Unique Supported-Model Classes of Logic Programs

Pascal Hitzler* and Anthony Karel Seda

Department of Mathematics, University College Cork, Cork, Ireland.
{phitzler,aks}@ucc.ie http://maths.ucc.ie/∼{pascal,seda}/

Abstract

We study classes of programs, herein called *unique supported-model classes*, with the property that each program in the class has a unique supported model. Elsewhere, the authors examined these classes from the point of view of operators defined relative to certain three-valued logics. In this paper, we complement our earlier results by considering how unique supported-model classes fit into the framework given by various classes of programs in several well-known approaches to semantics.

Keywords: logic programming, denotational semantics, supported-model semantics

1 Introduction

Logic programming is concerned with the use of logic as a programming language. The main manifestation of this computing paradigm is via the various versions of Prolog which are now available, in which computation is viewed as deduction from sets of Horn clauses, although there is also growing interest in the related form known as answer set programming, see [14]. The reference [1] contains a good survey of the growth of logic programming over the last twenty-five years both as a stand-alone programming language and as a software component of large information systems.

One advantage a logic program P has over conventional imperative and object-oriented programs is that it has a natural machine-independent meaning, namely, its logical meaning. This is often referred to as its declarative semantics, and is usually taken to be some "standard" model canonically associated with P. Unfortunately, there are often many possible choices of the standard model such as the well-founded model, see [8], the stable model, see [9], the perfect model, see [16], the weakly-perfect model, see [16] again, and so on, which do not in general coincide and all of which have a claim to be "the natural choice" depending on one's view of non-monotonic reasoning. It is therefore important and interesting to know when these various models coincide since this confirms coincidence of the various ways of considering non-monotonic reasoning. A particular case of this is the study of classes of programs which have only one supported model, for it is reasonable to expect that many of the standard models will coincide for such programs, and indeed we show here that this is sometimes the case and examine the extent to which it is so.

*The first named author acknowledges financial support under grant SC/98/621 from Enterprise Ireland.

The supported-model semantics or Clark-completion semantics [6] for logic programs is a very natural way of assigning meaning to logic programs. However, in contrast with some of the other semantical approaches mentioned above, a program can have several different meanings under the supported-model semantics, a property which it shares with its refinement, the stable model semantics [9]. Whether this is to be thought of as a weakness or a feature depends, of course, on one's point of view, and this is discussed in [14]. Nevertheless, some classes of programs which have been discussed in the literature have the property that each program in the class has a *unique* supported model. Such classes were called *uniquely determined* in [3], but are called here by the more revealing name *unique supported-model classes* and loosely denoted by *usm* (we sometimes write $P \in$ USM to indicate that P has a unique supported model). It is these classes which form the main object of study here.

Usm classes include some programs which have been studied in the context of termination: acyclic and acceptable programs [2] for example, and some straightforward generalizations of these such as the locally hierarchical programs [5]. In the earlier paper [11], we gave a unifying approach to some unique supported-model classes using operators in three-valued logics which, amongst other things, led to the definition of the usm class of all Φ^*-accessible programs. In fact, the definition of these programs was partly motivated by the study in [15] of the connections between acceptability and stratifiability. Indeed, the class of Φ^*-accessible programs is remarkable in that not only does each program in it have a unique supported model, but in addition it is computationally adequate in the sense that every partial recursive function can be implemented, under Prolog, by such a program. This does not hold for the subclass of acceptable programs which always terminate under the Prolog selection rule.

Thus, by working with usm classes, one is working in a setting which is, semantically, relatively unambiguous, and one may still study classes of programs which range from having strong termination properties to full computational adequacy. The approach in [11] mentioned earlier led to a comparison of several usm classes and the determination of some of their distinguishing properties. It is the main objective of this paper to complement these results by examining how usm classes fit into the framework given by other classes of programs and other approaches to semantics.

The paper is structured as follows. In Section 2, after some preliminaries, we will define Φ-accessible programs which generalize the Φ^*-accessible programs mentioned above. An alternative characterization of these will be given in Section 3. In Section 4, we will study Φ-accessible programs in the context of weak stratification and we will see that for these programs the supported, well-founded, weakly-perfect, and stable model semantics coincide. It will also be shown that this result cannot be generalized to all programs with unique supported models. Finally, in Section 5, we discuss some of the conclusions which can be drawn from our results.

2 Preliminaries

We study normal logic programs P, that is, finite sets of clauses of the form $A \leftarrow L_1, \ldots, L_n$ which may be written in the form $A \leftarrow \mathbf{body}$, where A is an atom and the L_i are literals drawn from the underlying first order language $\mathcal{L}$ of P. We work over Herbrand preinterpretations only, although many of our results carry over easily to more general settings, see [10]. Thus, B_P denotes the Herbrand base for a given program P, I_P denotes the power set of B_P

p	q	$p \wedge q$	$\neg p$
t	t	t	f
t	u	u	f
t	f	f	f

p	q	$p \wedge q$	$\neg p$
u	t	u	u
u	u	u	u
u	f	f	u

p	q	$p \wedge q$	$\neg p$
f	t	f	t
f	u	f	t
f	f	f	t

Table 1: Truth table for Kleene's strong three-valued logic.

which can be identified, as usual, with the set of all (Herbrand-) interpretations for P. Our reference to standard concepts and notation in logic programming is [13]. By ground(P) we will denote the program consisting of all ground instances of clauses in P. A level mapping for a program P is a mapping $l : B_P \to \alpha$, where α is a countable ordinal (we always extend l to ground literals by defining $l(\neg A) = l(A)$ for all ground atoms A).

We assume that the reader is familiar with the stable model semantics [9] and the well-founded-model semantics [8] for normal logic programs. Given a program P, the *supported models* for P are the fixed points of the single-step operator $T_P : I_P \to I_P$. As usual, $T_P(I)$ is defined to be the set of all $A \in B_P$ such that there exists a clause $A \leftarrow \texttt{body}$ in ground(P) with the property that `body` is true in I. Recall that supported models can be identified with the models of the Clark completion of P, see [6]. We also assume that the reader is familiar with the notion of weakly-stratified program, see [4, 16].

Following [7], a *partial* (or *three-valued*) *interpretation* I is a pair (I^+, I^-) of subsets of B_P such that I^+ and I^- are disjoint. Partial interpretations will be interpreted in Kleene's strong three-valued logic: given a partial interpretation $I = (I^+, I^-)$, atoms in I^+ carry the truth value *true* (t) in I and atoms in I^- the value *false* (f) in I. Atoms which are neither in I^+ nor in I^- carry the truth value *undefined* (u). For our purposes, we only need to know how conjunction and negation operate in this three-valued logic, and this is given by the truth table in Table 1.

A partial interpretation (I^+, I^-) is called *total* if $I^+ \cup I^- = B_P$, and such interpretations can be naturally identified with elements of I_P. The set $I_{P,3}$ of all partial interpretations is a complete partial order, indeed complete semi-lattice, under the ordering $(I_1^+, I_1^-) \leq (I_2^+, I_2^-)$ iff $I_1^+ \subseteq I_2^+$ and $I_1^- \subseteq I_2^-$, with bottom element $\bot = (\emptyset, \emptyset)$. Total interpretations are in fact maximal elements in the given ordering.

The three-valued operator Φ_P (see [7]) is defined as a mapping on partial interpretations K as follows. We set $\Phi_P(K) = (I^+, I^-)$, where I^+ is the set of all $A \in B_P$ with the property that there exists a clause $A \leftarrow \texttt{body}$ in ground(P) such that `body` is true in K, and I^- is the set of all $A \in B_P$ such that for all clauses $A \leftarrow \texttt{body}$ in ground(P) we have that `body` is false in K; truth and falsehood being taken here relative to Kleene's strong three-valued logic, as already noted. We note further that Φ_P is monotonic, and hence we define

$$\begin{aligned} \Phi_P{\uparrow}0 &= \bot, \\ \Phi_P{\uparrow}(k+1) &= \Phi_P(\Phi_P{\uparrow}k) \text{ for any ordinal } k, \text{ and} \\ \Phi_P{\uparrow}\alpha &= \sup\{\Phi_P{\uparrow}\beta \mid \beta < \alpha\} \text{ for any limit ordinal } \alpha. \end{aligned}$$

2.1 Definition A normal logic program P is called Φ-*accessible* if there exists an ordinal α such that $\Phi_P{\uparrow}\alpha$ is a total interpretation. The smallest ordinal with this property is called the *closure ordinal* of P with respect to Φ.

As observed in [11], each acceptable [2], acyclic [5], or locally hierarchical [5] program is Φ-accessible, and each Φ-accessible program has a unique supported model. Indeed, in [11], we described more variations of the operator Φ, each of which gives rise to a usm class analogous to the Φ-accessible programs. Of all the classes described in [11], the Φ-accessible programs form the largest.

3 Φ-accessible Programs

Following the definition of acceptable programs, we define next a superclass of these denoted by $[\Phi]$. As it turns out, this class contains exactly the Φ-accessible programs.

3.1 Definition Let P be a normal logic program. Then P is contained in $[\Phi]$ if and only if there exists a level mapping l for P and a (two-valued) model I for P such that the following condition holds. Each $A \in B_P$ satisfies either (i) or (ii):

(i) There exists a clause $A \leftarrow L_1, \ldots, L_n$ in ground(P) with head A such that $I \models L_1 \wedge \ldots \wedge L_n$ and $l(A) > l(L_i)$ for all $i = 1, \ldots, n$.
(ii) For each clause $A \leftarrow L_1, \ldots, L_n$ in ground(P) with head A there exists $i \in \{1, \ldots, n\}$ such that $I \not\models L_i$, $I \not\models A$ and $l(A) > l(L_i)$.

For every Φ-accessible program, we define a canonical level mapping as follows.

3.2 Definition Let P be Φ-accessible. For each $A \in B_P$, let $l_P(A)$ denote the least ordinal α such that A is not undefined in $\Phi_P \uparrow (\alpha+1)$. We call the resulting mapping l_P the *canonical level mapping for P with respect to Φ.*

3.3 Theorem The class $[\Phi]$ contains exactly the Φ-accessible programs.

Proof: Let P be Φ-accessible, let l_P be its canonical level mapping with respect to Φ, let α be its closure ordinal with respect to Φ and let $M_P = \Phi \uparrow \alpha^+$ be its unique supported (two-valued) model.
(a) Let $A \in M_P$ and let $l_P(A) = \beta$. By definition of l_P and Φ_P, there exists a clause $A \leftarrow L_1, \ldots, L_n$ in ground(P) such that the $L_1, \ldots, L_n$ are true in $\Phi \uparrow \beta$, and hence are also true in M_P. Again, by definition of l_P, we obtain $l_P(A) > l_P(L_i)$ for all i.
(b) Let $A \notin M_P$ and let $l_P(A) = \beta$. By definition of l_P and Φ_P, we obtain that for any clause $A \leftarrow L_1, \ldots, L_n$ in ground(P) we must have that $L_1 \wedge \ldots \wedge L_n$ is false in $\Phi_P \uparrow \beta$. So, there must be some i such that L_i is false in $\Phi_P \uparrow \beta$ and $l_P(L_i) < \beta$ by definition of l_P, and hence $l_P(A) > l_P(L_i)$. Thus, $P \in [\Phi]$.

Conversely, let $P \in [\Phi]$ so that P satisfies Definition 3.1 with respect to a model I and a level mapping l. We show by induction on β that any $A \in B_P$ with $l(A) = \beta$ is not undefined in $\Phi_P \uparrow (\beta + 1)$ and, furthermore, that I and $\Phi_P \uparrow (\beta + 1)$ agree on A. Indeed, if $l(A) = 0$, then A must be the head of a unit clause or does not appear in any head. In the first case, A is true in $\Phi_P \uparrow 1$, and in the second case, A is false in $\Phi_P \uparrow 1$. Note that in the first case A is also true in I since condition (i) of Definition 3.1 applies and I is a model of P. Also, in the second case, A is also false in I since condition (ii) of Definition 3.1 applies. Next, suppose $l(A) = \beta$. If there is no clause in ground(P) with head A, then A is false in $\Phi_P \uparrow 1 \leq \Phi_P \uparrow (\beta + 1)$ and also false in I since condition (ii) of Definition 3.1

applies. So assume there is a clause in ground(P) with head A. By definition of $[\Phi]$, either condition (i) or condition (ii) of Definition 3.1 applies. If condition (i) applies, then there is a clause $A \leftarrow L_1, \ldots, L_n$ in ground(P) such that $l(L_1), \ldots, l(L_n) < l(A)$ and therefore, by the induction hypothesis, the $L_1, \ldots, L_n$ are not undefined in $\Phi_P \uparrow \beta$ and I agrees with $\Phi_P \uparrow \beta$ on them. Now, since I is a model of P and $I \models L_1, \ldots, L_n$, we obtain that A is true in I and by definition of Φ_P also in $\Phi_P \uparrow \beta$. If condition (ii) applies, then for each clause $A \leftarrow L_1, \ldots, L_n$ in ground(P) there is some i such that $l(A) > l(L_i)$ and L_i is false in I. Hence we obtain that L_i is false in $\Phi_P \uparrow \beta$ by the induction hypothesis and it follows that A is false in both I and $\Phi_P \uparrow (\beta + 1)$. ■

4 Comparison with Other Approaches to Semantics

When studying various classes of programs, the question naturally arises as to how such classes relate to other known classes in the literature. From the definition, it follows immediately that the usm class of all locally hierarchical programs [5] is contained in the class of all locally stratified programs [16]. In this section, we will relate usm classes larger than the locally hierarchical programs to the notion of weak stratification.

It was pointed out in [4, Remark 5.3] that the original definition of weakly-stratified programs in [16] is ambiguous since the two conditions

(a) All strata of a program P consist of trivial components only.
(b) All layers of a program P are definite programs.

which were originally used for defining weakly-stratified programs are not equivalent. We will call a program *weakly stratified-a* if condition (a) holds, and *weakly stratified-b* if condition (b) holds. For a discussion of this, see [4, Section 5], and we refer to the same publication for notation concerning weakly-stratified programs.

In [15], it was shown that each acceptable program [2] is weakly stratified-a. From [8, Corollary 4.3], we immediately obtain that each Φ-accessible program has a total well-founded model, and hence is *effectively stratified* [4]. Again from [4, Proposition 5.4], we obtain that a program which is weakly stratified-b, is also effectively stratified.

It is easy to see that a program which is weakly stratified-b, is also weakly stratified-a. In the opposite direction, we have the following result.

4.1 Theorem If P is weakly stratified-a and if there exists no clause $A \leftarrow$ `body` in ground(P) with $\neg A$ occurring in `body`, then P is weakly stratified-b.

Proof: Since P is weakly stratified-a, all minimal components are trivial. Let $A \leftarrow$ `body` be a clause in a bottom layer. Without loss of generality assume that `body` contains some negative literal $\neg B$, thus[1] $B < A$, with $A \neq B$ by assumption. Since the component containing A is trivial, we obtain $A \not> B$ and therefore we obtain a contradiction. ■

It is clear from the last result that a locally hierarchical program is weakly stratified-a if and only if it is weakly stratified-b (this also holds for locally stratified programs [16]).

We next generalize the result mentioned earlier from [15].

[1] "$<$" denotes the dependency relation taking from the dependency graph of P [16].

4.2 Theorem If P is Φ-accessible, then P is weakly stratified-a and the unique supported model M_P of P is also its weakly-perfect-a model.

Proof: Let M_P be the unique supported model of P and let l be its canonical level mapping with respect to Φ. We can also assume without loss of generality that for each level α there exists some $A \in B_P$ with $l(A) = \alpha$.
(1) We first show that all components of the bottom stratum $S(P)$ of P are trivial. Assume that this is not the case, that is, that there exists a minimal component $C \subseteq S(P)$ which is not trivial. Then there must be some $A \in C$ with $l(A)$ minimal, and some $A' \in C$ with $A \neq A'$. Note that $A < A'$ and $A' < A$ [4, Definition 5.1]. Let B be an arbitrary atom occurring in a ground clause with head A. Then $B \leq A$ and $A < A'$, giving $B < A'$, and therefore $B < A$ since $A' < A$. Thus, by minimality of C we obtain $B \in C$. So all atoms B occurring in bodies of clauses in ground(P) with head A belong to C. Since P is Φ-accessible, however, there must exist some choice of B for which we have $l(B) < l(A)$, and this contradicts the minimality of $l(A)$. Note that the bottom stratum contains all atoms of level 0, and hence is non-empty.
(2) The model M of the bottom layer is compatible with M_P, that is, if a literal is true, respectively false, in M, then it is true, respectively false, in M_P. In order to see this, note that for every atom A in a minimal component, the bottom layer $L(P)$ contains all clauses with head A and all clauses with head being any of the body atoms of clauses in the bottom layer. Since the program P is Φ-accessible, it is easy to see that the subprogram formed by the bottom layer is also Φ-accessible and has a unique supported model which is compatible with M_P.
Now let A be an atom in $L(P)$ which occurs negatively in the body of some clause. Since all components are trivial, A must also be the head of the same clause so that $A < A$. If B is another body atom in the same clause, then we obtain $B < A$ and $A < B$ which contradicts triviality of all components. Hence, if some atom A occurs negatively in a clause in $L(P)$, then the clause is of the form $A \leftarrow \neg A$. All models of $L(P)$ must therefore assign the truth value true to all atoms occurring negatively in $L(P)$. The program which is obtained from omitting all these clauses is definite and has a least model which agrees with M_P. If we add to this model all atoms which occur negatively in $L(P)$, we obtain the least model of $L(P)$.
(3) We show that P/M is Φ-accessible (see [4]). This is indeed the case since (2) holds, and is easily seen by applying Theorem 3.3.
(4) We can now apply steps (1), (2) and (3) via transfinite induction as in [16], obtaining that P is indeed weakly stratified-a and that M_P is the weakly-perfect-a model of P. Thus, the proof is complete. ■

4.3 Theorem Let P be Φ-accessible. Then P has a unique supported model M_P which is the unique stable model, the well-founded model, and the weakly-perfect-a model of P.

Proof: We know that $M_P = \Phi_P \uparrow \alpha$ for some ordinal α and that M_P is total. By Theorem 4.2, we know that M_P is the weakly-perfect-a model of P. By [8, Corollary 4.3], $M_P = \Phi_P \uparrow \alpha$ is a subset of the well-founded model of P, and, since M_P is total, it must coincide with the well-founded model. By [8, Corollary 5.6], totality of the well-founded model implies that it coincides with the unique stable model of the program. This completes the proof. ■

A Φ-accessible program need not be weakly stratified-b, as can be seen from the program P consisting of the two clauses $p \leftarrow$ and $p \leftarrow q, \neg p$. The bottom layer contains the clause

$p \leftarrow q, \neg p$ and therefore is not a definite program. The program P is, however, Φ-accessible, even acceptable.

On the other hand, there exist programs $P \in \mathrm{USM}$ which are not weakly perfect-a. To see this, note that the program consisting of the three clauses $p \leftarrow \neg q$, $q \leftarrow \neg p$ and $p \leftarrow \neg p$ has unique supported model $\{p\}$ (which is a stable model). However, it has $\{p, q\}$ as a minimal component which is not trivial.

The property of having a unique stable model does not carry over to USM either: the program consisting of the two clauses $p \leftarrow p$ and $p \leftarrow \neg p$ has unique supported model $\{p\}$ but neither a stable model nor a total well-founded model.

5 Conclusions and Further Work

Many classes of programs are known in the literature, although the exact relationships between them are not yet fully understood. In this respect, the paper is a contribution towards the study of the "space" of all logic programs and the relationships between certain classes of its members.

The Φ- and Φ^*-accessible programs are remarkable from a semantical point of view. Both are natural generalizations of the locally hierarchical and acceptable programs and both classes are computationally adequate while maintaining an unambiguous semantics. They therefore provide an interesting programming environment. The smaller class of Φ^*-accessible programs is perhaps the more natural common generalization of the locally hierarchical and acceptable programs, since it treats all clauses equally, in the spirit of these two classes [11].

The supported-model semantics seems to be satisfactory for Φ-accessible programs but not for all programs in the larger class USM. Indeed, programs in $\mathrm{USM} \backslash [\Phi]$ may be considered to be unnatural from a logic programming point of view since they contain mutually recursive atoms whose truth values cannot be determined from the remaining program. Such a feature might make sense in the context of answer set programming [14], but the extent to which that is so remains to be seen.

It is natural to raise questions about other classes of programs which have unique (or unique total) models under semantics other than the supported-model semantics, and the authors are attempting to modify the approach presented here and in [11] for these purposes.

References

[1] Apt, K.R., Marek, V.W., Truszczyński, M. and Warren, D.S., The Logic Programming Paradigm: A 25-Year Perspective, Springer, Berlin, 1999.

[2] Apt, K.R. and Pedreschi, D., Reasoning about Termination of Pure Prolog Programs, Information and Computation 106 (1993), 109–157.

[3] Batarekh, A. and Subrahmanian, V.S., Topological Model Set Deformations in Logic Programming, Fundamenta Informaticae 12 (1989), 357–400.

[4] Bidoit, N. and Froidevaux, C., Negation by Default and Unstratifiable Logic Programs, Theoretical Computer Science 78 (1991), 85–112.

[5] Cavedon, L., Continuity, Consistency, and Completeness Properties for Logic Programs, in: Levi, G. and Martelli, M. (Eds.), Proceedings of the 6th International Conference on Logic Programming, MIT Press, Cambridge MA, 1989, pp. 571–584.

[6] Clark, K.L., Negation as Failure, in: Gallaire, H. and Minker, J. (Eds.), Logic and Data Bases, Plenum Press, New York, 1978, pp. 293–322.
[7] Fitting, M., A Kripke-Kleene-Semantics for General Logic Programs, Journal of Logic Programming 2 (1985), 295–312.
[8] Van Gelder, A., Ross, K.A. and Schlipf, J.S., The Well-Founded Semantics for General Logic Programs, Journal of the ACM 38:3 (1991), 620–650.
[9] Gelfond, G. and Lifschitz, V., The Stable Model Semantics for Logic Programming, in: Kowalski, R.A. and Bowen, K.A. (Eds.), Logic Programming. Proceedings of the 5th International Conference and Symposium on Logic Programming, MIT Press, 1988, pp. 1070–1080.
[10] Hitzler, P. and Seda, A.K., Acceptable Programs Revisited, Electronic Notes in Theoretical Computer Science, Vol. 30 No. 1, Elsevier 1999, pp. 1–18.
[11] Hitzler, P. and Seda, A.K., Characterizations of Classes of Programs by Three-Valued Operators, Lecture Notes in Artificial Intelligence, Vol. 1730, Springer, Berlin, 1999, pp. 357–371.
[12] Hitzler, P. and Seda, A.K., A New Fixed-point Theorem for Logic Programming Semantics, Proceedings of SCI2000 and ISAS2000, Orlando, Florida, USA, July, 2000, IIIS, 2000, pp. 418-423.
[13] Lloyd, J.W., Foundations of Logic Programming, Springer, Berlin, 1988.
[14] Marek, V.M. and Truszczyński, M., Stable Models and an Alternative Logic Programming Paradigm, in: Apt, K.R., Marek, V.W., Truszczyński, M. and Warren, D.S. (Eds.), The Logic Programming Paradigm, Springer, Berlin, 1999, pp. 375–398.
[15] Protti, F. and Zaverucha, G., On the Relations between Acceptable Programs and Stratifiable Classes, Lecture Notes in Computer Science, Vol. 1515, Springer, Berlin, 1998, pp. 141–150.
[16] Przymusinska, H. and Przymusinski, T.C., Weakly-Stratified Logic Programs, in: Apt, K.R. (Ed.), Special issue of Fundamenta Informaticae on Logical Foundations of Artificial Intelligence 13 (1990), 51–65.

Pascal Hitzler was born in Germany in 1971. He finished his studies of Mathematics and Computer Science at the University of Tübingen, Germany, in 1998 with first class honours. He is now a research assistant and Ph.D. student of Dr. A.K. Seda at University College, Cork, in Ireland. Besides his research activities in Mathematical Foundations of Computing he is also involved in running enhancement programmes for mathematically talented high-school students.

Anthony Karel Seda, Ph.D., C.Math, FIMA, is a Senior Lecturer in the Department of Mathematics, University College Cork, Ireland. His research interests in the past were in the area of mathematical analysis, but now are concerned with the mathematical foundations of computer science and, in particular, of computational logic and of artificial intelligence. He is a member of several learned societies including the New York Academy of Sciences and the European Association for Theoretical Computer Science (EATCS), and he edits the "News from Ireland" column of the Bulletin of the EATCS.