PROCEEDINGS OF THE INTERNATIONAL CONFERENCE
ON FIFTH GENERATION COMPUTER SYSTEMS 1992,
edited by ICOT.

Contributions to the Semantics of Open Logic Programs

A. Bossi[1], M. Gabbrielli[2], G. Levi[2] and M.C. Meo[2]

1) Dipartimento di Matematica Pura ed Applicata
Università di Padova
Via Belzoni 7, I-35131 Padova, Italy
mat010@IPDUNIVX.UNIPD.IT

2) Dipartimento di Informatica
Università di Pisa
Corso Italia 40, 56125 Pisa
{gabbri,levi,meo}@dipisa.di.unipi.it

Abstract

The paper considers *open logic programs* originally introduced in [Bossi and Menegus 1991] as a tool to build an OR-compositional semantics of logic programs. We extend the original semantic definitions in the framework of the general approach to the semantics of logic programs described in [Gabbrielli and Levi 1991b]. We first define an *OR-compositional operational semantics* $\mathcal{O}_\Omega(P)$ modeling computed answer substitutions. We consider next the semantic domain of *Ω-interpretations*, which are sets of clauses with a suitable equivalence relation. The fixpoint semantics $\mathcal{F}_\Omega(P)$ given in [Bossi and Menegus 1991] is proved equivalent to the operational semantics, by using an intermediate unfolding semantics. From the model-theoretic viewpoint, an Ω-interpretation is mapped onto a set of Herbrand interpretation, thus leading to a definition of *Ω-model* based on the classical notion of truth. We show that under a suitable partial order, the glb of a set of Ω-models of a program P is an Ω-model of P. Moreover, the glb of all the Ω-models of P is equal to the usual Herbrand model of P while $\mathcal{F}_\Omega(P)$ is a (non-minimal) Ω-model.

1 Introduction

An *Ω-open program* [Bossi and Menegus 1991] P is a program in which the predicate symbols belonging to the set Ω are considered partially defined in P. P can be composed with other programs which may further specify the predicates in Ω. Such a composition is denoted by $\cup_\Omega$. Formally, if $Pred(P) \cap Pred(Q) \subseteq \Omega$ then $P \cup_\Omega Q = P \cup Q$, otherwise $P \cup_\Omega Q$ is not defined ($Pred(P)$ denotes the predicate symbols in P). A typical partially defined program is a program where the intensional definitions are completely known while extensional definitions are only partially known and can be further specified.

Example 1.1 *Let us consider the following program*

$$\begin{aligned} Q_1 = \{\ & anc(X,Y) : -parent(X,Y). \\ & anc(X,Z) : -parent(X,Y), anc(Y,Z). \\ & parent(isaac, jacob). \\ & parent(jacob, benjamin). \qquad \} \end{aligned}$$

New extensional information defining new parent tuples can be added to Q_1 as follows

$$\begin{aligned} Q_2 = \{\ & parent(anna, elizabeth). \\ & parent(elizabeth, john). \quad \} \end{aligned}$$

The semantics of open programs must be Ω-compositional w.r.t. program union, i.e. the semantics of $P_1 \cup_\Omega P_2$ must be derivable from the semantics of P_1 and P_2. If Ω contains all the predicates in P, Ω-compositionality is the same as compositionality.

The least Herbrand model semantics, as originally proposed [van Emden and Kowalski 1976] and the computed answer substitution semantics in [Falaschi et al. 1988,Falaschi et al. 1989a], are not compositional w.r.t. program union. For example, in example 1.1, the atom $anc(anna, elizabeth)$ which belongs to the least Herbrand model semantics of $Q_1 \cup Q_2$ cannot be obtained from the least Herbrand model semantics of Q_1 and Q_2 (see also example 2.1).

In this paper we will introduce a semantics for Ω-open programs following the general approach in [Gabbrielli and Levi 1991b] which leads to semantics definitions which characterize the program operational behavior. This approach leads to the introduction of extended interpretations (*π-interpretations*) which are more expressive than Herbrand interpretations. The improved expressive power is obtained by accommodating more syntactic objects in π-interpretations, which are (possibly

infinite) programs. The semantics in terms of π-interpretations can be computed both operationally and as the least fixpoint of suitable continuous immediate consequence operators on π-interpretations. It can also be characterized from the model-theoretic viewpoint, by defining a set of extended models (π-models) which encompass standard Herbrand models. In the specific case of Ω-open programs, extended interpretations are called Ω-interpretations and are sets of *conditional atoms* (i.e. clauses such that all the atoms in the body are open). Each Ω-interpretation represents a set of Herbrand interpretations that could be obtained by composing the open program with a definition for the open predicates. Ω-interpretations of open programs are introduced to obtain a unique representative model, computable as the least fixpoint of a suitable continuous operator, in cases where no such a representative exists in the set of Herbrand models.

The main contribution of this paper is the definition of an OR-compositional (i.e. compositional w.r.t. program union) semantics of logic programs in the style of [Falaschi et al. 1988, Falaschi et al. 1989b]. Other approachs to OR-compositionality can be found in [Lassez and Maher 1984, Mancarella and Pedreschi 1988, Gaifman and Shapiro 1989a, Gaifman and Shapiro 1989b]. An OR-compositional semantics corresponds to an important program equivalence notion, according to which two programs P_1 and P_2 are equivalent iff for any program Q a generic goal G computes the same answers in $P_1 \cup Q$ and $P_2 \cup Q$. An OR-compositional semantics has also some interesting applications. Namely it can be used

- to model logic languages provided with a module-like structure,
- to model incomplete knowledge bases, where new chunks of knowledge can incrementally be assimilated,
- for program transformation
  (the transformed programs must have the same OR-compositional semantics of the original program),
- for semantics-based "modular" program analysis.

The paper is organized as follows. Subsection 1.1 contains notation and useful definitions on the semantics of logic programs. In section 2 we define an *operational semantics* $\mathcal{O}_\Omega(P)$ modeling computed answer substitutions which is *OR-compositional*. Section 3 introduces a suitable *semantic domain* for the $\mathcal{O}_\Omega(P)$ semantics and defines *Ω-interpretations* which are sets of clauses modulo a suitable equivalence relation. In section 4 the *fixpoint semantics* $\mathcal{F}_\Omega(P)$, is proved equivalent to the operational semantics by using an intermediate *unfolding semantics*. Section 5 is concerned with *model theory*. From the model-theoretic viewpoint, an Ω-interpretation is mapped onto a set of Herbrand interpretations, thus leading to a definition of Ω-model based on the classical notion of truth. We show that under a suitable partial order, the glb of a set of Ω-models of a program P is an Ω-model of P. Moreover, the glb of all the Ω-models of P is equal to the usual Herbrand model of P. Moreover, $\mathcal{F}_\Omega(P)$ is a (non-minimal) Ω-model, equivalent to the model-theoretic semantics defined in [Bossi and Menegus 1991] in terms of S_Ω-models. A comparison between Ω-models and the S_Ω-models is made in section 6. Section 7 is devoted to some conclusive remarks. All the proofs of the results given here can be found in [Bossi et al. 1991].

1.1 Preliminaries

The reader is assumed to be familiar with the terminology of and the basic results in the semantics of logic programs [Lloyd 1987,Apt 1988]. Let the signature S consist of a set F of *function symbols*, a finite set P of *predicate symbols*, a denumerable set V of *variable symbols*. All the definitions in the following will assume a given signature S. Let T be the set of terms built on F and V. Variable-free terms are called *ground*. A substitution is a mapping $\vartheta : V \to T$ such that the set $D(\vartheta) = \{X \mid \vartheta(X) \neq X\}$ (*domain* of ϑ) is finite. If $W \subset V$, we denote by $\vartheta_{|W}$ the *restriction* of ϑ to the variables in W, i.e. $\vartheta_{|W}(Y) = Y$ for $Y \notin W$. ε denotes the empty substitution. The *composition* $\vartheta\sigma$ of the substitutions ϑ and σ is defined as the functional composition. A *renaming* is a substitution ρ for which there exists the inverse ρ^{-1} such that $\rho\rho^{-1} = \rho^{-1}\rho = \varepsilon$. The pre-ordering $\leq$ (more general than) on substitutions is such that $\vartheta \leq \sigma$ iff there exists ϑ' such that $\vartheta\vartheta' = \sigma$. The result of the application of the substitution ϑ to a term t is an *instance* of t denoted by $t\vartheta$. We define $t \leq t'$ (t is more general than t') iff there exists ϑ such that $t\vartheta = t'$. A substitution ϑ is *grounding* for t if $t\vartheta$ is ground. The relation $\leq$ is a preorder. $\approx$ denotes the associated equivalence relation (*variance*). A substitution ϑ is a *unifier* of terms t and t' if $t\vartheta \equiv t'\vartheta$. The *most general unifier* of t_1 and t_2 is denoted by $mgu(t_1, t_2)$. All the above definitions can be extended to other syntactic expressions in the obvious way. An atom is an object of the form $p(t_1, \ldots, t_n)$ where $p \in P$. $t_1, \ldots, t_n \in T$.

A *clause* is a formula of the form $H : -L_1, \ldots, L_n$ with $n \geq 0$, where H (the *head*) and $L_1, \ldots, L_n$ (the *body*) are atoms. ": −" and "," denote logic implication and conjunction respectively, and all variables are universally quantified. If the body is empty the clause is a *unit clause*. A *program* is a finite set of clauses. A *goal* is a formula $L_1, \ldots, L_m$, where each L_i is an atom. By $Var(E)$ and $Pred(E)$ we denote respectively the sets of variables and predicates occurring in the expression E. A *Herbrand interpretation* I for a program P is a set of ground atoms. The intersection $M(P)$ of all the Herbrand models of a program P is a model (least Herbrand model). $M(P)$ is also the least fixpoint of a continuous transformation T_P (*immediate consequences operator*) on the complete lattice of Herbrand interpretations. If G is a goal, $G \overset{\vartheta}{\leadsto}_P B_1, \ldots, B_n$ denotes an SLD derivation with fair selection rule of $B_1, \ldots, B_n$ in the program P where ϑ is the composition of the mgu's used in the derivation. $G \overset{\vartheta}{\longmapsto}_P \square$ denotes the refutation of G in the program P with computed answer substitution ϑ. A computed answer substitution is always restricted to the variables occurring in G. The notations $\tilde{t}$, $\tilde{X}$ will be used to denote tuples of terms and variables respectively, while $\tilde{B}$ denotes a (possibly empty) conjunction of atoms.

2 Computed answer substitution semantics for Ω-open programs

The operational semantics is usually given by means of a set of inference rules which specify how derivations are made. From a purely logical point of view the operational semantics is simply defined in terms of successful derivations. However, from a programming language viewpoint, the operational semantics must be concerned with additional information, namely observable properties. A given program in fact may have different semantics depending on which of its properties can be observed. For instance in pure logic programs one can observe successes, finite failure, computed answer substitutions, partial computed answer substitutions or any combination of them. A given choice of the observable induces an equivalence on programs, namely two programs are equivalent iff they are observationally indistinguishable. When the semantics correctly captures the observable, two programs are equivalent if they have the same semantics. When also compositionality is taken into account, for a given observable property we can obtain different semantics (and equivalence relations) depending on which kind of program composition we consider. Indeed, the semantics of logic programs is usually concerned with AND-composition (of atoms in a goal or in a clause body). Consider for example logic programs with computed answer substitutions as observable [Falaschi et al. 1989a]. The operational semantics can be defined as

$\mathcal{O}(P) = \{p(\tilde{X})\theta \mid \tilde{X}$ are distinct var, $p(\tilde{X}) \overset{\vartheta}{\longmapsto}_P \square\}$

where the denotation of a program is a set of non-ground atoms, which can be viewed as a possibly infinite program [Falaschi et al. 1989a]. Since we have syntactic objects in the semantic domain, we need an equivalence relation in order to abstract from irrelevant syntactic differences. If the equivalence is accurate enough the semantics is fully abstract. According to [Gabbrielli and Levi 1991b], Herbrand interpretations are generalized by π-interpretations which are possibly infinite sets of (equivalence classes of) clauses. The operational semantics of a program P is then a π-interpretation I, which has the following property. P and I are observationally equivalent with respect to any goal G. This is the property which allows to state that the semantics does indeed capture the observable behavior [Falaschi et al. 1989a]. The following example shows that when considering OR-composition (i.e. union of sets of clauses), non-ground atoms (or unit clauses) are not sufficient any longer to define a compositional semantics.

Example 2.1 *Let us consider the following programs*

$$P_1 = \{\ q(X) : -p(X).\quad r(X) : -s(X).\quad s(b).\quad p(a).\ \}\qquad P_2 = \{\ p(b).\ \}$$

According to the previous definition of $\mathcal{O}(P)$, $\mathcal{O}(P_1) = \{p(a), q(a), r(b), s(b)\}$ and $\mathcal{O}(P_2) = \{p(b)\}$. Since $\mathcal{O}(P_1 \cup P_2) = \{p(a), p(b), q(a), q(b), r(b), s(b)\}$, the semantics of the union of the two programs cannot be obtained from the semantics of the programs.

In order for a semantics to be compositional, it must contain information in the form of a mapping from sets of atoms to sets of atoms. This is indeed the case of the semantics based on the closure operator [Lassez and Maher 1984] and on the T_P operator [Mancarella and Pedreschi 1988]. If we want a semantics expressed by the program syntax, OR-compositionality can only be obtained by choosing as semantic domain a set of (equivalence classes of) clauses. In example 2.1, for instance, the semantics of P_1 should contain also the clause $q(X) : -p(X)$.

Let us formally give the definition of the program composition we consider.

Definition 2.2 *Let P be a program and Ω be a set of predicate symbols. P is open w.r.t. Ω (or Ω-open) if the information on the predicates in Ω is considered to be partial. Moreover if P, Q are Ω-open programs and $(Pred(Q) \cap Pred(P)) \subseteq \Omega$ then $P \cup_\Omega Q$ is the Ω-open program $P \cup Q$. If $(Pred(Q) \cap Pred(P)) \not\subseteq \Omega$ then $P \cup_\Omega Q$ is not defined.*

Note that when considering an Ω-open program P and an Ω'-open program Q, the composition of P and Q is defined only if $(Pred(Q) \cap Pred(P)) \subseteq (\Omega \cap \Omega')$. Moreover, the composition of P and Q is a Ψ-open program, where $\Psi = \Omega \cup \Omega'$.

The definition of any predicate symbol $p \in \Omega$ in an Ω-open program P can always be extended or refined. For instance in example 1.1 program Q_1 is open w.r.t. the predicate *parent* and this predicate is refined in program Q_2. Therefore, a deduction concerned with a predicate symbol of an Ω-open program P can be either *complete* (when it takes place completely in the program P) or *partial* (when it terminates in P with an atom $p(\tilde{t})$ such that $p \in \Omega$ and $p(\tilde{t})$ does not unify with the head of any clause in P). A partial deduction can be completed by the addition of new clauses. Thus we have an *hypothetic deduction*, conditional on the extension of predicate p.

Let us consider again the program P_1 of example 2.1 and assume $\Omega = \{p\}$. Then, the goal $r(X)$ produces a complete deduction only, computing the answer substitution $\{X/b\}$. The goal $q(X)$ produces a complete deduction, computing the answer substitution $\{X/a\}$ and an hypothetical deduction returning any answer that could be computed by a definition of p external to P_1. The goal $q(b)$ instead has one hypothetical deduction only, conditional on the provability (outside P_1) of $p(b)$. We want to express this hypothetical reasoning, i.e. that $q(b)$ is refutable if $p(b)$ is refutable. Hence we will consider the following operational semantics (recall that by $\tilde{B}$ we denote $B_1, \ldots, B_n$ with $n \geq 0$).

Definition 2.3 *Let Ω be a set of predicate symbols. We define*

$$Id(\Omega) = \{p(\tilde{X}) : -p(\tilde{X}) \mid \; p \in \Omega, \tilde{X} \text{ are distinct variables } \}$$

Definition 2.4 (Ω-compositional computed answer substitutions semantics) *Let P be a program and let $P^\bullet = P \cup Id(\Omega)$. Then we define $\mathcal{O}_\Omega(P) =$*

$$\{A : -\tilde{B}_2 \mid \; p(\tilde{X}) \overset{\vartheta}{\rightsquigarrow}_P \tilde{B}_1 \overset{\gamma}{\rightsquigarrow}_{P^\bullet} \tilde{B}_2 \;\; \tilde{X} \text{ distinct variables}, \;\; A = p(\tilde{X})\vartheta\gamma, \;\; \{Pred(\tilde{B}_2)\} \subseteq \Omega\}$$

The set of clauses $Id(\Omega)$ in the previous definition is used to delay the evaluation of open atoms. This is a trick which allows to obtain by using a fixed fair selection rule R, all the derivations $p(X_1, \ldots, X_n) \overset{\vartheta}{\rightsquigarrow}_P B_1, \ldots, B_n$ which use any selection rule R', for $Pred(B_1, \ldots, B_n) \subseteq \Omega$. Note that the first step of the derivation uses a clause in P (instead than in $P^\bullet$) because we want $\mathcal{O}_\Omega(P)$ to contain a clause $p(\tilde{X}) : -p(\tilde{X})$ if and only if $p(\tilde{X}) \overset{\varepsilon}{\rightsquigarrow}_P p(\tilde{X})$.

Example 2.5 *Let P_1, P_2 be the Ω-open programs of example 2.1 where $\Omega = \{p\}$.*
Then $\mathcal{O}_\Omega(P_2) = \{p(b)\}$ and
$\mathcal{O}_\Omega(P_1) = \{q(X) : -p(X),\ p(a),\ q(a),\ r(b),\ s(b)\}$.
$\mathcal{O}_\Omega$ contains enough information to compute the semantics of compositions. Indeed $\mathcal{O}(P_1 \cup P_2) \subseteq \mathcal{O}_\Omega(P_1 \cup P_2)$ and $\mathcal{O}_\Omega(P_1 \cup P_2) = \mathcal{O}_\Omega(\mathcal{O}_\Omega(P_1) \cup \mathcal{O}_\Omega(P_2))$ (see theorem 2.9).

Example 2.6 *Let $\Omega = \{q, r\}$ and let Q_1, Q_2 be the following programs*

$$Q_1 = \{p(X,Y) : -r(X), q(Y). \quad r(a). \quad \} \qquad Q_2 = \{ \quad r(b). \quad \}$$

Then $\mathcal{O}_\Omega(Q_2) = \{r(b)\}$, $\mathcal{O}_\Omega(Q_1) =$
$\{p(X,Y) : -r(X), q(Y),\ p(a,Y) : -q(Y),\ r(a)\}$ and
$\mathcal{O}_\Omega(Q_1 \cup Q_2) = \mathcal{O}_\Omega(\mathcal{O}_\Omega(Q_1) \cup \mathcal{O}_\Omega(Q_2)) =$
$\{p(X,Y) : -r(X), q(Y),\ p(a,Y) : -q(Y),$
$p(b,Y) : -q(Y),\ r(a), r(b)\}$ (see theorem 2.9).

Note that $\mathcal{O}_\Omega(P)$ is essentially the result of the partial evaluation [Lloyd and Shepherdson 1987] of P, where derivations terminate at open predicates. This operational semantics fully characterizes hypothetic deductions, conditional on the extension of the predicates in Ω. Indeed the semantics of a program P can be viewed as a possibly infinite set of clauses and the partial computed answer substitutions can be obtained by executing the goal in the "program". The equivalence ($\cong_\Omega$) on programs induced by the computed answer substitution observable when considering also programs union, can be formally defined as follows.

Definition 2.7 *Let P_1, P_2 be Ω-open programs. Then $P_1 \cong_\Omega P_2$ if for every goal G and for every program Q s.t. $P_i \cup_\Omega Q$, $i = 1, 2$, is defined, $G \overset{\vartheta}{\longmapsto}_{P_1 \cup_\Omega Q} \square$ iff $G \overset{\vartheta\rho}{\longmapsto}_{P_2 \cup_\Omega Q} \square$ where ρ is a renaming.*

$\mathcal{O}_\Omega$ allows to characterize a notion of answer substitution which enhances the usual one, since also (unresolved) atoms, with predicate symbols in Ω, are considered. Therefore it is able to model computed answer substitutions in an OR compositional way. The following results show that $\mathcal{O}_\Omega(P)$ is compositional w.r.t. $\cup_\Omega$ and therefore it correctly captures

the computed answer substitution observable notion when considering also programs union.

Theorem 2.8 *Let P be an Ω-open program. Then $P \cong_\Omega \mathcal{O}_\Omega(P)$.*

Theorem 2.9 *Let P_1, P_2 be Ω-open programs and let $P_1 \cup_\Omega P_2$ be defined. Then $\mathcal{O}_\Omega(\mathcal{O}_\Omega(P_1) \cup_\Omega \mathcal{O}_\Omega(P_2)) = \mathcal{O}_\Omega(P_1 \cup_\Omega P_2)$.*

Corollary 2.10 *Let P_1, P_2 be Ω-open programs. If $\mathcal{O}_\Omega(P_1) = \mathcal{O}_\Omega(P_2)$ then $P_1 \cong_\Omega P_2$.*

3 Semantic domain for Ω-open programs

In this section we formally define the semantic domain which characterizes the above introduced operational semantics $\mathcal{O}_\Omega$. Since $\mathcal{O}_\Omega$ contains clauses (whose body predicates are all in Ω), we have to accommodate clauses in the interpretations we use. Therefore we will define the notion of Ω-interpretation which extends the usual notion of interpretation since an Ω-interpretation contains conditional atoms. As usual, in the following, Ω is a set of predicates.

Definition 3.1 (Conditional atoms)
An Ω-conditional atom is a clause $A : -B_1, \ldots, B_n$ such that $Pred(B_1, \ldots, B_n) \subseteq \Omega$.

In order to abstract from the purely syntactical details, we use the following equivalence $\approx$ on conditional atoms.

Definition 3.2 *Let $c_1 = A_1 : -B_1, \ldots, B_n$, $c_2 = A_2 : -D_1, \ldots, D_m$ be clauses. Then $c_1 \leq c_2$ iff $\exists \vartheta$ such that $\exists \{i_1, \ldots, i_n\} \subseteq \{1, \ldots, m\}$ such that $A_1\vartheta = A_2$, $i_h \neq i_k$ for $h \neq k$, and $(B_1\vartheta, \ldots, B_n\vartheta) = (D_{i_1}, \ldots, D_{i_n})$. Moreover we define $c_1 \approx c_2$ iff $c_1 \leq c_2$ and $c_2 \leq c_1$.*

Note that in the previous definition bodies of clauses are considered as multisets (considering sets would give the standard definition of subsumption). Equivalent clauses have the same body (considered as a multiset) up to renaming. Considering sets instead of multisets (subsumption equivalence) is not correct when considering computed answer substitutions. The following is a simple counterexample.

Example 3.3 *Let $c_1 = p(X,Y) : -q(X,Y), q(X,Y)$ and $c_2 = p(X,Y) : -q(X,Y)$. Let $P_1 = \{c_1\}$ and $P_2 = \{c_2\}$ be Ω-open programs where $\Omega = \{q\}$. Obviously, considering bodies of clauses as sets, $c_1 = c_2\epsilon$ where ϵ is the empty renaming. However, $P_1 \not\cong_\Omega P_2$ since by considering $Q = \{q(X,b), q(a,Y)\}$, $p(X,Y) \overset{\vartheta}{\mapsto}_{P_1 \cup Q} \square$ where $\vartheta = \{X/a, Y/b\}$, while the goal $p(X,Y)$ in the program $P_2 \cup Q$ can compute either $\{X/a\}$ or $\{Y/b\}$ only.*

Definition 3.4 *The Ω-conditional base, $\mathcal{C}_\Omega$, is the quotient set of all the Ω-conditional atoms w.r.t. $\approx$.*

In the following we will denote the equivalence class of a conditional atom c by c itself, since all the definitions which use conditional atoms are not dependent on the element chosen to represent an equivalence class. Moreover, any subset of $\mathcal{C}_\Omega$ will be considered implicitly as an Ω-open program. Before giving the formal definition of Ω-interpretation, we need the notion of *u-closed subset* of $\mathcal{C}_\Omega$.

Definition 3.5 *A subset I of $\mathcal{C}_\Omega$ is u-closed iff $\forall H : -B_1, \ldots, B_n \in I$ and $\forall B : -A_1, \ldots, A_m \in I$ such that $\exists \vartheta = mgu(B_i, B)$, for $1 \leq i \leq n$, $(H : -B_1, \ldots, B_{i-1}, A_1, \ldots, A_m, B_{i+1}, \ldots, B_n)\vartheta \in I$. Moreover if $I \subseteq \mathcal{C}_\Omega$, we denote by $\widehat{I}$ its u-closure defined as the least (w.r.t. $\subseteq$) $I' \subseteq \mathcal{C}_\Omega$ u-closed such that $I \subseteq I'$.*

Proposition 4.5 will show that the previous notion of u-closure is well defined. A u-closed interpretation I is an interpretation which, if viewed as a program, is closed under unfolding of procedure calls. Interpretations need to be u-closed for the validity of the model theory developed in section 5. Therefore, in order to define Ω-interpretations we will consider u-closed sets of conditional atoms only. Let us now give the formal definition of Ω-interpretation.

Definition 3.6 *An Ω-interpretation I is any subset of $\mathcal{C}_\Omega$ which is u-closed. The set of all the Ω-interpretations is denoted by $\Im$.*

Lemma 3.7 *$(\Im, \subseteq)$ is a complete lattice where the minimal element is $\emptyset$ and $glb(X) = \widehat{\bigcup_{x \in X} x}$ for any $X \subseteq \Im$.*

In the following the operational semantics $\mathcal{O}_\Omega$ will be formally considered as an Ω-interpretation.

4 Fixpoint semantics

In this section we define a fixpoint semantics $\mathcal{F}_\Omega(P)$ which in the next subsection is proved to be equivalent to the previously defined operational semantics $\mathcal{O}_\Omega(P)$. This can be achieved by defining an immediate consequence operator T_P^Ω on the lattice $(\Im, \subseteq)$ of Ω-interpretations. $\mathcal{F}_\Omega(P)$ is the least fixpoint of T_P^Ω.

The immediate consequences operator T_P^Ω is strongly related to the derivation rule used for Ω-open programs and hence to the unfolding rule. Therefore T_P^Ω models the observable properties in an OR compositional way, and may be useful for modular (i.e. OR compositional) bottom-up program analysis.

Definition 4.1 *Let P be an Ω-open program. Then $T_P^\Omega(I) = \widehat{\Gamma_P^\Omega(I)}$ where $\Gamma_P^\Omega(I)$ is the operator defined in [Bossi and Menegus 1991] as follows.*

$\Gamma_P^\Omega(I) =$
$\{(A : -\tilde{L}_1, \ldots, \tilde{L}_n)\vartheta \in \mathcal{C}_\Omega \mid$
$\exists A : -B_1, \ldots, B_n \in P,$
$\exists B_i' : -\tilde{L}_i \in I \cup Id(\Omega),\ i = 1, \ldots, n,\ m_i \geq 0$
$s.t.\ \ \vartheta = mgu((B_1, \ldots, B_n), (B_1', \ldots, B_n'))\}$

Proposition 4.2 *T_P^Ω is continuous in the complete lattice $(\Im, \subseteq)$.*

The notion of ordinal powers for T_P^Ω is defined as usual, namely $T_P^\Omega \uparrow 0 = \emptyset$, $T_P^\Omega \uparrow n+1 = T_P^\Omega(\ T_P^\Omega \uparrow n\ )$ and $T_P^\Omega \uparrow \omega = \bigcup_{n \geq 0}(\ T_P^\Omega \uparrow n\ )$. Since T_P^Ω is continuous on $(\Im, \subseteq)$, well known results of lattice theory allow to prove proposition 4.3 and hence to define the fixpoint semantics as follows.

Proposition 4.3 *$T_P^\Omega \uparrow \omega$ is the least fixpoint of T_P^Ω in the complete lattice $(\Im, \subseteq)$.*

Definition 4.4 *Let P be an Ω-open program. The fixpoint semantics $\mathcal{F}_\Omega(P)$ of P is defined as $\mathcal{F}_\Omega(P) = T_P^\Omega \uparrow \omega$.*

Remark
The original definition of $\Gamma_P^\Omega(I)$ does not require Ω-interpretations to be u-closed subsets of $\mathcal{C}_\Omega$. If we consider an Ω-interpretation as any subset of $\mathcal{C}_\Omega$ and the Γ_P^Ω operator, even if the intermediate results $\Gamma_P^\Omega \uparrow n$ are different, the following proposition 4.5 and theorem 4.6 show that the least fixpoint $\Gamma_P^\Omega \uparrow \omega$ is a u-closed set and it is equal to $\mathcal{F}_\Omega(P)$ (Γ_P^Ω is continuous on $(\wp(\mathcal{C}_\Omega), \subseteq)$). Therefore, when considering the fixpoint semantics we can use the Γ_P^Ω operator. Moreover, proposition 4.5 ensures us that the previous notion of u-closure is well defined.

Proposition 4.5 *Let $I \subseteq \mathcal{C}_\Omega$ and let $\Gamma_I^\Omega(I)$ be defined as in definition 4.7. Then the following hold*

1. *I is u-closed iff $I = \Gamma_I^\Omega(I)$,*
2. *for any program P, $\Gamma_P^\Omega \uparrow \omega$ is u-closed,*
3. *$I' = \Gamma_I^\Omega \uparrow \omega$ is the least (w.r.t. set inclusion) subset of $\mathcal{C}_\Omega$ such that it is u-closed and $I \subseteq I'$.*

Theorem 4.6 *Let P an Ω-open program. $\Gamma_P^\Omega \uparrow \omega = \mathcal{F}_\Omega(P)$.*

4.1 Unfolding semantics and equivalence results

To clarify the relations between the operational and the fixpoint semantics, before proving their equivalence, we introduce the intermediate notion of *unfolding semantics* $\mathcal{U}_\Omega(P)$ [Levi 1988, Levi and Mancarella 1988]. $\mathcal{U}_\Omega(P)$ is obtained as the limit of the unfolding process. Since the unfolding semantics can be expressed top-down in terms of the Γ_P^Ω operator, the unfolding semantics can be proved equal to the standard bottom-up fixpoint semantics. On the other hand, since $\mathcal{U}_\Omega(P)$ and $\mathcal{O}_\Omega(P)$ are based on the same inference rule (applied in parallel and in sequence respectively) $\mathcal{U}_\Omega(P)$ and $\mathcal{O}_\Omega(P)$ can easily be proven equivalent.

Definition 4.7 *Let P and Q be Ω-open programs. Then the unfolding of P w.r.t. Q is defined as*

$unf_P^\Omega(Q) =$
$\{(A : -\tilde{L}_1, \ldots, \tilde{L}_n)\vartheta \mid$
$\exists A : -B_1, \ldots, B_n \in P,$
$\exists B_i' : -\tilde{L}_i \in I \cup Id(\Omega),\ i = 1, \ldots, n,\ m_i \geq 0$
$s.t.\ \ \vartheta = mgu((B_1, \ldots, B_n), (B_1', \ldots, B_n'))\}$

Note that the only difference between $unf_P^\Omega(Q)$ and $\Gamma_P^\Omega(Q)$ is that the second restricts to clauses in $\mathcal{C}_\Omega$ the set resulting from the definition. Therefore if I is an Ω-interpretation (i.e. $I \subseteq \mathcal{C}_\Omega$), $\Gamma_P^\Omega(I) = unf_P^\Omega(I)$ holds. In general, $\Gamma_P^\Omega(I) = \iota_\Omega(unf_P^\Omega(I))$ where $\iota_\Omega(P)$ extracts from a program P an Ω-interpretation.

Definition 4.8 *Let P be an Ω-open program. Then we define*

$$\iota_\Omega(P) = \{c \in P \mid c \in \mathcal{C}_\Omega\}.$$

Definition 4.9 *Let P be an Ω-open program and let $\iota_\Omega(P)$ be as defined in definition 4.8. Then we define the collection of programs*

$$P_0 = P$$
$$P_i = unf_{P_{i-1}}(P)$$

The unfolding semantics $\mathcal{U}_\Omega(P)$ of the program P is defined as

$$\mathcal{U}_\Omega(P) = \bigcup_{i=1,2,\ldots} \iota_\Omega(P_i).$$

The following theorem states the equality of the unfolding and the operational semantics.

Theorem 4.10 *Let P be an Ω-open program. Then $\mathcal{O}_\Omega(P) = \mathcal{U}_\Omega(P)$.*

Note that $\Gamma_P^\Omega \uparrow n+1 = unf_{P_n'}^\Omega(\emptyset)$, where $P_0' = P$ and $P_{i+1}' = unf_P^\Omega(P_i')$. Therefore we have the following theorem.

Theorem 4.11 *Let P be a program. Then $\mathcal{F}_\Omega(P) = \mathcal{U}_\Omega(P)$.*

Corollary 4.12 *Let P be a program. Then $\mathcal{F}_\Omega(P) = \mathcal{O}_\Omega(P)$.*

5 Model Theory

As we have shown, the operational and fixpoint semantics of a program P define an Ω-interpretation I_P, which can be viewed as a syntactic notation for a set of Herbrand interpretations denoted by $\mathcal{H}(I_P)$. Namely, $\mathcal{H}(I_P)$ represents the set of the least Herbrand models of all programs which can be obtained by closing the program I_P with a suitable set of ground atoms defining the open predicates. Our aim is finding a notion of Ω-model such that $\mathcal{O}_\Omega(P)$ (and $\mathcal{F}_\Omega(P)$) are Ω-models and every Herbrand model is an Ω-model. This can be obtained as follows.

Definition 5.1 *Let J be an Ω-interpretation. Then we define*

$$Atom_\Omega(J) = \{p(\tilde{t}) \mid \ p \in \Omega \text{ and } p(\tilde{t}) \text{ is a ground instance of an atom in } J\}.$$

Example 5.2 *Let $\Omega = \{p, q\}$ and $J = \{p(a) : -q(b)\}$. Then $Atom_\Omega(J) = \{p(a), q(b)\}$.*

Definition 5.3 *Let I be an Ω-interpretation for an Ω-open program. Then we define*

$$\mathcal{H}(I) = \{M(I \cup J) \mid J \subseteq Atom_\Omega(I)\}$$

where $M(K)$ denotes the least Herbrand model of K.

Example 5.4 *Let $I = \{p(a) : -q(b)\}$ be an Ω-interpretation. Then*

1) *for $\Omega = \{q\}$*
   $Atom_\Omega(I) = \{q(b)\}$ and
   $\mathcal{H}(I) = \{\emptyset, \{p(a), q(b)\}\}$.

2) *for $\Omega = \{p, q\}$*
   $Atom_\Omega(I) = \{p(a), q(b)\}$ and
   $\mathcal{H}(I) = \{\emptyset, \{p(a)\}, \{p(a), q(b)\}\}$.

Definition 5.5 *Let P be an Ω-open program and I be an Ω-interpretation. I is an* Ω-model *of P iff $\forall\, J \in \mathcal{H}(I)$, J is a Herbrand model of P.*

Obviously, in general given a Herbrand model M of a program P, $M \cup N$ is not anymore a model of P for an arbitrary set of ground atoms N. Since we want a notion of Ω-model which encompasses the standard notion of Herbrand model, the "closure" of the interpretation I can be performed by adding only ground atoms which unify with atoms already in I. The following example 5.6 shows that if such a condition is not satisfied, a standard Herbrand model would not any more be an Ω-model.

Example 5.6 *Let us consider the Ω-open program $P = \{p(a) : -q(a)\}$ where $\Omega = \{q\}$. Then $\emptyset$ is a (the least) Herbrand model of P. If, by violating the $J \subseteq Atom_\Omega(I)$ condition, $\{q(a)\} \in \mathcal{H}(\emptyset)$, since $\{q(a)\}$ is not a Herbrand model of P, $\emptyset$ would not be an Ω-model of P.*

Example 5.7 *Let us consider the program P_1 where $\Omega = \{p\}$ of the example 2.1. Then*
$\mathcal{O}_\Omega(P_1) = \{q(X) : -p(X),\ p(a),\ q(a),\ r(b),\ s(b)\}$
is an Ω-model of P_1 since
$\mathcal{H}(\mathcal{O}_\Omega(P_1)) = \{H_1, H_2, H_3, \ldots\}$
where, denoting by $[p(X)]$ the set of ground instances of $p(X_0)$,
$H_1 = \{p(a), q(a), r(b), s(b)\}$
$H_2 = \{p(a), p(b), q(a), q(b), r(b), s(b)\}$
$\vdots$
$H_\omega = \{r(b), s(b)\} \cup [p(X)] \cup [q(X)]\}$
and $H_1, H_2, \ldots, H_\omega$ are Herbrand models of P_1.

The following proposition states the mentioned properties of Ω-models.

Proposition 5.8 *Let $P = \{c_1, \ldots, c_n\}$ be an Ω-open program. Then*

1. *every Herbrand model of P is an Ω-model of P,*
2. *$\mathcal{O}_\Omega(P)$ is an Ω-model of P.*

A relevant property of standard Herbrand models states that the intersection of a set of models of a program P is always a model of P. This allows to define the model-theoretic semantics of P as the least Herbrand model obtained by intersecting all the Herbrand models of P. The following example shows that this important property does not hold any more when considering Ω-models with set theoretic operations.

Example 5.9 *Let $\Omega = \{q\}$ and P be the following Ω-open program $P = \{p(b) : -q(b),\ p(X),\ q(a)\}$.*
Then $\mathcal{O}_\Omega(P) = \{p(b) : -q(b),\ p(x),\ q(a)\}$ and
$M(P) = \{q(a)\} \cup \{p(\tilde{t}) \mid \tilde{t}$ is a ground term $\}$.
By proposition 5.8 $\mathcal{O}_\Omega(P)$ and $M(P)$ are Ω-models of P. However $\mathcal{O}_\Omega(P) \cap M(P) = \{q(a)\}$ is not an Ω-model of P.

The Ω-model intersection property does not hold because set theoretic operations do not adequately model the operations on conditional atoms. Namely, the information of an Ω-interpretation I_1 may be contained in I_2 without I_1 being a subset of I_2. In order to define the model-theoretic semantics for Ω-open programs as a unique (least) Ω-model, we then need a partial order $\sqsubseteq$ on Ω-interpretations which

allows to restore the model intersection property. $\sqsubseteq$ should model the meaning of Ω-interpretations, in such a way that $(\Im, \sqsubseteq)$ is a complete lattice and the greatest lower bound of a set of Ω-models is an Ω-model. As we will show in the following, this can be obtained by considering $\sqsubseteq$ as given in definition 5.10. According to the above mentioned property, there exists a least Ω-model. It is worth noting that such a least Ω-model is the standard least Herbrand model (proposition 5.21). Moreover note that, the most expressive Ω-model $\mathcal{O}_\Omega(P)$ is a non-minimal Ω-model. The following definitions extend those given in [Falaschi et al. 1989b] for the non compositional semantics of positive logic programs.

Definition 5.10 *Let I_1, I_2 be Ω-interpretations. We define*

- $I_1 \leq I_2$ *iff* $\forall c_1 \in I_1\ \exists c_2 \in I_2$ *such that* $c_2 \leq c_1$.
- $I_1 \sqsubseteq I_2$ *iff* $(I_1 \leq I_2)$ *and* $(I_2 \leq I_1$ *implies* $I_1 \subseteq I_2)$.

Proposition 5.11 *The relation $\leq$ is a preorder and the relation $\sqsubseteq$ is an ordering.*

Note that if $I_1 \subseteq I_2$, then $I_1 \sqsubseteq I_2$, since $I_1 \subseteq I_2$ implies $I_1 \leq I_2$. The following definitions and propositions will be used to define the model-theoretic semantics.

Definition 5.12 *Let I be an Ω-interpretation. We define $Min'(I) = \{c \in I \mid \forall c' \in I,\ c' \leq c \Rightarrow c' = c\}$ and $Min(I) = \widehat{Min'}(I)$.*

Example 5.13 *We show Min and Min' for the following Ω-interpretations I and J. Let*

$$
\begin{array}{lll}
I = & \{p(x),\ q(b),\ p(a),\ p(a) : -q(b) & \} \\
J = \{ & q(x) : -p(x), r(x) & \\
 & q(b) : -p(b) & \\
 & q(b) : -p(x) & \\
 & r(b) & \}
\end{array}
$$

Then
$Min'(I) = Min(I) = \{p(x),\ q(b)\}$,
$Min'(J) = \{r(b),\ q(x) : -p(x), r(x),\ q(b) : -p(x)\}$,
$Min(J) = J$.

Definition 5.14
Let Λ be a set of Ω-interpretations. We introduce the following notations.

- $\bigtriangledown\Lambda = \bigcup_{I \in \Lambda} I$
- $Min(\Lambda) = Min(\bigtriangledown\Lambda)$
- $\bigsqcup\Lambda = \hat{A}$ *where* $A = Min(\Lambda) \cup \bigtriangledown\{I \in \Lambda \mid Min(\Lambda) \subseteq I\}$)

It is worth noting that $\forall\, I$ Min(I) $\subseteq$ I (recall that I is u-closed) and $Min(\Lambda) = Min(\bigsqcup\Lambda)$.

Proposition 5.15
For any set Λ of Ω-interpretations there exists the least upper bound of Λ , $lub(\Lambda)$, and $lub(\Lambda) = \bigsqcup\Lambda$ holds.

Proposition 5.16 *The set of all the Ω-interpretations $\Im$ with the ordering $\sqsubseteq$ is a complete lattice. C_Ω is the top element and $\emptyset$ is the bottom element.*

The model-theoretic construction is possible only if Ω-interpretations can be viewed as representations of Herbrand interpretations. Notice that every Herbrand interpretation is an Ω-interpretation. The following proposition generalizes the standard intersection property of Herbrand models to the case of Ω-models.

Proposition 5.17 *Let* M *be a non-empty set of Ω-models of an Ω-open program P. Then glb(M) is an Ω-model of P.*

Corollary 5.18 *The set of all the Ω-models of a program P with the ordering $\sqsubseteq$ is a complete lattice.*

We are now in the position to formally define the model-theoretic semantics.

Definition 5.19 *Let P be a program. Its model-theoretic semantics is the greatest lower bound of the set of its models, i.e.,*
$M_\Omega(P) = glb(\{I \in \Im \mid I$ *is a Ω-model of* $P\})$.

Proposition 5.21 shows that the above defined model-theoretic semantics is the standard least Herbrand model. This fact justifies our choice of the ordering relation.

Proposition 5.20 *For any Ω-model I there exists a standard Herbrand model I' such that $I' \sqsubseteq I$.*

Proposition 5.21 *The least standard Herbrand model is the least Ω-model.*

6 S_Ω-models

We will now consider the relation between Ω-models (definition 5.5) and the S_Ω-models defined in [Bossi and Menegus 1991] on the same set of interpretations. Both the Ω-models and the S_Ω-models are intended to capture specific operational properties, from a model-theoretic point of view. However, S_Ω-models are based on an ad hoc notion of truth (S_Ω-truth) and the least S_Ω-model is exactly $\mathcal{F}_\Omega(P)$.

Conversely, Ω-models are based on the usual notion of truth in a Herbrand interpretation through the function $\mathcal{H}$. Moreover the least Ω-model is the usual least Herbrand model, while $\mathcal{F}_\Omega(P)$ is a non-minimal Ω-model.

Definition 6.1 *[Bossi and Menegus 1991]* (S_Ω-Truth) *Let Ω be a set of predicate symbols and I be an Ω-interpretation. Then*

(a) *An atom A is Ω-true in I iff $A \in I$.*

(b) *A definite clause $A :- B_1, \ldots, B_m$ is Ω-true in I iff*
$\forall B'_1, \ldots, B'_n$ *such that*
$B'_1 : -\tilde{L}_1, \ldots, B'_n : -\tilde{L}_n \in I \cup Id(\Omega)$
if $\exists \vartheta = mgu((B_1, \ldots, B_n), (B'_1, \ldots, B'_n))$
then $(A : -\tilde{L}_1, \ldots, \tilde{L}_n)\vartheta \in I$.

S_Ω-models are defined in the obvious way.

Proposition 6.2 *Every S_Ω-model is an Ω-model (according to definition 5.5).*

Proposition 6.3 *[Bossi and Menegus 1991] If Λ is a non-empty set of S_Ω-models of an Ω-open program P, then $\bigcap_{M \in \Lambda} M$ is an S_Ω-model of P.*

The previous proposition allows to define the model theoretic semantics $\mathcal{M}_{S_\Omega}(P)$ for a program P in terms of the S_Ω-models as follows.

Definition 6.4 *[Bossi and Menegus 1991] Let P be an Ω-open program and let S be the set of all the S_Ω-models of P. Then $\mathcal{M}_{S_\Omega}(P) = \bigcap_{M \in S} M$.*

Corollary 6.5 *Let Λ be a non-empty set of S_Ω-models of an Ω-open program P. Then $\bigcap_{M \in \Lambda} M$ is an Ω-model of P.*

By definition and by proposition 6.3, $\mathcal{M}_{S_\Omega}(P)$ is the least S_Ω-model in the lattice $(\Im, \sqsubseteq)$ (recall that $\Im$ is the set of all the Ω-interpretations). The following proposition shows that $\mathcal{M}_{S_\Omega}(P)$ is also the least S_Ω-model in the lattice $(\Im, \sqsubseteq)$.

Proposition 6.6 *Let P be a program and let S be the set of all the S_Ω-models of P. Then $\mathcal{M}_{S_\Omega}(P) = glb(S)$ (according to $\sqsubseteq$ ordering).*

The following theorem shows the equivalence of the fixpoint semantics (definition 4.4) and the model-theoretic semantics $\mathcal{M}_{S_\Omega}(P)$.

Theorem 6.7 *[Bossi and Menegus 1991] Let P be an Ω-open program. Then $\mathcal{F}_\Omega(P) = \mathcal{M}_{S_\Omega}(P)$.*

Corollary 6.8 *Let P be an Ω-open program. Then $\mathcal{F}_\Omega(P)$ is an Ω-model of P.*

It is worth noting that, since $\mathcal{O}_\Omega(P) = \mathcal{F}_\Omega(P) = \mathcal{M}_{S_\Omega}(P)$, theorem 2.9 shows that the model-theoretic semantics $\mathcal{M}_{S_\Omega}(P)$ is compositional w.r.t. Ω-union of programs when considering computed answer substitutions as observables. This result was already proved in [Bossi and Menegus 1991] for the $\mathcal{M}_{S_\Omega}(P)$ model. Finally note that, as shown by the following example, T_P^Ω is not monotonic (and therefore it is not continuous) on the complete lattice $(\Im, \sqsubseteq)$. However, proposition 6.10 ensures us that $\mathcal{F}_\Omega(P)$ is still the least fixpoint of T_P^Ω on $(\Im, \sqsubseteq)$.

Example 6.9 *Consider the program*
$P = \{r(b) \quad p(x) : -q(x)\}$.
Let $\Omega = \emptyset$, $I_1 = \{q(a), q(x)\}$ and $I_2 = \{r(b), q(x)\}$.
Then $I_1 \sqsubseteq I_2$ while $T_P^\Omega(I_1) = \{p(x), p(a), r(b)\} \not\sqsubseteq T_P^\Omega(I_2) = \{p(x), r(b)\}$.

Proposition 6.10 *$T_P^\Omega \uparrow \omega$ is the least fixpoint of T_P^Ω on the complete lattice $(\Im, \sqsubseteq)$.*

7 Related work and conclusions

The result of our semantic construction has several similarities with the proof-theoretic semantics defined in [Gaifman and Shapiro 1989a, Gaifman and Shapiro 1989b]. Our construction however is closer to the usual characterization of the semantics of logic programs. Namely we define a top-down operational and bottom-up fixpoint semantics, and, last but not least a model-theoretic semantics which allows us to obtain a declarative characterization of syntactically defined models. The semantics in [Gaifman and Shapiro 1989a] does not characterize computed answer substitutions, while the denotation defined by the fully abstract semantics in [Gaifman and Shapiro 1989b] is not a set of clauses (i.e. a program). The framework of [Gaifman and Shapiro 1989a, Gaifman and Shapiro 1989b] can be useful for defining a program equivalence notion, even if our more declarative (model-theoretic) characterization is even more adequate. Moreover, the presence of an operational or a fixpoint semantics makes our construction useful as a formal basis for program analysis. Another related paper is [Brogi et al. 1991], where Ω-open logic programs are called open theories. Open theories are provided with a model-theoretic semantics which is based on ideas very similar to those underlying our definition 5.3. [Brogi et al. 1991] however does not consider semantic definitions in the style of our $\mathcal{O}_\Omega(P)$ which gives a unique denotation to any open program.

Let us finally remark some interesting properties of the Ω-model $\mathcal{O}_\Omega(P)$.

- By means of a syntactic device, we obtain a unique representation for a possibly infinite set of Herbrand models when a unique representative Herbrand model does not exist. A similar device was used in [Dung and Kanchanasut 1989, Kanchanasut and Stuckey 1990, Gabbrielli et al. 1991] to characterize logic programs with negation.
- Operators, such as $\cup_\Omega$ are quite easy and natural to define on $\mathcal{O}_\Omega(P)$.
- $\mathcal{O}_\Omega(P)$ can be used for modular program analysis [Giacobazzi and Levi 1991] and for studying new equivalences of logic programs, based on computed answer substitutions, which are not considered in [Maher 1988].
- It is strongly related to *abduction* [Eshghi and Kowalski 1989]. If Ω is the set of abducible predicates, the abductive consequences of any goal G can be found by executing G in $\mathcal{O}_\Omega(P)$.
- The delayed evaluation of open predicates which is typical of $\mathcal{O}_\Omega(P)$ can easily be generalized to other logic languages, to achieve compositionality w.r.t the union of programs. In particular this matches quite naturally the semantics of CLP and concurrent constraint programs given in [Gabbrielli and Levi 1990, Gabbrielli and Levi 1991a].

References

[Apt 1988] K. R. Apt. Introduction to Logic Programming. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science*, volume B: Formal Models and Semantics. Elsevier, Amsterdam and The MIT Press, Cambridge, 1990.

[Bossi et al. 1991] A. Bossi, M. Gabbrielli, G. Levi, and M. C. Meo. Contributions to the Semantics of Open Logic Programs. Technical Report TR 17/91, Dipartimento di Informatica, Università di Pisa, 1991.

[Bossi and Menegus 1991] A. Bossi and M. Menegus. Una Semantica Composizionale per Programmi Logici Aperti. In P. Asirelli, editor, *Proc. Sixth Italian Conference on Logic Programming*, pages 95–109, 1991.

[Brogi et al. 1991] A. Brogi, E. Lamma, and P. Mello. Open Logic Theories. In P. Krueger L.-H. Eriksson and P. Shroeder-Heister, editors, *Proc. of the Second Workshop on Extensions to Logic Programming*, Lecture Notes in Artificial Intelligence. Springer-Verlag, Berlin, 1991.

[Dung and Kanchanasut 1989] Phan Minh Dung and K. Kanchanasut. A Fixpoint Approach to Declarative Semantics of Logic Programs. In E. Lusk and R. Overbeck, editors, *Proc. North American Conf. on Logic Programming'89*, pages 604–625. The MIT Press, Cambridge, Mass., 1989.

[Eshghi and Kowalski 1989] K. Eshghi and R. A. Kowalski. Abduction compared with Negation by Failure. In G. Levi and M. Martelli, editors, *Proc. Sixth Int'l Conf. on Logic Programming*, pages 234–254. The MIT Press, Cambridge, Mass., 1989.

[Falaschi et al. 1988] M. Falaschi, G. Levi, M. Martelli, and C. Palamidessi. A new Declarative Semantics for Logic Languages. In R. A. Kowalski and K. A. Bowen, editors, *Proc. Fifth Int'l Conf. on Logic Programming*, pages 993–1005. The MIT Press, Cambridge, Mass., 1988.

[Falaschi et al. 1989a] M. Falaschi, G. Levi, M. Martelli, and C. Palamidessi. Declarative Modeling of the Operational Behavior of Logic Languages. *Theoretical Computer Science*, 69(3):289-318, 1989.

[Falaschi et al. 1989b] M. Falaschi, G. Levi, M. Martelli, and C. Palamidessi. A Model-Theoretic Reconstruction of the Operational Semantics of Logic Programs. Technical Report TR 32/89, Dipartimento di Informatica, Università di Pisa, 1989. To appear in *Information and Computation*.

[Gabbrielli and Levi 1990] M. Gabbrielli and G. Levi. Unfolding and Fixpoint Semantics of Concurrent Constraint Programs. In H. Kirchner and W. Wechler, editors, *Proc. Second Int'l Conf. on Algebraic and Logic Programming*, volume 463 of *Lecture Notes in Computer Science*, pages 204–216. Springer-Verlag, Berlin, 1990. Extended version to appear in *Theoretical Computer Science*.

[Gabbrielli and Levi 1991a] M. Gabbrielli and G. Levi. Modeling Answer Constraints in Constraint Logic Programs. In K. Furukawa, editor, *Proc. Eighth Int'l Conf. on Logic Programming*, pages 238– 252. The MIT Press, Cambridge, Mass., 1991.

[Gabbrielli and Levi 1991b] M. Gabbrielli and G. Levi. On the Semantics of Logic Programs. In J. Leach Albert. B. Monien, and M. Rodriguez-Artalejo, editors, *Automata, Languages and Programming, 18th International Colloquium*, volume 510 of *Lecture Notes in Computer Science*, pages 1–19. Springer-Verlag, Berlin, 1991.

[Gabbrielli et al. 1991] M. Gabbrielli, G. Levi, and D. Turi. A Two Steps Semantics for Logic Programs with Negation. Technical report, Dipartimento di Informatica, Università di Pisa, 1991.

[Gaifman and Shapiro 1989a] H. Gaifman and E. Shapiro. Fully abstract compositional semantics for logic programs. In *Proc. Sixteenth Annual ACM Symp. on Principles of Programming Languages*, pages 134–142. ACM. 1989.

[Gaifman and Shapiro 1989b] H. Gaifman and E. Shapiro. Proof theory and semantics of logic programs. In *Proc. Fourth IEEE Symp. on Logic In Computer Science*, pages 50–62. IEEE Computer Society Press. 1989.

[Giacobazzi and Levi 1991] R. Giacobazzi and G. Levi. Compositional Abstract Interpretation of Constraint Logic Programs. Technical report, Dipartimento di Informatica. Università di Pisa. 1991.

[Kanchanasut and Stuckey 1990] K. Kanchanasut and P. Stuckey. Eliminating Negation from Normal Logic Programs. In H. Kirchner and W. Wechler, editors, *Proc. Second Int'l Conf. on Algebraic and Logic Programming*, volume 463 of *Lecture Notes in Computer Science*, pages 217–231. Springer-Verlag, Berlin. 1990.

[Lassez and Maher 1984] J.-L. Lassez and M. J. Maher. Closures and Fairness in the Semantics of Programming Logic. *Theoretical Computer Science*. 29:167–184, 1984.

[Levi 1988] G. Levi. Models. Unfolding Rules and Fixpoint Semantics. In R. A. Kowalski and K. A. Bowen, editors. *Proc. Fifth Int'l Conf. on Logic Programming*, pages 1649–1665. The MIT Press, Cambridge, Mass., 1988.

[Levi and Mancarella 1988] G. Levi and P. Mancarella. The Unfolding Semantics of Logic Programs. Technical Report TR-13/88. Dipartimento di Informatica. Università di Pisa. 1988.

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

[Lloyd and Shepherdson 1987] J. W. Lloyd and J. C. Shepherdson. Partial Evaluation in Logic Programming. Technical Report CS-87-09, Computer Science Department, University of Bristol, 1987. Revised version 1989, to appear in *Journal of Logic Programming*.

[Maher 1988] M. J. Maher. Equivalences of Logic Programs. In J. Minker, editor, *Foundations of Deductive Databases and Logic Programming*, pages 627–658. Morgan Kaufmann, Los Altos, Ca., 1988.

[Mancarella and Pedreschi 1988] P. Mancarella and D. Pedreschi. An Algebra of Logic Programs. In R. A. Kowalski and K. A. Bowen, editors, *Proc. Fifth Int'l Conf. on Logic Programming*, pages 1006–1023. The MIT Press, Cambridge, Mass., 1988.

[van Emden and Kowalski 1976] M. H. van Emden and R. A. Kowalski. The semantics of predicate logic as a programming language. *Journal of the ACM*, 23(4):733–742, 1976.