R-stable Models for Logic Programs

H. Jakobovits* and D. Vermeir

Free University of Brussels, VUB
Dept. of Computer Science
Pleinlaan 2, Brussels 1050, Belgium

Abstract. We propose a new semantics for general logic programs which stems from first principles of logic-programming semantics. Our theory unifies previous approaches and is applicable to some useful programs which are not properly handled by existing semantics.

Keywords: logic programming, semantics

1 Introduction

The problem of defining semantics for general logic programs is a rapidly progressing area of research. Thus, there have recently emerged semantics such as the well-founded semantics [2], stable semantics [1], stable partial models [9], acceptable semantics [6], three-valued stable models [8], and regular models [11]. Each of these approaches suggests ways of associating models to a program. The various approaches seem to differ slightly in their motivations, in that they have different expectations for models. In addition, there are some reasonable models of certain useful programs which are not delivered by some of the approaches. This paper derives a unifying logic-programming semantics from first principles.

The paper is organized as follows: in Section 2, we define a new logic-programming semantics, called *simple models*, which is based on "first principles", and show that it is consistent with most other semantics that have been proposed in the literature. In Section 3, we show how simple models can be obtained from so-called *complete interpretations*, which are possibly inconsistent interpretations that satisfy a strict interpretation of the "negation-as-failure" principle. Section 4 introduces a proper restriction on simple models, called *r-stable models*. Intuitively, r-stable models are stable in that a further refinement of such a model (i.e. deciding on undefined literals) does not result in an inconsistency. Finally, in Section 5, we discuss the relationship of r-stable semantics with other approaches.

2 First Principles

* acknowledges the support of the National Science Foundation, NFWO

Definition 1. A **logic program** P is a countable set of rules of the form

$$p \leftarrow C$$

where p is an atom and C is a finite set of literals, i.e. atoms or negated atoms. The conclusion, p, is called the *head* and the set C of subgoals is called the *body*. The arrow may be read "if".

The **Herbrand universe** of P is the set of all possible ground (i.e. variable-free) terms that can be constructed using symbols from P. The **Herbrand base**, denoted $\mathcal{B}_P$, is the set of all possible ground atoms whose predicate symbols occur in the program and whose arguments are elements of the Herbrand universe.

An **interpretation** I of P is any set of literals from $\mathcal{B}_P \cup \neg\mathcal{B}_P$, where $\neg\mathcal{B}_P$ denotes the set $\{\neg p \mid p \in \mathcal{B}_P\}$. We use I^+ to denote the set of atoms in I. Similarly, I^- is used to denote the set of atoms that occur negatively in I, i.e. $I^- = \{p \in \mathcal{B}_P \mid \neg p \in I\}$. I is said to be **consistent** if $I^+ \cap I^- = \emptyset$. A **positive interpretation** of P is any subset of $\mathcal{B}_P$.

Notice that we do not require P to be finite. In this paper, we simply equate a traditional (finite) logic program with the set of ground instances of its rules.

In the logic programs of the form considered here, the heads of the rules are positive literals. There is no way, therefore, to prove that a negative literal holds. The accepted practice is thus to rely on the "closed-world assumption", which implies accepting the negation of a literal when the literal cannot be shown to hold. This well-known principle, known as "negation-as-failure" (naf), is widely recognized, but its exact implementation remains controversial. One of the requirements of any logic-program semantics, therefore, is to provide an implementation of naf. In most classical semantics this is achieved by means of the notion of unfoundedness, which was introduced in [2] as follows:

Definition 2. Let P be a program and I an interpretation of P. A set $U \subseteq \mathcal{B}_P$ of atoms is an **unfounded set** of P with respect to I if for each $p \in U$ and for each rule $p \leftarrow C$ in P, either

- $C \cup I$ is inconsistent, or
- $C \cap U \neq \emptyset$.

Intuitively, a set U of literals is unfounded with respect to an interpretation I if, for any $p \in U$, all rules that could motivate p are disqualified, where a rule $p \leftarrow C$ may be disqualified either because it is "blocked" (i.e. the body of the rule is inconsistent with the interpretation I) or because it depends on another unfounded literal in U. Thus, if U is unfounded with respect to I then no literal in U can be proven once one has accepted I.

Clearly, if U_1 and U_2 are unfounded w.r.t. an interpretation I, then so is their union $U_1 \cup U_2$. This motivates the following definition from [2]:

Definition 3. Let I be an interpretation of a logic program P. The **greatest unfounded set** of P with respect to I, denoted $\mathcal{U}_P(I)$ (or $\mathcal{U}(I)$ if P is understood) is the union of all sets that are unfounded with respect to I.

The fundamental significance of the notion of unfoundedness is evident, since, as has been pointed out in [7], many recent approaches (including all those semantics which are included in the pure semantics, i.e. stable models, stable partial models, the well-founded (partial) model, three-valued stable models, and regular models) have required that $\mathcal{U}(I^+) = I^-$. However, we show by the following example that unfoundedness is insufficient to capture naf:

Example 1. Consider the following program:

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

The well-founded model (as defined in [2]) for this program is $I = \emptyset$. Notice that the literal q, which is not in $\mathcal{U}(I^+)$, cannot be proven since any proof of q would require $\neg p$ to hold which, in turn, would imply p and, thus, an inconsistency. The underivability of q is captured in the suggested interpretation $\{\neg q\}$.

Thus, there may be positive literals which are not in $\mathcal{U}(I^+)$ and which cannot be proven. This motivates us to provide a semantics which extends the implementation of naf to include the negation of such literals.

The following definition will be useful in the definition of our semantics:

Definition 4. Let P be a logic program and let I be an interpretation of P.

- A literal p is a **logical consequence** of I with respect to P, denoted $I \vdash_P p$ (or $I \vdash p$ if P is understood) if either
  - $p \in I$, or
  - there exists a rule $p \leftarrow C$ in P such that $I \vdash_P c$ for each $c \in C$.
- A set of literals X is a logical consequence of I if each of its elements is a logical consequence of I, in which case we write $I \vdash X$.
- The **deductive closure** of a set of literals X with respect to P is the set $\{p \mid X \vdash_P p\}$ and is denoted X^*_P (or X^* if P is understood).

We consider the following principles as criteria for an interpretation I to be included in a semantics. The first requirement, which is universally accepted, is that I be consistent and deductively closed. Second, as described in [9], I should contain no assumption sets. This means that any set of positive literals $X \subseteq I^+$ must be derivable from $I \backslash X$. We express this as the condition $\neg I^- \vdash I^+$. Of course, there remains then to specify which negative literals are acceptable. Certainly, any $p \in \mathcal{U}(I^+)$ must be negated since there is no way to prove p once one has accepted I^+. In order to include in our semantics the models such as that mentioned in the previous example, we relax the forementioned naf equality to the inequality $\mathcal{U}(I^+) \subseteq I^-$. Thus, we suggest the following axiomatic definition of our new semantics:

Definition 5. Let P be a logic program and let I be an interpretation. I is called a **simple model** of P if I satisfies the following conditions:

1. I is a **model**, i.e.

- I is consistent.
- I is deductively closed (in other words, I satisfies the rules of P).

2. I is **founded**, i.e.
   - $\neg I^- \vdash I^+$
3. I respects the minimal requirement of **negation as failure**, i.e.
   - $\mathcal{U}(I^+) \subseteq I^-$

In order to show that most classical semantics satisfy the forementioned criteria, we refer to the pure semantics, which is defined in [7] as follows:

Definition 6. Let P be a program and J a positive interpretation of P. A set of atoms A is an **assumption set** of P with respect to J if for each $p \in A$ and for each rule $p \leftarrow C$ in P, either

- $C \cup J$ is inconsistent, or
- $C \cap A \neq \emptyset$, or
- $C \not\subseteq J \cup \neg\mathcal{U}(J)$.

Clearly, if A_1 and A_2 are assumption sets w.r.t. an interpretation J, then so is their union $A_1 \cup A_2$. Thus, the **greatest assumption set** $\mathcal{A}(J)$ with respect to a positive interpretation J is well-defined. An interpretation I is a **pure model** of P if $I^- = \mathcal{U}(I^+)$ and $\mathcal{A}(I^+) = \mathcal{B}_P \backslash I^+$, i.e. all atoms not in I are assumptions.

Theorem 7. *Let P be a logic program. An interpretation I of P is a pure model iff I is a simple model and $I^- = \mathcal{U}(I^+)$.*

Theorems 2, 3 and 4 from [7] then imply:

Corollary 8. *Let P be a logic program. All stable models, stable partial models, the well-founded (partial) model, three-valued stable models, and regular models of P are simple models of P.*

Example 2. Consider again the program P of example 1:

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

The simple models of P are $\emptyset$ and $\{\neg q\}$.

Example 3. Consider the following program P:

$$p \leftarrow \neg p$$
$$q \leftarrow \neg p$$
$$r \leftarrow \neg q$$
$$s \leftarrow \neg r$$

It is easy to verify that the simple models of P are $I_1 = \emptyset$, $I_2 = \{\neg s\}$, $I_3 = \{\neg r, s\}$, and $I_4 = \{\neg q, r, \neg s\}$. Notice that the empty set is the only model which is delivered by the pure semantics.

3 Reduced Models

As we mentioned in the previous section, the implementation of the negation-as-failure principle is not completely determined in the simple semantics since, in a simple model I, the set $\mathcal{U}(I^+)$ may be a strict subset of I^-. In this section we show that any simple model I is generated by a so-called "complete" interpretation J which does satisfy the equality $J^- = \mathcal{U}(J^+)$, but which is not necessarily consistent nor deductively closed. The transition from complete interpretations to simple models, which we call a "reduction", consists of the elimination of inconsistencies. Thus, the reduction of a complete interpretation is motivated by the requirement of consistency and deductive closure, but also results in the loss of the equality $J^- = \mathcal{U}(J^+)$.

Definition 9. An interpretation J of a logic program P is called a **complete interpretation** of P if J satisfies the following conditions:

1. $\mathcal{U}(J^+) = J^-$
2. $\neg J^- \vdash J^+$
3. J is **total**, i.e. $J^+ \cup J^- = \mathcal{B}_P$.

Definition 10. An interpretation I of a logic program P is called a **reduced model** of P if there is a complete interpretation J of P such that

$$I^+ = J^+ \backslash J^-$$
$$I^- = J^- \backslash J^+.$$

Theorem 11. *An interpretation I of a logic program P is a simple model iff it is a reduced model.*

Example 4. The simple model $I_3 = \{\neg r, s\}$ of the program in example 3 is a reduction of the complete interpretation $J = \{p, \neg p, q, \neg q, \neg r, s\}$.

Example 5. Consider the following program:

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

This program has three simple models. The simple model $\emptyset$ is a reduction of the complete interpretation $\{p, \neg p, q, \neg q\}$. The simple models $\{p, \neg q\}$ and $\{\neg p, q\}$ are reductions of themselves.

4 R-stable Models

In the previous sections we have defined a semantics which generates partial models for logic programs. Thus, our semantics differs from those approaches which define only total models. In those approaches, models contain truth values for all literals in the Herbrand base and there are, thus, no "undecided literals".

It is only natural now for us to investigate the stability of a partial model upon extension.

When a model I is proposed, the so-called remainder program P_I consists of rules which concern the remaining undecided literals and which take into account the truth values of the literals contained in the model. Thus, the remainder program P_I does not refer to any literals from $I \cup \neg I$. A model J of P_I may then provide a possible "extension" of I to $(I \cup J)^*$. However, J may be incompatible with I because $(I \cup J)^*$ might be inconsistent.

In this section, we examine a subclass of simple models that is "stable" under extension as described above, i.e. in order for I to be a stable model, we demand that $(I \cup J)^*$ be consistent for any (stable) model J of P_I.

The following definition uses a result due to Tarski in [10], which says that every monotonic function on a complete lattice has a least fixpoint.

Definition 12. Let I be a simple model of a logic program P. The **remainder program** of P with respect to I, denoted P_I, is defined as follows:

1. Let R be the set of rules $p \leftarrow C$ which can be constructed using literals from $\mathcal{B}_P \cup \neg\mathcal{B}_P$. Let Φ_P be the least fixpoint of the following monotonic function:

   $\phi_P : R \rightarrow R$

   $\phi_P(X) = P \cup X \cup \{p \leftarrow C \mid p \leftarrow C' \in X, q \in C', q \leftarrow D \in X, C = (C' \backslash q) \cup D\}.$

2. From the program Φ_P, delete all rules $p \leftarrow C$ such that
   - C contains a positive literal, or
   - $p \in I^+ \cup I^-$, or
   - $C \cup I^+$ is inconsistent.

Intuitively, every rule is replaced by the corresponding derivations of a literal. Any remaining rule that contains a positive literal in the body can be deleted since either it is subsumed by new rules, or it does not correspond to a derivation (i.e. the head cannot be proven since its truth value depends on the truth value assigned to another positive literal). In particular, rules that represent endless loops are deleted. In addition, rules for $p \in I^+ \cup I^-$ are left out since we already decided on p. A remaining rule $q \leftarrow C$ is useless if $C \cup I^+$ is inconsistent (note that $C \subseteq \neg\mathcal{B}_P$), so we can remove it.

Example 6. Consider the following program P:

$$p \leftarrow q$$
$$q \leftarrow \neg p$$
$$r \leftarrow \neg q$$
$$s \leftarrow \neg r$$
$$s \leftarrow \neg s$$

It is easy to verify that $L = \{\neg r, s\}$ is a simple model of P. The remainder program P_L consists of the following rules:

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

Stability of the deciding process is captured by the following recursive notion:

Definition 13. A simple model I of a logic program P is called **r-stable** if for any r-stable model J of P_I, the set $(I \cup J)^*_P$ is consistent.

Notice that a base case for the recursion, which includes two possible situations, is included in definition 13. In the first situation, the simple model I is empty, and is therefore trivially r-stable; indeed, when I is empty, for any simple model J of P_I, $(I \cup J)^*_P = J$ is consistent. In the second situation, I is non-empty but there is no non-empty simple model J of P_I. In this case, the only possible simple model of P_I is $J = \emptyset$, and $(I \cup \emptyset)^*_P = I$ is consistent; therefore, I is r-stable. Since $\mathcal{B}_{P_I} \subseteq \mathcal{B}_P$, we can show by standard techniques [4] that a base case is reached.

Example 7. Consider the simple model $I_3 = \{\neg r, s\}$ of the program P in example 3:

$$p \leftarrow \neg p$$
$$q \leftarrow \neg p$$
$$r \leftarrow \neg q$$
$$s \leftarrow \neg r$$

It can easily be verified that $J = \{\neg q\}$ is a simple model of the remainder program P_{I_3}:

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

J is r-stable since the remainder program $P_{I_3\,J}$ contains only the rule

$$p \leftarrow \neg p$$

and, clearly, the only simple model of $P_{I_3\,J}$ is $K = \emptyset$.

However, $(I_3 \cup J)^*_P = \{\neg r, s, \neg q, r\}$ is inconsistent; thus, the simple model $I_3 = \{\neg r, s\}$ of P is not r-stable.

Example 8. Consider the simple model $I_4 = \{\neg q, r, \neg s\}$ of the program P in example 3. The remainder program P_{I_4} is the following:

$$p \leftarrow \neg p$$

The only simple model of P_{I_4} is $J = \emptyset$, so I_4 is r-stable.

Example 9. Consider the following program P:

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

The simple models of P are $\emptyset$, $\{\neg r\}$, $\{\neg p, q, \neg r\}$ and $\{p, \neg q, r\}$. The r-stable models of P are $\emptyset$, $\{\neg p, q, \neg r\}$ and $\{p, \neg q, r\}$. The simple model $\{\neg r\}$ is not r-stable. Indeed, the remainder program with respect to $\{\neg r\}$ is

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

which has, among its r-stable models, the model $\{p, \neg q\}$. This r-stable model is incompatible with $\{\neg r\}$, since $(\{p, \neg q\} \cup \{\neg r\})^*_P = \{p, \neg q, \neg r, r\}$ is inconsistent.

Example 10. The simple model $L = \{\neg r, s\}$ of the program P in example 6 is not r-stable. The r-stable models are $\{\neg q, r\}$ and $\emptyset$. The unique pure model is the empty set.

Example 11. The simple models of the program in example 5 are all r-stable.

5 Relationships with Other Semantics

In this section we show that the inclusion relationships between the various semantics are as depicted in Fig. 1.

Lemma 14. *Let I be a simple model of a logic program P. If $I^- = \mathcal{U}(I^+)$ then I is r-stable.*

The following theorem is a consequence of Theorem 7 and Lemma 14:

Theorem 15. *Let I be a pure model of a logic program P. Then I is a r-stable model.*

Corollary 16. *Let P be a logic program. All stable models, stable partial models, the well-founded (partial) model, three-valued stable models, and regular models of P are r-stable models of P.*

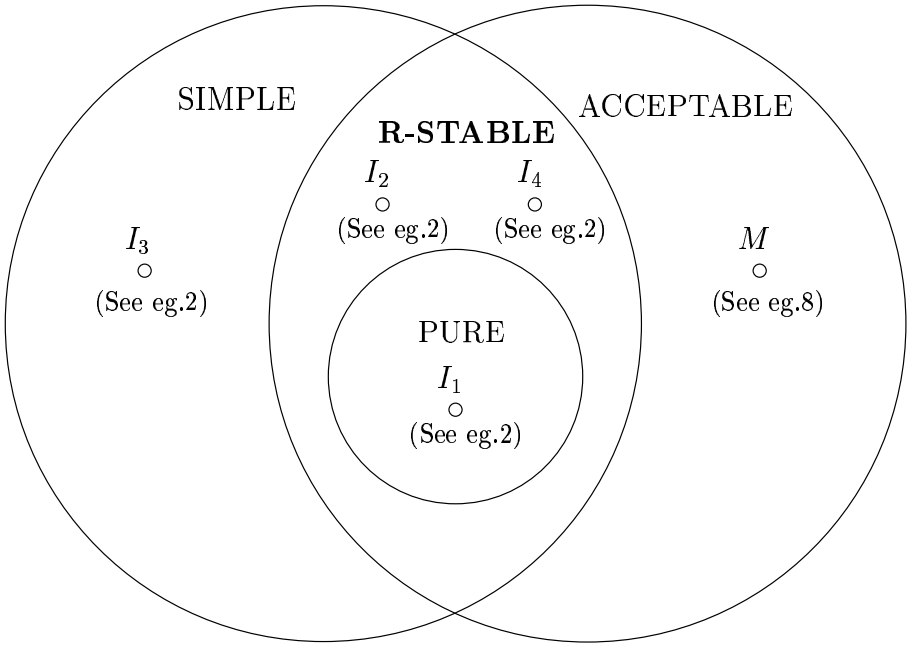

Fig. 1

Thus, all of these models are included in our semantics. Our examples showed that there are reasonable simple models which are not included in the forementioned approaches. Notice that all of the r-stable models mentioned in our examples are delivered by the model-theoretic semantics associated with the acceptable semantics in [6]; however, we shall show in example 12 that the set of acceptable models is actually too large. The following definition is taken from [6]:

Definition 17. Let P be a logic program and let A and B be subsets of $\neg\mathcal{B}_P$. Then A **attacks** B iff $A \vdash p$ for some $\neg p \in B$.

A is **acceptable** w.r.t. B iff for any $X \subseteq \neg\mathcal{B}_P$ such that X attacks $A \backslash B$, X is not acceptable w.r.t. $A \cup B$.

A is an **acceptable extension** of P iff A is acceptable w.r.t. $\emptyset$.

In the following definition we show that the acceptable extensions can be used to define a model-theoretic semantics:

Definition 18. Let I be an interpretation of a logic program P. I is an **acceptable model** of P iff there is an acceptable extension A of P such that $I = A^*$.

Theorem 19.
- *Let I be a simple model of a program P. Then I is an acceptable model iff it is r-stable.*
- *Let J be an acceptable model of a program P. Then J is simple iff it is r-stable.*

The following example illustrates that acceptable models do not necessarily comply with the first principles of logic-programming semantics described in definition 5:

Example 12. Consider the program

$$p \leftarrow \neg p$$
$$q \leftarrow \neg p$$
$$r \leftarrow \neg q, \neg s$$
$$s \leftarrow$$

Clearly, $\emptyset$ is an acceptable extension. Its deductive closure is $M = \{s\}$. This model does not respect negation as failure (in the sense of Condition 3 in definition 5), since $\neg\mathcal{U}(M^+) = \{\neg r\} \not\subseteq M$. Therefore, M is not a simple model.

Superfluous extensions such as the one in the above example can be avoided by considering only maximal acceptable extensions, as suggested for stable theories in [5]; however, the resulting semantics would then not include the well-founded model $\{s, \neg r\}$ of the above program, nor the model $\{p, \neg q\}$ of the following program:

$$p \leftarrow \neg q$$
$$q \leftarrow \neg p$$
$$r \leftarrow \neg s$$
$$s \leftarrow \neg r$$

The model $\{p, \neg q\}$ of this program reflects the ability to choose between the alternative literals p, q and not to decide on either of the alternative literals r, s. Maximizing acceptable extensions thus implies losing the appeal of a single unifying definition covering all possibilities between well-founded and r-stable semantics.

In [3] we have defined a logic-programming semantics based on argumentation frameworks. The idea developed there is that every logic program can be associated with an argumentation framework $AF = (A, \rightsquigarrow)$, where A is a set of arguments and $\rightsquigarrow$ is a binary relation on A which represents attacks between arguments. Each argument in the framework is a derivation of a literal p. A derivation of p attacks a derivation of q if the derivation of q relies on $\neg p$. This is reasonable since, if p is true, a proof for q that uses $\neg p$ is useless. Once a logic program is represented as an argumentation framework, any semantics for argumentation frameworks generates models for the logic program. The argumentation semantics which we have developed in [3] thus results in a logic-programming semantics, and the following theorem shows that the semantics derived from first principles of logic programming coincides with the semantics which results from argumentation theory.

Theorem 20. *Let P be a logic program.*

– *An interpretation of P is a simple model iff it is a labelling-model, and*
– *a simple model of P is r-stable iff it is robust,*

where labelling-models and robustness are defined in [3].

References

1. Michael Gelfond and Vladimir Lifschitz. The stable model semantics for logic programming. In Robert A. Kowalski and Kenneth A. Bowen, editors, *Proceedings of the Fifth International Conference and Symposium on Logic Programming*, pages 1081–1086, Seattle, 1988. ALP, IEEE, The MIT Press.
2. A. Van Gelder, K. Ross, and J. S. Schlipf. Unfounded sets and well-founded semantics for general logic programs. In *Proceedings of the Seventh ACM Symposium on Principles of Database Systems*, pages 221–230, Austin, Texas, 1988. Association for Computing Machinery.
3. H. Jakobovits and D. Vermeir. Contradiction in argumentation frameworks. In *Proceedings of the IPMU conference*, 1996.
4. H. Jakobovits and D. Vermeir. Argumentation and logic-programming semantics. In preparation.
5. A.C. Kakas and P. Mancarella. Stable theories for logic programs. In *Proceedings of the International Symposium of Logic Programming*. 1991.
6. A. C. Kakas, P. Mancarella, and Phan Minh Dung. The acceptability semantics for logic programs. In P. Van Hentenrijck, editor, *Proceedings of the 11th International Conference on Logic Programming*, pages 504–519. MIT Press, 1994.
7. E. Laenens, D. Vermeir, and C. Zaniolo. Logic programming semantics made easy. In *Proceedings of the International Conference on Automata, Languages and Programming*, pages 499–508. 1992.
8. T. Przymusinski. Well-founded semantics coincides with three-valued stable semantics. *Fundamenta Informaticae*, 13:445–463, 1990.
9. D. Sacca and C. Zaniolo. Stable models and non-determinism for logic programs with negation. In *Proceedings of the 9th ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*. Association for Computing Machinery, 1990.
10. A. Tarski. A lattice theoretical fixpoint theorem and its application. *Pacific Journal of Mathematics*, (5):285–309, 1955.
11. Jia-Huai You and Li Yan Yuan. Three-valued formalization of logic programming: Is it needed? In *Proc. of the PODS'90 conference*, pages 172–182. 1990.

This article was processed using the LaTeX macro package with LLNCS style