New Generation Computing, 11 (1992) 1-21
OHMSHA, LTD. and Springer-Verlag

Compositional Model-Theoretic Semantics for Logic Programs

Antonio BROGI
Dipartimento di Informatica,
Università di Pisa,
Corso Italia 40, 56125 Pisa, Italy.
Evelina LAMMA and Paola MELLO
DEIS,
Università di Bologna,
Viale Risorgimento 2, 40136 Bologna, Italy.

Received 27 May 1991
Revised manuscript received 12 March 1992

Abstract We present a compositional model-theoretic semantics for logic programs, where the composition of programs is modelled by the composition of the admissible Herbrand models of the programs. An Herbrand model is admissible if it is supported by the assumption of a set of hypotheses. On one hand, the hypotheses supporting a model correspond to an open interpretation of the program intended to capture possible compositions with other programs. On the other hand, admissible models provide a natural model-theory for a form of hypothetical reasoning, called abduction. The application of admissibel models to programs with negation is discussed.

Keywords: Logic Programming, Herbrand Models, Compositionality

§1 Introduction and Motivations

One of the most fascinating properties of using logic as a programming language is its semantics and declarativeness. Definite Horn clauses have simple model-theoretic, fixpoint and operational semantics, which have been proved to be all equivalent.[29] Several efforts have been devoted to broaden the application area of logic programming to cover different aspects of computing. In many areas of computer science, modularity has been recognised as a basic means for the incremental construction of programs. Programming-in-the-large, metalevel programming and object-oriented programming are only some examples of such applications.

From a theoretical point of view, the semantics of program composition operations represents a crucial issue in computational logic. The union of programs is the simplest composition operation between logic programs. Actually, every program can be viewed as the composition by union of all its clauses. Unfortunately, the standard model-theoretic semantics of logic programming does not extend in a straightforward way to the composition by union of programs. The minimal Herbrand model is usually taken as the semantic counterpart of a definite logic program.[29] A propositional example suffices to show that the minimal Herbrand model of the nuion of two programs cannot be obtained in a compositional way from the minimal Herbrand models of the two separate programs. Indeed, the two programs $p \leftarrow q$ and $p \leftarrow r$ have the same (empty) minimal Herbrand model. However, if these programs are composed with the program $r \leftarrow$ we obtain two programs which have different minimal Herbrand models ($\{r\}$ and $\{p, r\}$, respectively).

Some works have been devoted to provide semantic frameworks to cope with program composition, by defining concepts such as algebrae of programs.[17,18,21] In these papers, the semantics of each program P is based on the immediate consequence operator (T_P),[2] and the union of programs is mapped onto a transformation of the semantics of the programs. Although the immediate consequence semantics for the union of programs indirectly entails a model-theoretic characterisation, we are interested in providing a direct model-theoretic characterisation, that is to define the semantics of the composition of two programs in terms of a composition of their models.

The reason why the minimal Herbrand model semantics does not properly cope with program composition derives from the adoption of the Closed World Assumption.[26] According to the Closed World Assumption, and the corresponding completion semantics,[9] a logic program is interpreted as a complete knowledge specification. Such an interpretation does not reflect the implicit assumption underlying program composition, that is that a program is conceived as an incomplete chunk of knowledge to be possibly completed with other knowledge. As a consequence, each program cannot be simply denoted by its minimal Herbrand model, where only provable formulae are considered. Also non-minimal Herbrand models of a program must be considered, including formulae not provable in the program, but which can possibly become provable after some program composition.

Under this perspective, a compositional semantics of logic programs has been defined in Ref. 7). Each program is denoted by the set of all its Herbrand models, and the minimal Herbrand model of the union of two programs is proved to be the minimal Herbrand model which is a model of both programs.

In this paper, we refine such a characterisation by considering only a subset of the Herbrand models of the program, called the *admissible* Herbrand models. An Herbrand model is admissible if it is supported by the assumption of a set of hypotheses. On one hand, the hypotheses supporting a model correspond to an open interpretation of the program intended to capture possible compositions with other programs. On the other hand, admissible models provide a natural model-theory for a form of hypothetical reasoning, called abduction.[4,8,23)]

The notion of *open* programs has been originally introduced in Ref. 6) to denote incomplete chunks of knowledge, which can be dynamically composed together. Admissible models provide a semantical counterpart of open programs, and have been fruitfully applied to model different forms of program composition, including fine-grained ones such as the composition of logic modules with import/export declarations.

Admissible models can also be applied for characterising the class of general logic programs, that is programs containing negation in the bodies of clauses. In Section 5, we describe this application by discussing the generality and the advantages of the approach, and by summarising the main results. A complete treatment of the semantics of general logic programs through admissible models is outside the scope of this paper, and is reported in Ref. 5). To give an example of the results contained in Ref. 5), we show the relation between admissible and stable models, as a case study.

The plan of the paper follows. First, the condition of admissibility on Herbrand models is formally introduced. Compositionality is achieved by mapping the composition of programs onto the composition of the admissible models of the programs. Then we illustrate the relations between admissible Herbrand models and abduction, and between admissible Herbrand models and negation. Finally, related work is discussed in some detail, and some conclusions are drawn. To make the paper more good reading, some proofs are reported in the appendix.

§2 Admissible Herbrand Models

The condition of *admissibility* on the Herbrand models of a program restricts the set of models to work with when handling the composition of programs. Roughly, an Herbrand model is considered *admissible* if it corresponds to the assumption of a set of hypotheses, where each of the hypotheses occurs in the premise part (body) of some clause of the program.

Before introducing the formal definition of admissible Herbrand models, let us consider a simple example.

Example 2.1
Consider the following program P:

$$p \leftarrow q$$
$$r \leftarrow$$

The minimal Herbrand model of P (denoted by $\mathcal{M}_P$) is $\{r\}$. The refutation of the goal $\leftarrow p$ fails on the sub-goal $\leftarrow q$. In alternative, q may be interpreted as an admissible hypothesis which may become true (i.e. provable) after a composition with some other program (trivially with the program $q \leftarrow$). Therefore the (non-minimal) Herbrand model $\{p, q, r\}$ can be considered admissible for P under the hypothesis $\{q\}$. □

The formal definition of admissible hypotheses for a program follows.

Definition 2.2 (Admissible Hypotheses)
Let P be a program. A subset H of the Herbrand base of P is an admissible set of hypotheses for P if and only if for all $h \in H$ there exists a ground instance $A \leftarrow B$ of a clause in P such that $h \in B$. □

For each program P, sets of hypotheses occurring in the premise part of the clauses of P are considered admissible to deal with possible program compositions. For instance, in the previous example the sets of admissible hypotheses for the program P are $\{\}$ and $\{q\}$.

The Herbrand models of a program which are admissible are those which are supported by some set of admissible hypotheses. Roughly speaking, an Herbrand model M of a program P is admissible if there exists an admissible set of hypotheses H such that all the atoms in M are logical consequences of $P \cup H$. Let us introduce a formal definition of admissible Herbrand models.

Definition 2.3 (Admissible Herbrand Model)
Let P be a program, M be an Herbrand model of P, and let H be an admissible set of hypotheses for P. M is an admissible Herbrand model of P under the hypotheses H if and only if $M = \mathcal{M}_{P \cup H}$. □

Notice that $\mathcal{M}_{P \cup H}$ denotes the minimal Herbrand model of $P \cup H$, where $P \cup H$ stands for $P \cup \{h_1 \leftarrow\} \cup ... \cup \{h_n \leftarrow\}$ if $H = \{h_1, ..., h_n\}$.

There are other alternative equivalent ways of formulating the condition $M = \mathcal{M}_{P \cup H}$ of Definition 2.3. For instance, M can be expressed in terms of the immediate consequence operator[29]:

$$T_P(I) = \{a \mid a \leftarrow b_1, ..., b_n \in \mathit{ground}(P) \wedge \{b_1, ..., b_n\} \subseteq I\} \cup I$$

by defining

$$M = T_P \uparrow \omega(H)$$

that is M is the least fixpoint of the T_P operator starting from the set of hypotheses H.

The fact that the admissible models of a program are a subset of the Herbrand models of the program is best illustrated by defining M as an intersection of Herbrand models of the program:

$$\begin{aligned} M &= \bigcap \{M' \mid M' \models P \wedge H \subseteq M'\} \\ &= \bigcap \{M' \mid M' \models P \wedge M' \models H\} \end{aligned}$$

Notice that if the admissibility condition on the hypotheses would be relaxed, we would obtain the set of all Herbrand models of a program (as in Ref. 7).

Remark 2.4
Definition 2.3 allows one to determine the admissible models among the Herbrand models of a program. Given a program P, for each admissible set of hypotheses H there is a unique Herbrand model of P which is admissible under H (by Definition 2.3). The converse is not necessarily true: an Herbrand model of a program can be admissible under possibly several different sets of hypotheses. Consider the following program P:

$$p \leftarrow q$$
$$q \leftarrow p$$

The Herbrand models of P are $\{\}$ and $\{p, q\}$. Trivially, $\{\}$ is admissible under the empty set of hypotheses, while the model $\{p, q\}$ is admissible under three different sets of hypotheses: $\{p\}$, $\{q\}$ and $\{p, q\}$. □

Notation
In the rest of the paper, $AHM(P)$ will stand for the set of admissible Herbrand models of the program P. The notations M *if* H and M^H will be interchangeably used as shorthands for "M is an admissible Herbrand model under the hypotheses H".

In the standard model theory of logic programming, a program is denoted by its minimal Herbrand model. Notably, such a denotation also characterises the operational behaviour of a program. A well known result[29] states the equivalence of the operational and of the model-theoretic semantics of logic programming. The success set of a program P ($SS(P)$), that is the set of ground atomic formulae which can be derived in P, coincides with the minimal Herbrand model of P, i.e. $\mathcal{M}_P = SS(P)$.

On the other hand, the minimal Herbrand model does not contain enough information on the program to model possible compositions with other programs. To address compositionality issues, a program is here denoted by a set of Herbrand models, that is:

$$[\![P]\!] = AHM(P).$$

Roughly, denoting a program with a set of models rather than with a single model corresponds to interpreting the program as an incomplete rather than a complete chunk of knowledge. In other words, a program is viewed as "open" rather than "closed" with respect to possible additions of new knowledge (via compositions with other programs). The set of admissible models of a program characterises the set of possible behaviours of the program in presence of different possible extensions. A model M for a program P which is admissible under the hypotheses H is the minimal Herbrand model of the program obtained extending P with H. Thanks to the equivalence between the operational and model-theoretic semantics of logic programs,[29] M also coincides with the success set of $P \cup H$, that is:

$$M = \mathcal{M}_{P \cup H} = SS(P \cup H).$$

It is worth observing that the admissible models of a program also characterise the semantics of the program viewed as "closed" with respect to possible compositions with other programs. In other words, it is possible to identify the minimal Herbrand model of a program among the admissible models of the program. Actually, the minimal Herbrand model is admissible under the empty set of hypotheses and can be obtained by intersecting the admissible models of the program.

Proposition 2.5
For any program P:

$$\mathcal{M}_P = \bigcap \{M \mid M \in \llbracket P \rrbracket\}$$

Proof
Immediate since $\mathcal{M}_P \in \llbracket P \rrbracket$ and since Herbrand models are closed under intersection.[29] □

§3 Composition of Logic Programs

We now turn our attention to compositionality issues. We show how to determine the semantics of the union of two programs ($\llbracket P \cup Q \rrbracket$) from the semantics of the two separate programs ($\llbracket P \rrbracket$ and $\llbracket Q \rrbracket$). This corresponds to proving that the following diagram commutes

$$\begin{tikzcd}
P, Q \arrow[r, "\llbracket\,\rrbracket"] \arrow[d, "\cup"'] & \llbracket P \rrbracket, \llbracket Q \rrbracket \arrow[d, "\bullet"] \\
P \cup Q \arrow[r, "\llbracket\,\rrbracket"'] & \llbracket P \cup Q \rrbracket
\end{tikzcd}$$

Fig. 1

where "$\bullet$" is the composition operation on the semantics of programs corresponding to the composition operation $\cup$ on the syntax of programs.

Since each program is denoted by the set of its admissible Herbrand models, the composition of the semantics of the programs corresponds to a function working on sets of admissible models. A composition operator τ is introduced, which works on a set of (admissible) Herbrand models and maps

Herbrand interpretations into Herbrand interpretations.

Definition 3.1 (Composition Operator τ)
Let S be a set of admissible Herbrand models, the composition operator τ mapping Herbrand interpretations into Herbrand interpretations is defined as follows:

$$\tau_S(I) = (\bigcup \{M \mid M^H \in S \textit{ and } H \subseteq I\}) \cup I \quad \square$$

The operator τ is monotonic and continuous (see Appendix, Proposition 7.1), and its powers are defined as usual[1]:

$$\begin{aligned} \tau \uparrow 0(I) &= I \\ \tau \uparrow (n+1)\,(I) &= \tau(\tau \uparrow n(I)) \\ \tau \uparrow \omega(I) &= \bigcup_{n<w} \tau \uparrow n(I) \end{aligned}$$

The definition of τ_S parallels the definition of the immediate consequence operator T_P. Both T_P and τ_S map Herbrand interpretations into Herbrand interpretations. While T_P is defined in terms of programs, τ_S is defined in terms of admissible Herbrand models. A program P is a set of clauses, that is of implications of the form A *if* B. Given an interpretation I, $T_P(I)$ yields all the atoms A whose premise B in P is true in I. Similarly, a set of admissible models S can be viewed as a set of implications of the form: M *if* H. Given an interpretation I, $\tau_S(I)$ yields the union of the models M whose premise H (hypothesis) is true in I.

Proposition 3.2
Given two programs P and Q, let H be an admissible set of hypotheses for $P \cup Q$. M is an admissible Herbrand model of $P \cup Q$ under the set of hypotheses H if and only if $M = \tau_{[\![P]\!] \cup [\![Q]\!]} \uparrow \omega(H)$.

Proof
See Appendix. $\square$

Notice that the condition "let H be an admissible set of hypotheses for $P \cup Q$" in the former Proposition 3.2 can be entirely expressed in terms of the semantics of the programs. Given two programs P and Q, the set K of all the admissible hypotheses for $P \cup Q$ is defined as follows:

$$K = \bigcup \{H \mid M^H \in [\![P]\!] \cup [\![Q]\!]\}.$$

Then, H is an admissible set of hypotheses for $P \cup Q$ is and only if $H \subseteq K$.

The $\bullet$ composition operation is defined as follows.

Definition 3.3 (Composition $\bullet$)
Given two programs P and Q, let K be the set of all the admissible hypotheses for $P \cup Q$:

$$[\![P]\!] \bullet [\![Q]\!] = \{M \mid M = \tau_{[P] \cup [Q]} \uparrow \omega(H),\ H \subseteq K\}. \quad \square$$

Finally, the following proposition states the compositionality result of the admissible model semantics.

Proposition 3.4
Given two programs P and Q:

$$[\![P \cup Q]\!] = [\![P]\!] \bullet [\![Q]\!].$$

Proof
Immediate by Proposition 3.2. □

Let us consider a simple example of program composition.

Example 3.5
Consider the following two programs.

P	Q
$p \leftarrow$	$r \leftarrow$
$q \leftarrow r$	$s \leftarrow t$

The sets of admissible Herbrand models of P and Q are, respectively:

$AHM(P)$		$AHM(Q)$	
$\{p\}$	*if* $\{\}$	$\{r\}$	*if* $\{\}$
$\{p, q, r\}$	*if* $\{r\}$	$\{r, s, t\}$	*if* $\{t\}$

To determine the set of admissible models of $P \cup Q$, we can apply the τ operator. Let us consider the empty set of hypotheses. We obtain (let $S = [\![P]\!] \cup [\![Q]\!]$):

$$\tau_S \uparrow 0(\{\}) = \{\}$$
$$\tau_S \uparrow 1(\{\}) = \{p, r\}$$
$$\tau_S \uparrow 2(\{\}) = \{p, q, r\}$$
$$\tau_S \uparrow 3(\{\}) = \{p, q, r\}$$

Proposition 3.2 states that $\{p, q, r\}$ is an admissible Herbrand model of $P \cup Q$ under the hypotheses $\{\}$. The other admissible Herbrand models are derived analogously. For instance, for the set $\{t\}$ of hypotheses we have:

$$\tau_S \uparrow 0(\{t\}) = \{t\}$$
$$\tau_S \uparrow 1(\{t\}) = \{p, r, s, t\}$$
$$\tau_S \uparrow 2(\{t\}) = \{p, q, r, s, t\}$$
$$\tau_S \uparrow 3(\{t\}) = \{p, q, r, s, t\}$$

Then $\{p, q, r, s, t\}$ if $\{t\}$ is an admissible Herbrand model of $P \cup Q$. □

In Ref. 12) the authors have studied the problem of compositionality in the semantics of logic programs, by adopting the definition that "a semantics is compositional if equivalent program parts exhibit equal behaviours in all contexts". The denotation of a program induces an equivalence relation on programs, which states that two programs are equivalent if and only if they have the same denotation. Formally:

$$P \equiv Q \Leftrightarrow [\![P]\!] = [\![Q]\!]$$

Let *Ob* be a function which associates with every program *P* an object *Ob*(*P*) which is the observable behaviour of *P*.

Definition 3.6 (Compositionality)
Let *P*, *Q*, *R* be programs. An equivalence relation $\equiv$ is compositional if:

(1) $P \equiv Q \Rightarrow Ob(P) = Ob(Q)$
(2) $P \equiv Q \Rightarrow P \cup R \equiv Q \cup R$ □

We consider $[\![P]\!] = AHM(P)$ and the corresponding equivalence relation:

$$P \equiv_A Q \Leftrightarrow AHM(P) = AHM(Q)$$

which states that two programs are equivalent if and only if they have the same admissible Herbrand models. It is easy to show that the admissible Herbrand model semantics satisfies the former definition of compositionality. According to the standard semantics of logic programming,[1,16,29)] we consider the ground atoms which are derivable from a program *P* (i.e. the success set of *P*) as the set of observables *Ob*(*P*) of *P*.

Proposition 3.7
The equivalence relation $\equiv_A$ is compositional.

Proof
Suppose that $P \equiv_A Q$.

(1) $Ob(P)$
$=$ $\{\text{since } Ob(P) = SS(P) = \mathcal{M}_P\}$
$\mathcal{M}_P$
$=$ $\{\text{by Proposition 2.5}\}$
$\bigcap \{M \mid M \in [\![P]\!]\}$
$=$ $\{\text{since } P \equiv_A Q\}$
$\bigcap \{M \mid M \in [\![Q]\!]\}$
$=$ $\{\text{by Proposition 2.5}\}$
$\mathcal{M}_Q$
$=$ $\{\text{since } Ob(Q) = SS(Q) = \mathcal{M}_Q\}$
$Ob(Q)$

(2) $[\![P \cup R]\!] = [\![P]\!] \bullet [\![R]\!] = [\![Q]\!] \bullet [\![R]\!] = [\![Q \cup R]\!]$ □

§4 Admissible Models and Abduction

So far, admissible Herbrand models have been introduced to cope with program composition issues, and the hypotheses supporting a model intend to denote the possible addition of new knowledge, viewed as a program composition. The admissibility of an Herbrand model relies on a set of hypotheses, which when assumed to hold support the model itself. The explicit relation between Herbrand models and the hypotheses supporting them enables to model a form of abduction.[8,23]

A simplified notion of abductive reasoning can be formulated as follows[15,22]:

> Given a program P and a possible conclusion C, an abductive explanation of C is a set of sentences Δ such that: $P \cup \Delta \models C$.

Let us consider a simple example showing how abduction is a natural procedure for performing fault diagnosis. Consider the following example (taken from Ref. 15)) describing some of the causes of bicycle faults.

wobbly_wheel ← *flat_tyre*
wobbly_wheel ← *broken_spokes*
flat_tyre ← *leaky_valve*

The minimal Herbrand model of the former program is empty. Thus, for instance, the sentence *wobbly_wheel* cannot be derived from the program. However, several abductive explanations of *wobbly_wheel* can be determined by reasoning backward and extending the standard deduction mechanism of logic programming. The goal ← *wobbly_wheel* can be proved by assuming any of the (normally) failing sub-goals

← *flat_tyre*
← *broken_spokes*
← *leaky_valve*

to hold as hypotheses. Abduction provides a simple way of determining answers possibly conditioned by the assumption of some hypotheses. In this example, the answers to the query ← *wobbly_wheel* include:

yes if *flat_tyre*
yes if *broken_spokes*
yes if *leaky_valve*

It is easy to observe the strict correspondence between such conditional (abductive) answers and admissible Herbrand models. For instance, the conditional answer

yes if *flat_tyre*

corresponds to the admissible model:

$\{$*wobbly_wheel*, *flat_tyre*$\}$ if $\{$*flat_tyre*$\}$

The strict relation between abduction in logic programming and admissible Herbrand models best shows the significance of the admissiblilty condition imposed on Herbrand models. In general, only some predicates of an abductive theory are designed as abducible.[14] The declaration of the set of abducible predicates can be modelled by imposing a further constraint on the set of admissible hypotheses of a program. Let *abducible(P)* be the set of abducible predicates in a program P, then the constraint $p(t) \in H \Rightarrow p \in abducible(P)$ is added to the Definition 2.2 of the set of admissible hypotheses of a program.

In this section, we have referred to a simplified notion of abductive reasoning without taking into account integrity constraints, which play a crucial role in that setting. Actually, the set of possible abductive explanations Δ for a conclusion C with respect to a program P is restricted by a set of integrity constrains IC which Δ must satisfy.[15] A discussion of the modelisation of abductive reasoning with integrity constraints is outside the scope of this paper (see e.g. Ref. 14)). The interested reader may refer to Ref. 4) where a compositional model-theoretic semantics for abduction in logic programming has been defined in terms of admissible Herbrand models.

§5 Admissible Models and Negation

Admissible models can be applied for characterising the class of general logic programs, that is programs containing negation in the bodies of clauses. In this section, we try to describe this application by discussing the generality and the advantages of the approach, and by summarising the main results. A complete treatment of the semantics of general logic programs through admissible models is outside the scope of this paper, and is reported in Ref. 5).

Unfortunately, there is no common agreement on what the intended meaning of a general logic program is, as shown by the number of different semantics which have been proposed for general programs. Among others, we mention:

- Well-founded semantics[30]
- Stable model semantics[13] and its extensions such as partial stable models,[27] P-stable models,[28] and 3-valued stable models[24]
- Preferential semantics[10]
- Stationary semantics.[25]

Although some relations between these semantics are intuitively clear, formal comparisons are not easy to be drawn mainly because of the different constructions which are used in the definitions. Moreover, as far as compositionality problems are concerned, they are not addressed in the above mentioned approaches. It is not clear in general how to determine the models of a program with negation from the models of its components. The following example suffices to show that, for instance, the stable models of the union of two programs cannot be obtained in a compositional way from the stable models of the two separate programs. Indeed, the two programs $p \leftarrow$ *not* q and $p \leftarrow$ have the same stable model (i.e. $\{p\}$). However, if these programs are composed with the program $q \leftarrow$ we obtain two programs which have different stable models ($\{q\}$ and $\{p, q\}$, respectively).

In Ref. 5) we have studied the semantics of general programs in terms of admissible Herbrand models. According to Refs. 10), 11) and 25), general programs are treated as positive programs by considering each negative literal of the kind "*not A*" as a new positive literal "*not_A*". Each program can be thus denoted by the set of its admissible Herbrand models. The core of this study consists of relating admissible models with other semantics which have been proposed for general programs.

As we have discussed in Section 3, the intended meaning of a positive logic program when considered as "closed" is its minimal Herbrand model.[29] We have also shown how the minimal Herbrand model of a program can be identified among the admissible models of the program itself (see Proposition 2. 5). This corresponds to defining a function ϕ which given the denotation of a program as open (w.r.t. possible compositions with other programs) yields the minimal Herbrand model of the program.

$$\phi\colon AHM(P) \rightarrow \mathcal{M}_P$$

The situation is obviously different when considering the class of general logic programs, as there is no agreement on what the intended meaning of a programs is. Following this remark, in Ref. 5) we have shown the correspondence between admissible models and several different semantics for general programs. We have considered a number of different semantics: well-founded models,[30] stable models,[13] 3-valued stable models,[24] preferential semantics[10] and stationary semantics.[25] For each semantics S, we have defined a suitable function ϕ_S which given the admissible models of a program P yields the models $S(P)$ of the semantics S.

$$\phi_S\colon AHM(P) \rightarrow S(P)$$

A first advantage of this approach is that different semantics for general programs are defined in terms of the Herbrand models of programs, rather than in terms of the syntax of programs. This actually simplifies in many cases the formulation of the various semantics. For example, the definition of models given through syntactic transformations over programs is reformulated in terms of simple properties on Herbrand models of programs.

The main advantage of the approach is the possibility of drawing comparisons between different semantics for negation in logic programming. Equivalencies between different model-theoretic semantics can be proved as well as differences can be formally clarified. Notably, these comparisons are drawn in terms of the functions ψ_S which are used to define the various semantics through properties on the Herbrand models of the program. In other words, to compare two different semantics $S1$ and $S2$, we simply compare the corresponding functions ψ_{S1} and ψ_{S2}. For example, in Ref. 5) we have shown the equivalence between complete scenaria by Dung[10] and stationary expansions by Przymusinski.[25]

Another interesting perspective of the approach concerns compositionality issues. As general programs are denoted by their admissible models, the methodology defined in this paper for positive programs applies to general programs as well. The admissible models of the union of two general programs can be determined by the admissible models of the separate programs.

This property induces interesting results on the equivalence of general programs. Actually, it is not clear when two general programs should be considered equivalent. The compositionality of admissible model semantics together with the correspondences ψ_S do induce an interesting equivalence relation on general programs. Programs which have the same admissible models also have the same stable models, the same preferential models, three-valued models, and so on for all the semantics S for which a ψ_S is defined. Such a notion of equivalence—as any compositional notion of equivalence (see Definition 3.6)—supports ways of reasoning on programs and to transform them. For example, a part P of a program can be replaced by a syntactically different part Q without changing the intended semantics of the whole program, provided that P and Q are equivalent.

To give an example of the results contained in Ref. 5), we now show the relation between admissible and stable models, as a case study.

5.1 Stable Models and Admissible Models

Let us first give some notations which are introduced to treat general programs as positive programs according to Refs. 10), 11) and 25).

Notations

Let P be a general logic program and B_P the Herbrand base of P. The positive version P' of P is obtained by replacing each negative literal *not* A in P by a new literal not_A. The Herbrand base of P' is obtained by extending B_P with $not\_B_P = \{not\_A \mid A \in B_P\}$. With abuse of notation, we will not distinguish any further between P and its positive version P', and we will denote directly by P its positive version. An interpretation I for a program P is any subset of $B_P \cup not\_B_P$. We denote by I^+ and by I^- the positive and the negative part of I, respectively. Formally: $I^+ = I \cap B_P$ and $I^- = I \cap not\_B_P$. Given a set of positive atoms $I \subseteq B_P$, the complement $\overline{I}$ of I w.r.t. B_P is the set of negative atoms which do not occur positively in I, i.e.: $\overline{I} = \{not\_A \mid A \in B_P \wedge A \notin I\}$. Given a program P and an interpretation $I \subseteq B_P \cup not\_B_P$: I is *inconsistent* if $\exists A$: $A \in I \wedge not\_A \in I$, it is *consistent* otherwise. I is *total* if $\forall A \in B_P$: A

$\in I \vee not\_A \in I$.

The definition of stable models was introduced by Gelfond and Lifschitz[13)] by means of a syntactic transformation of programs. It can be equivalently stated in terms of the least fixed point of the immediate consequence operator as follows (see Refs. 10) and 11)).

Definition 5.1 (Stable Model)
Let P be a general logic program.
$M \subseteq B_P$ is a stable model for P iff $M \cup \overline{M} = T_{P \cup \overline{M}} \uparrow \omega$ □

Example 5.2
Let us consider the program P:

$$p \leftarrow not\_q$$

The only stable model of P is $M = \{p\}$. In fact $\overline{M} = \{not\_q\}$ and $M \cup \overline{M} = T_{P \cup \overline{M}} \uparrow \omega = \{p, not\_q\}$. □

In Ref. 5), the condition of admissibility on hypotheses for a program is a relaxation of that given in Definition 2.2: any subset of the Herbrand base of a general program P (i.e. $B_P \cup not\_B_P$) is an admissible set of hypotheses.

Each general program is associated with the set of its admissible models. We now show how to identify the stable models of a program P among the admissible models of P. Put another way, this corresponds to defining a function ψ_{SM} such that

$$\psi_{SM}([\![P]\!]) = Stable\ Models(P)$$

for any general program P. (Notice that for those programs which do not have any stable model, the function ψ_{SM} will give the empty set as result.)

We first state the existing correspondence between admissible and stable models of a program.

Proposition 5.3
Let P be a general logic program.
$M \subseteq B_P$ is a stable model for P iff N is an admissible Herbrand model for P under the hypotheses H such that:

(1) $H^+ = \phi$
(2) N is consistent and total
(3) $N^+ = M$.

Proof
($\Rightarrow$) By definition of stable model $M \cup \overline{M} = T_{P \cup \overline{M}} \uparrow \omega$. Notice that: $M = (T_{P \cup \overline{M}} \uparrow \omega)^+$. Consider now the Herbrand model N for P which is admissible under hypotheses $\overline{M}$ (it always exists by definition of admissible model). Obviously $(\overline{M})^+ = \phi$ since $M \subseteq B_P$. Moreover: $N = T_{P \cup \overline{M}} \uparrow \omega = M \cup \overline{M}$.

We observe that N is consistent and total since $N = M \cup \overline{M}$. Finally $N^+ = M$.

($\Leftarrow$) Notice that by definition of admissible model $H = N^-$. Since N is consistent and total: $N = N^+ \cup \overline{N^+}$. Then since $N^+ = M$: $M \cup \overline{M} = N^+ \cup \overline{N^+} = N = T_{P \cup \overline{M}} \uparrow \omega$. □

Example 5.4
Consider again the program P of Example 5.2:

$$p \leftarrow not\_q$$

To determine the stable models of P, we have to look at the Herbrand models of P which are admissible under a (possibly empty) set of negative hypotheses (see Proposition 5.3, condition (1)). Such admissible models are:

$$\begin{array}{lll}
\{\} & \textit{if} & \{\} \\
\{not\_p\} & \textit{if} & \{not\_p\} \\
\{p,\ not\_q\} & \textit{if} & \{not\_q\} \\
\{p,\ not\_p,\ not\_q\} & \textit{if} & \{not\_p,\ not\_q\}
\end{array}$$

We observe that only the third model satisfies all the conditions required by Proposition 5.3, as the first two models are not total and the fourth one is inconsistent. Therefore the only stable model for P is $\{p\} = \{p,\ not\_q\}^+$. □

The definition of the function ψ_{SM} directly follows by Proposition 5.3.

Definition 5.5 (ψ_{SM})
Let P be a general program.

$$\begin{aligned}
\psi_{SM}([\![P]\!]) = \{M \mid\ & N \in [\![P]\!] \textit{ admissible under } H \\
& \wedge H^+ = \phi \\
& \wedge N \textit{ is consistent and total} \\
& \wedge N^+ = M\}. \quad \square
\end{aligned}$$

§6 Related Work

Admissible Herbrand models have been exploited in Ref. 6) to model the representation of incomplete, possibly evolving, knowledge with logic. On one hand, evolving knowledge can be modelled by suitable metalevel operators for the dynamic composition of logic programs (see Refs. 7), 19) and 21)). On the other hand, the Open World Assumption adequately models the incompleteness of knowledge in a logic program. In this perspective, the notion of *open* program is introduced in Ref. 6) to denote an incomplete chunk of knowledge, along with two metalevel operators on programs: union and close. The former supports the composition of (possible open) programs, while the latter allows the enforcement of the Closed World Assumption on programs. Admissible Herbrand models provide a semantical counterpart for open programs, whereas union and close operators are mapped onto corresponding functions on admissible models. In a knowledge representation perspective, it is shown that these two metalevel operators suffice to reconstruct a number of policies for structuring logic programs, including modules with import/export declarations, forms of hypothetical reasoning[19] and contextual logic programming.[20].

As already mentioned, the sematics presented here is strongly related to that proposed in Ref. 7). They denote each program with the set of all its Herbrand models, and prove that given two programs P, Q:

$$M \models P \cup Q \Leftrightarrow M \models P \wedge M \models Q$$

As a consequence, the minimal Herbrand model of the union of two programs turns out to be the minimal Herbrand model which is a model for both programs. Since different programs may have different vocabularies, to cope with the composition of programs, the Herbrand base of each program is determined by the union of the sets of function and predicate symbols of all the programs being considered. More generally, the existence of a set $\langle F, \Pi \rangle$ of function and predicate symbols including all the function and predicate symbols of the programs being considered is assumed. Our semantics refines the semantics of Ref. 7) in two directions. First, each program is denoted by a subset of—instead than by all—the Herbrand models of a program. Second, the assumption on the set of predicate symbols of a program to be considered is relaxed. In the admissible Herbrand model semantics, to determine the Herbrand base of a program, there is no need for considering predicate symbols not occurring in it. A further major advantage of considering the admissible Herbrand models with respect to Ref. 7) is to provide a unified semantic framework for modelling program composition, abduction and negation.

Kakas and Mancarella[14] have defined a semantics for abduction by characterising each abductive program with a set of models, called generalized stable models (obtaind as suitable extensions of stable models[13]). The definition of generalized stable models is motivated by the view that abducibles represent basic possible beliefs, and Kakas and Mancarella show a close connection between their semantics and autoepistemic logic. A direct correspondence between admissible Herbrand models and generalized stable models can be easily defined. The admissible Herbrand models of a program correspond to the pre-generalized stable models of the program, where integrity constaints are not dealt with. Generalized stable models are obtained by considering only those pre-generalized models which satisfy the integrity constraints, as *strongly* admissible models are derived the same way in Ref. 4).

The problem of modelling the composition by union of programs has also been studied in Ref. 3). The main contribution of the paper is to study a notion of observable behaviour of programs richer than the standard notion of success set defined in Ref. 29). To capture this richer notion of behaviour, programs are denoted by programs (obtained through unfolding) rather than by

Herbrand models (as in Ref. 29) and here).

§7 Conclusions

The major result of the paper is the definition of a compositional model-theoretic semantics of logic programs. Rather than introducing *ad hoc* models to capture the composition by union of programs, the standard Herbrand model semantics is retained. Moreover, admissible models support a uniform treatment of other mechanisms, such as abduction and modules with import/export declarations. Most interesting, admissible models have been applied to the class of logic programs with negation.

Acknowledgements

We would like to thank Paolo Mancarella who has contributed to the application of admissible models both to abduction and negation. Thanks also to Dino Pedreschi, Franco Turini, Phan Minh Dung, Michael Gelfond, Tony Kakas and to the anonymous referees for pertinent remarks and suggestions. This work has been partially supported by the *C. N. R. Progetto Finalizzato Sistemi Informatici e Calcolo Parallelo*, under grants 92. 01564. PF69 and 92. 01606. PF69.

References

1) Apt, K. R., "Logic Programming," in *Handbook of Theoretical Computer Science* (J. van Leeuwen, ed.), Vol. B, Elsevier, pp. 493-574, 1990.
2) Apt, K. R. and van Emden, M. H., "Contributions to the Theory of Logic Programming," *Journal of the ACM*, *29*, 3, pp. 841-862, 1982.
3) Bossi, A., Gabbrielli, M., Levi, G., and Meo, M. C., "Contribution to the Semantics of Open Logic Programs," in *Proceedings of FGCS92* (ICOT, ed.), pp. 570-580, Ohmsha, 1992.
4) Brogi, A., Lamma, E., Mancarella, P., and Mello, P., "Abductive Reasoning in a Multi-Theory Framework," in *Trends in Artificial Intelligence* (S. Gaglio, ed.), *Proceedings of the Second Congress of the Italian Association for Artificial Intelligence, Lecture Notes in Artificial Intelligences*, *549*, Palermo, Springer-Verlag, pp. 137-146, 1991.
5) Brogi, A., Lamma, E., Mancarella, P., and Mello, P., "Normal Logic Programs as Open Positive Programs" in *Proceedings 1992 Joint International Conference on Logic Programming* (K. R. Apt and J. Minker, eds.) the MIT Press, 1992.
6) Brogi, A., Lamma, E., and Mello, P., "Open Logic Theories," in *Proceedings of Second Workshop on Extensions of Logic Programming*, *Lectures Notes in Artificial Intelligence*, H.-H. Eriksson, P. Krueger, and P. Schroeder-Heister, eds., Kista, Springer-Verlag, Jan. 1991.
7) Brogi, A., Mancarella, P., Pedreschi, D., and Turini, F., "Composition Operators for Logic Theories," in *Computational Logic* (J. W. Lloyd, ed.) , *Symposium Proceedings*, Springer-Verlag, 596, pp. 73-88 Brussels, pp. 117-134, Nov. 1990.
8) Charniak, E. and McDermott, D., *Introduction to Artificial Intelligence*, Addison-

Wesely, 1985.

9) Clark, K., "Negation as Failure," in *Logic and Data Bases* (H. Gallaire and J. Minker, eds.), Plenum, pp. 293-322, 1978.

10) Dung, P. M., "Negation as Hypothesis: An Abductive Foundation for Logic Programming," in *Proc. 8th International Conference on Logic Programming* (K. Furukawa, ed.), The MIT Press, pp. 3-17, 1991.

11) Eshgi, K. and Kowalski, R. A., "Abduction Compared with Negation by Failure," in *Proc. Sixth International Conference on Logic Programming* (G. Levi and M. Martelli, eds.), The MIT Press, pp. 234-254, 1989.

12) Gaifman, H. and Shapiro, E., "Fully Abstract Compositional Semantics for Logic Programs," in *Proc. Sixteenth POPL*, pp. 134-142, 1989.

13) Gelfond, M. and Lifschitz, V., "The Stable Models Semantics for Logic Programs," in *Proc. Fifth International Conference on Logic Programming* (R. A. Kowalski and K. A. Bowen, eds.), The MIT Press, pp. 1070-1080, 1988.

14) Kakas, A. C. and Mancarella, P., "Generalized Stable Models: a Semantics for Abduction," in *Proceedings of 9th European Conference on Artificial Intelligence* (L. Carlucci Aiello, ed.) Pitman, pp. 385-391, 1990.

15) Kowalski, R. A., "Problems and Promises of Computational Logic," in *Computational Logic, Symposium Proceedings* (J. W. Lloyd, ed.), Springer-Verlag, Brussels, pp. 1-36, Nov.1990.

16) Lloyd, J. W., *Foundations of Logic Programming*, second edition, Springer-Verlag, 1987.

17) Mancarella, P. and Pedreschi, D., "An Algebra of Logic Programs," in *Proc. Fifth International Conference on Logic Programming* (R. A. Kowalski and K. A. Bowen, eds.), The MIT Press, pp. 1006-1023, 1988.

18) Mancarella, P., Pedreschi, D., Rondinelli, M., and Tagliatti, M., "Algebraic Properties of a Class of Logic Programs," in *Proc. NACLP* (S. Debray and M. Hermenegildo, eds.), The MIT Press, pp. 23-49, 1990.

19) Miller, D., "A Logical Analysis of Modules in Logic Programming," *Journal of Logic Programming*, *6*, pp. 79-108, 1989.

20) Monteiro, L. and Porto, A., "Contextual Logic Programming," in *Proc. Sixth International Conference on Logic Programming* (G. Levi and M. Martelli, eds.), The MIT Press, pp. 284-302, 1989.

21) O'Keefe, R., "Towards an Algebra for Constructing Logic Prograams," in *Proceedings of IEEE Symposium on Logic Programming* (J. Cohen and J. Conery, eds.), IEEE Computer Society Press, pp. 152-160, 1985.

22) Poole, D., "Compiling a Default Reasoning System into Prolog," *New Generation Computing*, *9*, *1*, pp. 3-38, 1991.

23) Poole, D. L., "A Logical Framework for Default Reasoning," *Artificial Intelligence*, *36*, pp. 27-47, 1988.

24) Przymusinski, T. C., "Extended Stable Semantics for Normal and Disjunctive Programs," in *Proc. 7th International Conference on Logic Programming* (D. H. D. Warren and P. Szeredi, eds.), The MIT Press, pp. 459-477, 1990.

25) Przymusinski, T. C., "Semantics of Disjunctive Logic Programs and Deductive Databases," in *Proc. DOOD91* (C. Delobel, M. Kifer and Y. Masunaga, eds.), Springer-Verlag, pp. 85-107, 1991.

26) Reiter, R., "On Closed World Data Bases," in *Logic and Data Bases* (Gallaire and J. Minker, eds.), Plenum, pp. 293-322, 1978.

27) Sacca, D. and Zaniolo, C., "Stable Models and Nondeterminism in Logic Programs with Negation," in *Proc. 9th ACM Symp. on Principles of Database Systems*, ACM, pp. 205-217, 1990.

28) Sacca, D. and Zaniolo, C., "Partial Models and Three-Valued Models in Logic Programs with Negation," in *Proc. Workshop on Logic Programming and Non-Monotonic Reasoning*, The MIT Press, pp. 87-101, 1991.
29) van Emden, M. H. and Kowalski, R. A., "The Semantics of Predicate Logic as a Programming Language," *Journal of the ACM*, *23*, *4*, pp. 733-742, 1976.
30) Van Gelder, A., Ross, K. A., and Schlipf, J. S., "Unfounded Sets and the Well-Founded Semantics of General Logic Programs," in *Proc. ACM SIGMODSIGACT Symp. on Principles of Database Systems*, ACM, pp. 221-230, 1988.

Appendix

The proofs of the propositions referred in the paper are reported in this appendix. All the proofs are given by slightly adapting corresponding proofs well-known from the theory of logic programming. For example, the skeleton of the proof of monotonicity and continuity of the τ operator is the same of that of the T_P operator (see Ref. 16)). To simplify the proofs, we will use the following definition of τ:

$$(\,*\,)\ \tau_S(I) = \bigcup \{M \mid M^H \in S \textit{ and } H \subseteq I\}$$

which resembles the definition of T_P more closely than Definition 3.1. According to Definition 3.2, the admissible Herbrand model of $P \cup Q$ under a set of hypotheses H is determined by the formula $M = \tau_S \uparrow \omega(H)$ (where $S = [\![P]\!] \cup [\![Q]\!]$). It is easy to see that this is equivalent to the formula $M = \tau_{S \cup \{\{H\}^{\emptyset}\}} \uparrow \omega(\phi)$ where the ($*$) definition of τ is used. Intuitively, the set of hypotheses H is represented as an admissible Herbrand model under the empty set of hypotheses. This guarantees that $\forall i > 0$: $H \subseteq \tau_S \uparrow i(\phi)$ (it is thus superfluous to add I at each step of τ).

Proposition 7.1 (Monotonicity and Continuity of τ)
Let S be a set of admissible Herbrand models, the composition operator τ mapping Herbrand interpretations into Herbrand interpretations is monotonic and continuous.

Proof
(Monotonicity): trivial.
(Continuity): Let X be a chain. To prove that τ_S is continuous we have to show that: $\tau_S(lub(X)) = lub(\tau_S(X))$. We have that:

$$A \in \tau_S(lub(X))$$
$$\Leftrightarrow$$
$$\exists M^H \in S \text{ s. t. } A \in M \wedge H \subseteq lub(X)$$
$$\Leftrightarrow$$
$$\exists I \in X. \text{ s. t. } \exists M^H \in S \wedge A \in M \wedge H \subseteq I$$
$$\Leftrightarrow$$
$$\exists I \in X. \text{ s. t. } A \in \tau_S(I)$$
$$\Leftrightarrow$$
$$A \in lub(\tau_S(X)) \quad \square$$

The following lemma is used to prove proposition 3.2 (see below).

Lemma 7.2
Let P and Q be two programs, H an admissible set of hypotheses for $P \cup Q$, and let $S = [\![P]\!] \cup [\![Q]\!] \cup \{H^{\emptyset}\}$

$$\tau_S(I) \subseteq I \Leftrightarrow I \text{ is an Herbrand model of } P \cup Q \cup H.$$

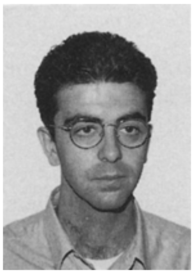
Proof

$(\Rightarrow)$ Assume that I is not a model of $P \cup Q \cup H$. Then there exists a clause $a \leftarrow b_1, ..., b_n$ in P (or Q) such that: $\{b_1, ..., b_n\} \subseteq I$ and $a \not\subseteq I$. Therefore there exists also an admissible Herbrand model $M^{\{b_1,...,b_n\}}$ of P (or Q) such that: $\{b_1, ..., b_n\} \subseteq I$ and $M \not\subseteq I$. We have that: $\tau_S(I) \not\subseteq I$ which contradicts the hypothesis.
$(\Leftarrow)$ Assume that $\tau_S(I) \not\subseteq I$. Then there exists an admissible Herbrand model $M'^{H'}$ of P (or Q) such that: $H' \subseteq I$ and $M' \not\subseteq I$. Since $M' = \mathcal{M}_{P \cup H'}$ then I is not a model for $P \cup H'$. This means that I is not a model for P (since $I \models P \cup H' \Leftrightarrow I \models P$ and $I \models H'$[7])). □

Proposition 3.2 (Composition of Admissible Herbrand Models)
Given two programs P and Q, let H be an admissible set of hypotheses of $P \cup Q$. Let $S = [\![P]\!] \cup [\![Q]\!] \cup \{H^{\{\}}\}$. M is an admissible Herbrand model for $P \cup Q$ under the hypotheses H if and only if $M = \tau_S \uparrow \omega(\phi)$.

Proof
The structure of the proof is analogous to Proposition 6.5. in Ref. 16).

$$\begin{aligned}
M &= \mathcal{M}_{P \cup Q \cup H} \\
&= glb\ \{I \mid I \text{ is an Herbrand model of } P \cup Q \cup H\} \\
&= gib\ \{I \mid \tau_S(I) \subseteq I\} && \text{(by Lemma 7.2)} \\
&= lfp\ \tau_S && \text{(by Proposition 7.1 and 5.1 of Ref. 16))} \\
&= \tau_S \uparrow \omega(\phi) && \text{(by Proposition 7.1)} \quad \square
\end{aligned}$$



Antonio Brogi: Dipartimento di Informatica, Universitã di Pisa, Corso Italia 40, 56125 Pisa, Italy
Research interests: Programming Language Design and Semantics, Logic Programming and Artificial Intelligence

Evelina Lamma, Ph. D.: Associate Professor, The University of Udine, (Present Address) Dipartimento di Electronica, Informatica e Sistemistica, Universitã di Bologna, Viale Risorgimento 2, 40136 Bologna, Italy
Research interests: Programming Languages and Knowledge Representation, with particular reference to Logic Languages and their extentions.

Paola Mello, Ph. D.: Associate Professor, Dipartimento di Electronica, Informatica e Sistemistica, Università di Bologna, Viale Risorgimento 2, 40136 Bologna, Italy
Research interests: Programming Languages and Knowledge Representation, with particular reference to theoretical and practical aspects of extentions of Logic Programming