, \ldots, B_m}$ and $\gamma_2 = \gamma|_{B'_1, \ldots, B'_m}$,
- $\gamma \leqslant (\delta \circ \psi'|_{G'}) \cup \psi'|_V$, and therefore $\gamma_1 \leqslant \delta|_H \circ \psi'|_{G'}$,
- $H\gamma_1 \leqslant H\delta|_H \psi'|_{G'} = A_1\delta|_{A_1}\psi'|_{G'}$.

Let ξ be the least substitution such that $H\gamma_1\xi = A_1\delta|_{A_1}\psi'|_{G'}$. For the S-truth of C in M_S, $H\gamma_1 \in M_S$.

(a) Consider the following choice of the A'_is (belonging to M_S):

$$A'_1 = H\gamma, \qquad A'_2 = A''_2, \ldots, A'_n = A''_n.$$

(b) $\zeta = \xi \cup \psi'|_V \cup \delta|_{A_1} \circ \psi'|_{G'}$ is a unifier of $(A_1, \ldots, A_n)$ and $(A'_1, \ldots, A'_n)$, and therefore there exists $\vartheta' = mgu((A_1, \ldots, A_n), (A'_1, \ldots, A'_n))$. In fact

$$\begin{aligned}(A'_1, A'_2, \ldots, A'_n)\zeta &= (A'_1, A'_2, \ldots, A'_n)(\xi \cup \psi'|_V)\\ &= (H\gamma_1\xi, A''_2\psi'|_V, \ldots, A''_n\psi'|_V)\\ &= (A_1\delta|_{A_1}\psi'|_{G'}, A_2\delta|_{A_1}\psi'|_{G'}, \ldots, A_n\delta|_{A_1}\psi'|_{G'})\\ &= (A_1, A_2, \ldots, A_n)\delta|_{A_1}\psi'|_{G'}\\ &= (A_1, A_2, \ldots, A_n)\zeta.\end{aligned}$$

(c) $\vartheta' \leqslant \zeta$, and then $\vartheta'|_G \leqslant \delta|_{A_1} \circ \psi'|_{G'} = \delta|_{A_1} \circ \psi|_{G'}$. Therefore $\vartheta'|_G \leqslant (\delta \circ \psi)|_G$.

Now, we only need to show that $(\delta \circ \psi)|_G \leqslant \vartheta'|_G$. Let $Z = (Var(G) - Dom(\delta|_{A_1})) \cup Var(Im(\delta|_{A_1}))$. We note that

- by Proposition A.4, $\vartheta'|_G = \delta|_{A_1} \circ mgu((A_1\delta|_{A_1}, A_2\delta|_{A_1}, \ldots, A_n\delta|_{A_1}), (A'_1, A'_2, \ldots, A'_n))|_Z$ then, $\vartheta'|_G = \delta|_{A_1} \circ \vartheta''$, where

$$\vartheta'' = mgu((A_1\delta|_{A_1}, A_2\delta|_{A_1}, \ldots, A_n\delta|_{A_1}), (H\gamma_1, A''_2, \ldots, A''_n))|_Z.$$

Let $\varphi = mgu(A_1\delta|_{A_1}, H\gamma_1)|_Z$, then

- by Proposition A.5, $\vartheta'' = (\varphi \circ mgu((A_2\delta|_{A_1}\varphi, \ldots, A_n\delta|_{A_1}\varphi), (A''_2, \ldots, A''_n)))|_Z$,
- $\varphi \geqslant \eta$, where $\eta = mgu((B_1, \ldots, B_m)\delta|_H, (B'_1, \ldots, B'_m))|_{H\delta}$.

In fact,

$$\begin{aligned}
\varphi &= mgu(A_1\delta|_{A_1}, H\gamma_1)|_Z \\
&= mgu(A_1\delta|_{A_1}, H\gamma_1)|_{A_1\delta} \\
&= mgu(H\delta|_H, H\gamma_1)|_{H\delta} \\
&= mgu((x_1,\ldots,x_r)\delta|_H, (x_1,\ldots,x_r)\gamma_1)|_{H\delta}, \\
&\qquad \text{where } \{x_1,\ldots,x_r\} = (Dom(\gamma_1)\cap Var(H))\cup Dom(\delta|_H) \\
&= mgu((y_1,\ldots,y_s)\delta|_H, (y_1,\ldots,y_s)\gamma_1)|_{H\delta}, \\
&\qquad \text{where } \{y_1,\ldots,y_s\} = Dom(\gamma_1)\cup Dom(\delta|_H) \\
&\geq mgu((z_1,\ldots,z_t)\delta|_H, (z_1,\ldots,z_t)\gamma_1)|_{H\delta}, \\
&\qquad \text{where } \{z_1,\ldots,z_t\} = Dom(\gamma_1)\cup (Dom(\delta|_H)\cap Var(B_1,\ldots,B_m)) \\
&= mgu((B_1,\ldots,B_m)\delta|_H, (B_1,\ldots,B_m)\gamma_1)|_{H\delta} \\
&= mgu((B_1,\ldots,B_m)\delta|_H, (B'_1,\ldots,B'_m)\gamma_2)|_{H\delta} \\
&\geq mgu((B_1,\ldots,B_m)\delta|_H, (B'_1,\ldots,B'_m))|_{H\delta} \quad \text{(by Proposition A.6)} \\
&= \eta.
\end{aligned}$$

Then

- by Proposition A.7, $\vartheta'' \geq (\eta \circ mgu((A_2\delta|_{A_1}\eta,\ldots,A_n\delta|_{A_1}\eta),(A''_2,\ldots,A''_n)))|_Z$,
- by Proposition A.5, $(\eta \circ mgu((A_2\delta|_{A_1}\eta,\ldots,A_n\delta|_{A_1}\eta),(A''_2,\ldots,A''_n)))|_Z = mgu(((B_1,\ldots,B_m)\delta|_H,(A_2,\ldots,A_n)\delta|_{A_1}),(B'_1,\ldots,B'_m,A''_2,\ldots,A''_n))|_Z = \psi'|_Z$.

Moreover, we note that $Z\cap Dom(\psi) = Z\cap Dom(\psi')\subseteq Var(G')$, and then $\psi'|_Z = \psi|_Z$. Therefore

$$\vartheta'|_G = (\delta|_{A_1}\circ\vartheta'')|_G \geq (\delta|_{A_1}\circ\psi'|_Z)|_G = (\delta|_{A_1}\circ\psi|_Z)|_G = \vartheta|_G. \qquad \square$$

The same result of strong soundness holds also in the case of C-models, as shown in the following corollary.

7.2. Corollary (Strong soundness with respect to the C-semantics). *Let W be a program, let G be a goal $\leftarrow A_1,\ldots,A_n$ and assume $G \mapsto^{\vartheta *} \square$. Then $\exists A'_1,\ldots,A'_n \in M_C$ and $\exists \vartheta' = mgu((A_1,\ldots,A_n),(A'_1,\ldots,A'_n))$ such that $\vartheta'|_G = \vartheta|_G$.*

Proof. Immediate by Theorem 7.1, since M_C is contained in M_S (in fact, by Proposition 5.14, $M_C = up(M_S)$). $\square$

The standard soundness property [17] can be inferred from our strong correctness result, as shown by the next corollary.

7.3. Corollary (Standard soundness). *Let W be a program, let G be a goal $\leftarrow A_1, \ldots, A_n$ and assume $G \mapsto^{\vartheta *} \square$. Then $\forall G\vartheta$ is a logical consequence of W.*

Proof. By Theorem 7.1, $G\vartheta$ is S-true in M_S. Then, by Theorem 5.7, $\forall G\vartheta$ is true in every model of W. □

Now, to prove the completeness theorem, we borrow the following two technical lemmas from [17] (see Lemmas 8.1 and 8.2), and we prove an important technical lemma. Let us recall that an unrestricted reduction is a resolution step in which a unifier instead of a *mgu* is used. An unrestricted refutation is a sequence of unrestricted reductions whose final goal is the empty goal.

7.4. Lemma (Mgu lemma). *Let G be a goal. Assume that G has an unrestricted* SLD*-refutation $G \mapsto^{\vartheta'}_{C_1 \cdots C_n} * \square$, then there exists an* SLD*-refutation $G \mapsto^{\gamma}_{C_1 \cdots C_n} * \square$ of the same length with the same clauses and $\gamma \leqslant \vartheta'$.*

Proof. See [17]. □

7.5. Lemma (Lifting lemma). *Let G be a goal $\leftarrow A_1, \ldots, A_n$ and $G\vartheta \mapsto^{\vartheta'}_{C_1 \cdots C_m} * \square$, then there exists $G \mapsto^{\gamma}_{C_1 \cdots C_m} * \square$ and $\gamma \leqslant \vartheta \circ \vartheta'$.*

Proof. See [17]. □

In the following we use the symbol $\underline{E}$ to represent an n-tuple of expressions $E_1, \ldots, E_n$.

7.6. Lemma. *Let $A'_1, \ldots, A'_n \in M_S$ and G' be the goal $\leftarrow A'_1, \ldots, A'_n$, then there exists a choice of clauses $C_1, \ldots, C_m$ such that*

(a) *$\forall A''_1, \ldots, A''_n$ such that $(A''_1, \ldots, A''_n) \geqslant (A'_1, \ldots, A'_n)$ there exists a refutation for the goal $G'' = \leftarrow A''_1, \ldots, A''_n$, $G'' \mapsto^{\xi}_{C_1 \cdots C_m} * \square$, where $\xi|_{G''} = \varepsilon$ (as a particular case we obtain $G' \mapsto^{\vartheta}_{C_1 \cdots C_m} * \square$).*

(b) *If a goal $\leftarrow A_1, \ldots, A_n \mapsto^{\gamma}_{C_1 \cdots C_m} * \square$ then $(A_1, \ldots, A_n)\gamma \geqslant (A'_1, \ldots, A'_n)$.*

Proof. If $A'_1, \ldots, A'_n \in M_S$ then there exists k such that $A'_1, \ldots, A'_n \in T_S^k(\emptyset)$: We prove (a) and (b) together by induction on k.

($k = 1$) (*Part* a): There exist n unit clauses $C_1, \ldots, C_n$ with heads $H_1, \ldots, H_n$ that are variants of $A'_1, \ldots, A'_n$. Then $A''_1 \geqslant H_1, \ldots, A''_n \geqslant H_n$. Hence $\leftarrow A''_1, \ldots, A''_n$ has a refutation, with an empty computed answer substitution, which will use the clauses $C_1, \ldots, C_n$, independently from the order of application; for example $G'' \mapsto^{\xi}_{C_1 \cdots C_n} * \square$, where $\xi|_{G''} = \varepsilon$.

($k=1$)(*Part* b): Let us consider the same clauses of part (a). Given the refutation $\leftarrow A_1, \ldots, A_n \mapsto^{\gamma}_{C_1 \cdots C_n} * \square$, $A_1\gamma_1 = H_1\gamma_1, \ldots, A_n\gamma_n = H_n\gamma_n$, where $\gamma_i = mgu(A_i\gamma_1 \ldots \gamma_{i-1}, H_i)$, $1 \leq i \leq n$ and $\gamma_1 \circ \cdots \circ \gamma_n = \gamma$. Then $(A_1, \ldots, A_n)\gamma \geq (H_1, \ldots, H_n) = (A'_1, \ldots, A'_n)$. In fact,

$$A_1\gamma \geq A_1\gamma_1 = H_1\gamma_1 \geq H_1 = A'_1,$$

$$A_2\gamma \geq A_2\gamma_1\gamma_2 = H_2\gamma_2 \geq H_2 = A'_2,$$

$$\vdots$$

$$A_n\gamma = A_n\gamma_1 \ldots \gamma_n = H_n\gamma_n \geq H_n = A'_n.$$

($k>1$)(*Part* a): There exist n (variants of) clauses $C_1 \equiv H_1 \leftarrow \underline{B}_1, \ldots, C_n \equiv H_n \leftarrow \underline{B}_n$ such that $\exists \underline{B}'_1, \ldots, \underline{B}'_n \in T_S^{k-1}(\emptyset)$ and $\eta_1 = mgu(\underline{B}_1, \underline{B}'_1), \ldots, \eta_n = mgu(\underline{B}_n, \underline{B}'_n)$ and $H_1\eta_1 = A'_1, \ldots, H_n\eta_n = A'_n$. By inductive hypothesis,

$$\exists \underline{C}'_1, \ldots, \underline{C}'_n \text{ such that } \underline{B}''_1 \overset{\xi_1}{\underset{C'_1}{\longmapsto}} * \square, \ldots, \underline{B}''_n \overset{\xi_n}{\underset{C'_n}{\longmapsto}} * \square$$

and $\xi_1|_{\underline{B}''_1} = \varepsilon, \ldots, \xi_n|_{\underline{B}''_n} = \varepsilon$ for every $(\underline{B}''_1, \ldots, \underline{B}''_n) \geq (\underline{B}'_1, \ldots, \underline{B}'_n)$.

Let us consider any order of application of the clauses $C_1, \ldots, C_n$. Being $A''_i \geq A'_i$ and $H_i\eta_i = A'_i$, there exists $\eta'_i = mgu(H_i, A''_i)$ with $H_i\eta'_i = A''_i$, $i = 1, \ldots, n$, and, consequently,

$$G'' \equiv \leftarrow A''_1, \ldots, A''_n \overset{\eta'}{\underset{C_1 \cdots C_n}{\longmapsto}} * \leftarrow (\underline{B}_1, \ldots, \underline{B}_n)\eta'$$

where $\eta' = \eta'_1 \circ \cdots \circ \eta'_n$ and $\eta'|_{G''} = \varepsilon$. (Note that $(\underline{B}_1, \ldots, \underline{B}_n)\eta' = (\underline{B}_1\eta'_1, \ldots, \underline{B}_n\eta'_n)$ because $\eta'|_{G''} = \varepsilon$ and the $\underline{B}_i$s do not share variables.) Moreover, by similar arguments, there exist η''_i and γ_i such that $\eta'_i \circ \eta''_i = \eta_i \circ \gamma_i$ and $H_i\eta_i\gamma_i = A''_i$, $i = 1, \ldots, n$, and each η''_i does not instantiate the variables in $A''_1, \ldots, A''_n$. Thus there exists the unrestricted reduction

$$G'' \equiv \leftarrow A''_1, \ldots, A''_n \overset{\eta}{\underset{C_1 \cdots C_n}{\longmapsto}} * \leftarrow (\underline{B}_1\eta'_1\eta''_1, \ldots, \underline{B}_n\eta'_n\eta''_n)$$

where $\eta = \eta'_1 \circ \eta''_1 \circ \cdots \circ \eta'_n \circ \eta''_n$ and

$$\begin{aligned}(\underline{B}_1, \ldots, \underline{B}_n)\eta &= (\underline{B}_1\eta'_1\eta''_1 \ldots \eta'_n\eta''_n, \ldots, \underline{B}_n\eta'_1\eta''_1 \ldots \eta'_n\eta''_n) \\ &= (\underline{B}_1\eta'_1\eta''_1, \ldots, \underline{B}_n\eta'_n\eta''_n) = (\underline{B}_1\eta_1\gamma_1, \ldots, \underline{B}_n\eta_n\gamma_n) \\ &= (\underline{B}'_1\eta_1\gamma_1, \ldots, \underline{B}'_n\eta_n\gamma_n) \geq (\underline{B}'_1, \ldots, \underline{B}'_n).\end{aligned}$$

By inductive hypothesis

$$G''' \equiv \leftarrow (\underline{B}_1, \ldots, \underline{B}_n)\eta \overset{\xi}{\underset{C'_1 \cdots C'_n}{\longmapsto}} * \square \quad \text{where } \xi|_{G'''} = \varepsilon.$$

Therefore there exists an unrestricted refutation

$$G'' \overset{\eta}{\underset{C_1 \cdots C_n}{\longmapsto}} * G''' \overset{\xi}{\underset{C'_1 \cdots C'_n}{\longmapsto}} * \square \quad \text{with } (\eta \circ \xi)|_{G''} = \varepsilon.$$

Now, by Lemma 7.4, Part (a) is proved.

($k>1$) (*Part* b): Let us consider the same (variants of) clauses of part (a). $C_1 \equiv H_1 \leftarrow \underline{B}_1, \ldots, C_n \equiv H_n \leftarrow \underline{B}_n$ such that $\exists \underline{B}'_1, \ldots, \underline{B}'_n \in T_S^{k-1}(\emptyset)$ and $\eta_1 = mgu(\underline{B}_1, \underline{B}'_1), \ldots, \eta_n = mgu(\underline{B}_n, \underline{B}'_n)$ and $H_1\eta_1 = A'_1, \ldots, H_n\eta_n = A'_n$. Let $\vartheta = \eta_1 \circ \cdots \circ \eta_n$. ϑ is the $mgu((\underline{B}_1, \ldots, \underline{B}_n), (\underline{B}'_1, \ldots, \underline{B}'_n))$ because, for an appropriate choice of the variants, the atoms in different clauses do not share variables. Moreover

$$(A'_1, \ldots, A'_n) = (H_1\eta_1, \ldots, H_n\eta_n) = (H_1, \ldots, H_n)\vartheta. \tag{i}$$

We know (Part (a)) that $\leftarrow \underline{B}_1, \ldots, \underline{B}_n \longmapsto^{\xi}_{C'_1 \cdots C'_n} * \square$, where $\xi|_{(\underline{B}_1, \ldots, \underline{B}_n)} = \varepsilon$.

Thus, if we assume that our derivation is the following:

$$\leftarrow A_1, \ldots, A_n \overset{\gamma_1}{\underset{C_1 \cdots C_n}{\longmapsto}} * \leftarrow (\underline{B}_1, \ldots, \underline{B}_n)\gamma_1 \overset{\gamma_2}{\underset{C'_1 \cdots C'_n}{\longmapsto}} * \square \quad (\gamma = \gamma_1 \circ \gamma_2),$$

by inductive hypothesis, $(\underline{B}_1, \ldots, \underline{B}_n)\gamma_1\gamma_2 \geqslant (\underline{B}'_1, \ldots, \underline{B}'_n)$. By the definition of *mgu*,

$$\gamma_1 \circ \gamma_2 \geqslant \vartheta|_{(\underline{B}_1, \ldots, \underline{B}_n)} = \vartheta|_{(H_1, \ldots, H_n, \underline{B}_1, \ldots, \underline{B}_n)}.$$

Thus

$$(H_1, \ldots, H_n)\vartheta \leqslant (H_1, \ldots, H_n)\gamma_1\gamma_2. \tag{ii}$$

We can prove, in analogy to the case $k=1$, that $(H_1, \ldots, H_n)\gamma_1 = (A_1, \ldots, A_n)\gamma_1$, and therefore, together with relations (i) and (ii),

$$A'_1, \ldots, A'_n = (H_1, \ldots, H_n)\vartheta \leqslant (H_1, \ldots, H_n)\gamma_1\gamma_2 = (A_1, \ldots, A_n)\gamma_1\gamma_2$$
$$= (A_1, \ldots, A_n)\gamma. \qquad \square$$

Let us now give the strong completeness theorem.

7.7. Theorem (Strong completeness). *Let W be a program, let G be a goal $\leftarrow A_1, \ldots, A_n$. If $\exists A'_1, \ldots, A'_n \in M_S$ and $\exists \vartheta' = mgu((A_1, \ldots, A_n), (A'_1, \ldots, A'_n))$, then $\exists \vartheta$ such that $G \longmapsto^{\vartheta} * \square$, and $\vartheta'|_G = \vartheta|_G$.*

Proof. By Lemma 7.6(a), there exist m clauses $C_1, \ldots, C_m$ such that

$$G' \equiv \leftarrow A'_1, \ldots, A'_n \overset{\xi}{\underset{C_1 \cdots C_m}{\longmapsto}} * \square \quad (\text{with } \xi|_{G'} = \varepsilon).$$

Since $(A'_1, \ldots, A'_n) \leqslant (A'_1, \ldots, A'_n)\vartheta' = (A_1, \ldots, A_n)\vartheta'$, by Lemma 7.6(a),

$$G'' \equiv \leftarrow (A_1, \ldots, A_n)\vartheta' \overset{\xi}{\underset{C_1 \cdots C_m}{\longmapsto}} * \square \quad (\text{with } \xi|_{G''} = \varepsilon).$$

By the lifting lemma, $\leftarrow A_1, \ldots, A_n \longmapsto^{\vartheta}_{C_1 \cdots C_m} * \square$ and

$$\vartheta|_G \leqslant (\vartheta' \circ \xi)|_G = (\vartheta' \circ \xi|_{G''})|_G = (\vartheta' \circ \varepsilon)|_G = \vartheta'|_G.$$

By Lemma 7.6(b), $(A'_1, \ldots, A'_n) \leq (A_1, \ldots, A_n)\vartheta$. Since ϑ' is an *mgu*, $\vartheta'|_G \leq \vartheta|_G$. Therefore $\vartheta|_G = \vartheta'|_G$. □

In the case of the C-semantics the completeness result we obtain is less strong, due to the lower degree of structure of the C-models with respect to the S-models.

7.8. Corollary. *Let W be a program, let G be a goal* $\leftarrow A_1, \ldots, A_n$. *If* $\exists A'_1, \ldots, A'_n \in M_C$ *and* $\exists \vartheta' = mgu((A_1, \ldots, A_n), (A'_1, \ldots, A'_n))$, *then* $\exists \vartheta$ *such that* $G \longmapsto^{\vartheta *} \square$, *and* $\vartheta'|_G \geq \vartheta|_G$.

Proof. Assume the hypotheses of the corollary are verified. By Proposition 5.14, $M_C = up(M_S)$. Therefore, $\exists A''_1, \ldots, A''_n \in M_S$ such that $(A''_1, \ldots, A''_n) \leq (A'_1, \ldots, A'_n)$. Let δ be the substitution such that $(A''_1, \ldots, A''_n)\delta = (A'_1, \ldots, A'_n)$. Then

$$(A''_1, \ldots, A''_n)\delta\vartheta' = (A'_1, \ldots, A'_n)\vartheta' = (A_1, \ldots, A_n)\vartheta'.$$

By Proposition A.3, $(\delta \circ \vartheta')|_{A''_1, \ldots, A''_n} \cup \vartheta'|_{A_1, \ldots, A_n}$ is a unifier of $A''_1, \ldots, A''_n$ and $A_1, \ldots, A_n$. Let $\vartheta'' = mgu((A''_1, \ldots, A''_n), (A_1, \ldots, A_n))$. Then $\vartheta''|_G = \vartheta''|_{A_1, \ldots, A_n} \leq \vartheta'|_{A_1, \ldots, A_n} = \vartheta'|_G$. By Theorem 7.7, $G \mapsto^{\vartheta *} \square$, and $\vartheta''|_G = \vartheta|_G$. Therefore, $\vartheta|_G = \vartheta''|_G \leq \vartheta'|_G$. □

The standard completeness theorem is just a special case of Corollary 7.8.

7.9. Corollary. *Let W be a program, let G be a goal* $\leftarrow A_1, \ldots, A_n$, *and* ϑ' *be a substitution. If* $\forall (A_1, \ldots, A_n)\vartheta'$ *is a logical consequence of W then* $\exists \vartheta$ *such that* $G \mapsto^{\vartheta *} \square$, *and* $\vartheta'|_G \geq \vartheta|_G$.

Proof. By Theorem 5.15, if $\forall (A_1, \ldots, A_n)\vartheta'$ is true in every model of W, then $(A_1, \ldots, A_n)\vartheta' \in M_C$. Of course, $\vartheta' = mgu((A_1, \ldots, A_n), (A_1, \ldots, A_n)\vartheta')$, then apply Corollary 7.8. □

Note that our completeness results are actually stronger than the standard ones, since we are able to denotationally characterize exactly the set of c.a.s. In fact, in Theorem 7.7, the substitutions we can infer from the minimal model are obtainable as c.a.s., while this is not the case in the standard result (only an approximation can be computed). Note that $(A, \vartheta) \in SS'$ iff $\exists A' \in M_S$ and $\exists \vartheta' = mgu(A, A')$, with $\vartheta'|_A = \vartheta$.

Theorems 7.1 and 7.7 also state that if a goal G has an empty computed answer substitution (restricted to the variables of G), then a less or equal set of atoms can be found in M_S, and vice versa.

8. Examples

Let us now consider some examples to clarify our construction.

8.1. Example. Let us consider the following definition:

(1) $p(x, y)$.

(2) $r(0)$.

(3) $r(s(x))$.

In this simple case, the minimal fixpoints of standard semantics, C-semantics and S-semantics are the following,

$$M_S = \{p(x, y), r(0), r(s(x))\},$$

$$M_C = \{p(x, y) \mid x, y \in U_V\} \cup \{r(s(x)) \mid x \in U_V\} \cup \{r(0)\},$$

$$M_W = \{p(x, y) \mid x, y \in U\} \cup \{r(x) \mid x \in U\},$$

where the standard Herbrand universe is $U = \{t \mid t = s^n(0), n \geq 0\}$.

This example shows that the model M_S of a program without recursion is finite, while both M_C and M_W can be infinite.

8.2. Example. Let us consider the following definition:

(1) $\mathit{Length}([\ ], 0)$.

(2) $\mathit{Length}([x \mid y], s(z)) \leftarrow \mathit{Length}(y, z)$.

(3) $\mathit{List2}(l) \leftarrow \mathit{Length}(l, x), x \leq s(s(0))$.

(4) $0 \leq x$.

(5) $s(y) \leq s(x) \leftarrow y \leq x$.

This is the usual program to implement the operation *Length* of a list, and the relation $\leq$. *List2* is a predicate on lists which is true if the list has a length less than or equal to two. As usual, $[\ ]$ represents the empty list, and $[x_1, \ldots, x_n \mid y]$ represents the list which has $x_1, \ldots, x_n$ as first elements and y as the rest of the list.

Let us compute the minimal fixpoint in the case of the S-semantics. The minimal S-model is the following:

$$T_S^1(\emptyset) = \{\mathit{Length}([\ ], 0), 0 \leq x\}, \quad \text{by clauses (1) and (4)}$$

$$T_S^2(\emptyset) = T_S^1(\emptyset) \cup \{\mathit{Length}([x], s(0)), s(0) \leq s(x), \mathit{List2}([\ ])\},$$

by clauses (2), (3), (5)

$$\vdots$$

Thus

$$M_S = \{\mathit{Length}([x_1, \ldots, x_n], s^n(0)) \mid n \in N\} \cup \{s^n(0) \leq s^n(x) \mid n \in N\}$$

$$\cup \{\mathit{List2}([\ ]), \mathit{List2}([x]), \mathit{List2}([x_1, x_2])\} \quad (\text{with } s^0(0) \equiv 0).$$

Note that the model of the predicate *List2* is finite even if it depends on a recursive one.

If the definitions of the predicate *Length* were inserted into another program with different constants and constructors, the part of the model for the predicate *Length* would not change. Thus we have a more compositional and clear semantics. In fact, the semantics for the predicate *Length* is fully independent from that of other possible predicates and from the functions and constants which appear in them.

Let us give a final example where the relations between the various models are discussed.

8.3. Example. The following program defines a predicate whose third argument is the sum of the others.

(1) $Plus(x, 0, x)$.

(2) $Plus(x, s(y), s(z)) \leftarrow Plus(x, y, z)$.

Let us compare the minimal fixpoint in the case of standard semantics, C-semantics and S-semantics. The minimal S-model is the following:

$$T_S^1(\emptyset) = \{Plus(x, 0, x)\}, \quad \text{by clause (1)}$$

$$T_S^2(\emptyset) = T_S(\{Plus(x, 0, x)\}) = \{Plus(x, 0, x), Plus(x, s(0), s(x))\}$$

$$= T_S^1(\emptyset) \cup \{Plus(x, s(0), s(x))\},$$

where $Plus(x, s(0), s(x))$ is added by applying $Plus(x, 0, x)$ to clause (2)

$$T_S^3(\emptyset) = T_S(\{Plus(x, 0, x), Plus(x, s(0), s(x))\})$$

$$= \{Plus(x, 0, x), Plus(x, s(0), s(x)), Plus(x, s^2(0), s^2(x))\}$$

$$= T_S^2(\emptyset) \cup \{Plus(x, s^2(0), s^2(x))\}$$

$$\vdots$$

Thus

$$M_S = \{Plus(x, s^n(0), s^n(x)) \mid n \in N\} \quad (\text{with } s^0(0) \equiv 0).$$

The standard Herbrand universe is $U = \{t \mid t = s^n(0), n \geq 0\}$ and $M_C = M_S \cup \{Plus(t, s^n(0), s^n(t)) \mid n \in N \text{ and } t \in U_V\}$, while the standard minimal model M_W is $\{Plus(t, s^n(0), s^n(t)) \mid n \in N \text{ and } t \in U\}$. Hence $M_W \subseteq M_C$.

Note that all universally quantified conjunctions of formulae in M_S are theorems of the theory given by the program. For example, all the formulas $\forall x.\ Plus(x, s^n(0), s^n(x))$, which, roughly speaking, assert that the third argument is $x + n$ when the first argument is x and the second argument is n, are theorems.

Let us now consider some queries: for example $\leftarrow Plus(x, s^2(0), y)$. There is only one possibility for the reduction of this query, that is to apply clause (2) twice and then clause (1). Thus we obtain the computed answer substitution, restricted to the variables of our query, $\vartheta|_{\{x,y\}} = \{y/s^2(x)\}$. Let us now consider which kind of information the three minimal models can give us about the possible computed answer substitutions. With M_W we obtain no useful information. In fact, atoms

such as $Plus(t, s^n(0), s^n(t))$ with $t \in U$ are not computed as answers. M_C contains information about variables, but it is too large to characterize the answers. It is useful to give the most general answers plus all their possible instances.

In the case of M_S, instead, according to Theorems 7.1 and 7.7, the only possible unification between atoms in M_S and the initial query is

$$\vartheta' = mgu(Plus(x, s^2(0), y), Plus(x, s^2(0), s^2(x))).$$

Thus we obtain exactly $\vartheta'|_{\{x,y\}} = \{y/s^2(x)\} = \vartheta|_{\{x,y\}}$, as expected.

Let us now consider the query $\leftarrow Plus(x, y, s(0))$. There are two possible success paths which give the computed answer substitutions $\vartheta_1|_{\{x,y\}} = \{x/0, y/s(0)\}$ and $\vartheta_2|_{\{x,y\}} = \{x/s(0), y/0\}$. Through unification with atoms in M_S, we obtain

$$\vartheta_1' = mgu(Plus(x, y, s(0)), Plus(x, s(0), s(x)))$$

and

$$\vartheta_2' = mgu(Plus(x, y, s(0)), Plus(x, 0, x)).$$

Therefore

$$\vartheta_1'|_{\{x,y\}} = \{x/0, y/s(0)\} = \vartheta_1|_{\{x,y\}}$$

and

$$\vartheta_2'|_{\{x,y\}} = \{x/s(0), y/0\} = \vartheta_2|_{\{x,y\}}.$$

Note that $\forall x.\ Plus(x, s^2(0), s^2(x))$, $Plus(0, s(0), s(0))$, and $Plus(s(0), 0, s(0))$, i.e. the universally closed quantifications of the atoms resulting from the computations, are instances of corresponding more general theorems given by M_S.

9. Conclusions

In the case of pure logic programs, our semantics differs from the standard vanEmden-Kowalski semantics [6] essentially for the presence of variables in interpretations (and models). This allows

- the truth of universally quantified atoms to be modeled: $\forall x.\ p(\ldots x \ldots)$ is valid iff $p(\ldots y \ldots)$ belongs to M (for counterexamples in the standard case, see [17]).
- a completeness theorem (in the non-ground case) more elegant than the standard one.

Our declarative semantics therefore capture the difference between answers which are effectively computed and answers obtained by instantiation of universally quantified variables. The ability to model such a difference fills the gap between the operational and the declarative semantics. Moreover, it provides a declarative characterization of relevant operational properties. We only mention two aspects, i.e. logic data bases and partially determined data structures. One important property of logic data bases is that the query evaluation process always computes ground answers. This property has a straightforward counterpart in the declarative semantics (no atom in M_S contains variable symbols). Logic programs which compute partially

determined data structures, on the other hand, have in M_S atoms containing variables (possibly within a data structure).

We are currently looking into some promising applications of our approach, namely:

- the characterization of the non-ground finite failure set by $T\downarrow\omega$.
- the semantics of general logic programs which contain negation and, in particular, universally quantified atoms, which are valid iff they belong to our minimal model.
- the generalization of our construction to a logic programming scheme, in the style suggested in [12]. In our case, the initial algebra is a set of (possibly) non-ground terms.
- the possibility of using our notion of models as the basis for abstract interpretations and program analysis and transformation techniques [2, 5, 9, 18].
- the possibility of using and extending our notion of models to cope with the semantics of concurrent logic languages such as CP and GHC [14, 15, 16] and of committed-choice logic languages [7].

Appendix A. Technical properties of substitutions

A.1. Definition (*Union of substitutions*). If ϑ_1, ϑ_2 are substitutions such that $Dom(\vartheta_1) \cap Dom(\vartheta_2) = \emptyset$, then $\vartheta_1 \cup \vartheta_2$ is the substitution whose domain is $Dom(\vartheta_1) \cup Dom(\vartheta_2)$ and such that

$$\vartheta_1 \cup \vartheta_2(x) = \begin{cases} \vartheta_1(x) & \text{if } x \in Dom(\vartheta_1) \\ \vartheta_2(x) & \text{if } x \in Dom(\vartheta_2). \end{cases}$$

Note that the operator $\cup$ (on substitutions) is associative and commutative.

A.2. Proposition. *Let $E_1, \ldots, E_m$, $E'_1, \ldots, E'_m$ be expressions such that $E_1 \leqslant E'_1, \ldots, E_m \leqslant E'_m$ and $E_1, \ldots, E_m$ do not share variables. Then $(E_1, \ldots, E_m) \leqslant (E'_1, \ldots, E'_m)$ holds.*

Proof. Let $\vartheta_1, \ldots, \vartheta_m$ be the substitutions such that $E_1\vartheta_1 = E'_1, \ldots, E_m\vartheta_m = E'_m$. Define the substitution $\vartheta = \vartheta_1|_{E_1} \cup \cdots \cup \vartheta_m|_{E_m}$. Since $E_1, \ldots, E_m$ do not share variables, $(E_1, \ldots, E_m)\vartheta = (E_1\vartheta_1, \ldots, E_m\vartheta_m)$ holds, then $(E_1, \ldots, E_m)\vartheta = (E'_1, \ldots, E'_m)$. □

A.3. Proposition. *Let E_1 and E_2 be two expressions such that $E_1\vartheta_1 = E_2\vartheta_2$ and E_1 and E_2 do not share variables. Then $\vartheta = \vartheta_1|_{E_1} \cup \vartheta_2|_{E_2}$ is a unifier for E_1 and E_2.*

Proof. Since E_1 and E_2 do not share variables, $E_1\vartheta = E_1\vartheta_1$ and $E_2\vartheta = E_2\vartheta_2$ hold, therefore $E_1\vartheta = E_2\vartheta$. □

A.4. Proposition. *Let $E_1, \ldots, E_m$ and $E'_1, \ldots, E'_m$ be expressions, and let γ be a substitution such that*

- *$E_1, \ldots, E_m$ and $E'_1, \ldots, E'_m$ do not share variables,*

- *there exists* $\vartheta = mgu(E_1, E'_1)$,
- *Dom*(γ) *is contained in* $Var(E'_1)$, E'_1 *does not share variables with* $E'_2, \ldots, E'_m$ *and* $(E_1, \ldots, E_m)\vartheta$ *and* $(E'_1\gamma, E'_2, \ldots, E'_m)$ *do not share variables,*
- *there exists* $\sigma = mgu((E_1, \ldots, E_m), (E'_1\gamma, E'_2, \ldots, E'_m))$.

Then, there exists $mgu((E_1, \ldots, E_m)\vartheta, (E'_1\gamma, E'_2, \ldots, E'_m))$ *and*

$$\sigma|_{E_1,\ldots,E_m} = (\vartheta \circ mgu((E_1, \ldots, E_m)\vartheta, (E'_1\gamma, E'_2, \ldots, E'_m)))|_{E_1,\ldots,E_m}.$$

Proof. We prove the proposition for the case $m = 1$. The extension to the case $m > 1$ is left to the reader. Since $E_1\sigma = E'_1\gamma\sigma$, by Proposition A.3 we have that $\sigma|_{E_1} \cup \gamma \circ \sigma|_{E'_1\gamma}$ is a unifier for E_1 and E'_1. Therefore, $\vartheta|_{E_1} \leq \sigma|_{E_1}$. Let δ be the minimal substitution such that $\vartheta|_{E_1} \circ \delta = \sigma|_{E_1}$. We have $E_1\vartheta\delta = E_1\sigma = E'_1\gamma\sigma$. Then, by Proposition A.3, $\delta|_{E_1\vartheta} \cup \sigma|_{E'_1\gamma}$ is a unifier of $E_1\vartheta$ and $E'_1\gamma$. Let $\alpha = mgu(E_1\vartheta, E'_1\gamma)$. Then, $\alpha|_{E_1\vartheta} \leq \delta|_{E_1\vartheta}$. On the other side, $(\vartheta|_{E_1} \circ \alpha|_{E_1\vartheta}) \cup \alpha|_{E'_1\gamma}$ is a unifier of E_1 and $E'_1\gamma$, then $\vartheta|_{E_1} \circ \delta = \sigma|_{E_1} \leq \vartheta|_{E_1} \circ \alpha|_{E_1\vartheta}$, and, therefore, $\delta \leq \alpha|_{E_1\vartheta}$. Then, we obtain $\delta|_{E_1\vartheta} = \alpha|_{E_1\vartheta} = mgu(E_1\vartheta, E'_1\gamma)|_{E_1\vartheta}$, and thus

$$\sigma|_{E_1} = \vartheta|_{E_1} \circ mgu(E_1\vartheta, E'_1\gamma)|_{E_1\vartheta} = (\vartheta \circ mgu(E_1\vartheta, E'_1\gamma))|_{E_1}. \quad \square$$

A.5. Proposition. *Let* $E_1, \ldots, E_m, E'_1, \ldots, E'_m$ *be expressions such that*

- *there exists* $\vartheta = mgu((E_1, \ldots, E_m), (E'_1, \ldots, E'_m))$,
- E_1 *and* E'_1 *do not share any variable with* $E'_2, \ldots, E'_m$,
- *there exists* $\delta = mgu(E_1, E'_1)$.

Then, $\vartheta = \delta \circ mgu((E_2, \ldots, E_m)\delta, (E'_2, \ldots, E'_m))$.

Proof. By definition, $(E_1, \ldots, E_m)\vartheta = (E'_1, \ldots, E'_m)\vartheta$, then $E_1\vartheta = E'_1\vartheta$, and therefore $\delta \leq \vartheta$. Let σ be the substitution such that $\delta \circ \sigma = \vartheta$. Hence, $(E_2, \ldots, E_m)\delta\sigma = (E'_2, \ldots, E'_m)\delta\sigma$. Then, since $Var(E'_1) \cap Var(E'_2, \ldots, E'_m) = \emptyset$, $Dom(\delta) \cap Var(E'_2, \ldots, E'_m) = \emptyset$ holds; thus $(E_2, \ldots, E_m)\delta\sigma = (E'_2, \ldots, E'_m)\sigma$. Therefore there exists $\alpha = mgu((E_2, \ldots, E_m)\delta, (E'_2, \ldots, E'_m))$ and $\alpha \leq \sigma$, from which it follows $\delta \circ \alpha \leq \delta \circ \sigma = \vartheta$. On the other side, $(E_2, \ldots, E_m)\delta\alpha = (E'_2, \ldots, E'_m)\alpha = (E'_2, \ldots, E'_m)\delta\alpha$, and $E_1\delta\alpha = E'_1\delta\alpha$, then $(E_1, \ldots, E_m)\delta\alpha = (E'_1, \ldots, E'_m)\delta\alpha$. Therefore, $\vartheta \leq \delta \circ \alpha$, and thus $\vartheta = \delta \circ \alpha$. $\square$

A.6. Proposition. *Let* E_1 *and* E_2 *be two expressions, and let* γ *be a substitution such that*

- E_1 *and* E_2 *do not share variables,*
- *Dom*(γ) *is contained in* $Var(E_2)$, *and* E_1 *and* $E_2\gamma$ *do not share variables,*
- *there exists* $\vartheta = mgu((E_1, E_2\gamma))$.

Then, there exists $\sigma = mgu((E_1, E_2))$ *and* $\sigma|_{E_1} \leq \vartheta|_{E_1}$ *holds.*

Proof. Immediate, since, by Proposition A.3, $\vartheta|_{E_1} \cup \gamma \circ \vartheta|E_2\gamma$ is a unifier of E_1 and E_2. $\square$

A.7. Proposition. *Let* E_1 *and* E_2 *be two expressions sharing no variables, and let* γ, δ *be two substitutions such that*

- *Dom*(γ) *and Dom*(δ) *are contained in* $Var(E_1)$,

- *$Im(\gamma)$ and $Im(\delta)$ do not share variables with E_2,*
- *there exists $\vartheta = mgu((E_1\gamma, E_2))$, and*
- *there exists $\sigma = mgu((E_1\delta, E_2))$,*
- *$\delta \leqslant \gamma$.*

Then, $(\delta \circ \sigma)|_{E_1} \leqslant (\gamma \circ \vartheta)|_{E_1}$.

Proof. Let ψ be the substitution such that $\delta \circ \psi = \gamma$. We have $E_1\delta\psi\vartheta = E_1\gamma\vartheta = E_2\vartheta$. By Proposition A.3, $(\psi \circ \vartheta)|_{E_1\delta} \cup \vartheta|_{E_2}$ is a unifier of $E_1\delta$ and E_2, then $\sigma|_{E_1\delta} \leqslant (\psi \circ \vartheta)|_{E_1\delta}$. Therefore $(\delta \circ \sigma)|_{E_1} \leqslant (\delta \circ \psi \circ \vartheta)|_{E_1\delta} = (\gamma \circ \vartheta)|_{E_1}$. □

References

[1] K.R. Apt and M. Van Emden, Contributions to the theory of logic programming, *J. ACM* **29** (1982) 841-862.

[2] M. Bruynooghe, G. Janssens, A. Callebaut and B. Demoen, Abstract interpretation: towards the global optimisation of Prolog programs, in: *Proc. 1987 Symp. on Logic Programming*, San Francisco, CA (IEEE Society Press, 1987) 192-204.

[3] K.L. Clark, Predicate logic as a computational formalism, Research Report 79/59, Dept. of Computing, Imperial College, 1979.

[4] P. Deransart and G. Ferrand, Programmation en logique avec negation: presentation formelle, Rapport de Recherche No. 87/3, Laboratoire d'Informatique, Département de Mathématiques et d'Informatique, Université d'Orléans, France, 1987.

[5] W. Drabent and J. Maluszynski, Inductive assertion method for logic programs, in: *Proc. TAPSOFT '87*, Lecture Notes in Computer Science **250** (Springer, Berlin, 1987) 167-181.

[6] M. Van Emden and R.A. Kowalski, The semantics of predicate logic as a programming language, *J. ACM* **23** (1976) 733-742.

[7] M. Falaschi and G. Levi, Operational and fixpoint semantics of committed-choice logic languages, Internal Report, Department of Computer Science, University of Pisa, Italy, 1987.

[8] G. Ferrand, A reconstruction of logic programming with negation, Rapport de Recherche No. 86/5, Laboratoire d'Informatique, Département de Mathématiques et d'Informatique, Université d'Orléans, France, 1986.

[9] G. Ferrand, Error diagnosis in logic programming, an adaptation of E.Y. Shapiro's method, *J. Logic Programming* **4** (1987) 177-198.

[10] W.G. Golson, Toward a declarative semantics for infinite objects in logic programming, Internal Report, Department of Computer Science, Rice University, Houston, TX, 1987.

[11] L. Henschen and L. Wos, Unit refutations and Horn sets, *J. ACM* **21** (1974) 590-605.

[12] J. Jaffar, J.-L. Lassez and M.J. Maher, A logic programming language scheme, in: D. De Groot and G. Lindström, eds., *Logic Programming: Relations, Functions and Equations* (Prentice-Hall, Englewood Cliffs, NJ, 1986) 441-468.

[13] J. Jaffar and J.-L. Lassez, Constraint Logic Programming, in: *Proc. SIGACT-SIGPLAN Symp. on Principles of Programming Languages* (1987) 111-119.

[14] G. Levi and C. Palamidessi, The declarative semantics of read-only variables, in: *Proc. 1985 Symp. on Logic Programming* (IEEE Computer Society Press, 1985) 128-137.

[15] G. Levi and C. Palamidessi, An approach to the declarative semantics of synchronization of logic languages, in: *Logic Programming. Proc. of the Fourth International Conference* (MIT Press, Cambridge, MA, 1987).

[16] G. Levi, A new declarative semantics for GHC, Technical Report, ICOT, January 1988.

[17] J. W. Lloyd, *Foundations of Logic Programming* (Springer, Berlin, 1984).

[18] H. Tamaki and T. Sato, Unfold/fold transformation of logic programs, in: *Proc. 2nd Internat. Logic Programming Conf.*, Uppsala, Sweden (1984) 127-138.

[19] S. Yamasaki, M. Yoshida and S. Doshita, A fixpoint semantics of Horn sentences based on substitution sets, *Theoret. Comput. Sci.* **51** (1987) 309-324.