ARITHMETIC CLASSIFICATION OF PERFECT MODELS OF STRATIFIED PROGRAMS (Extended Version)*

Krzysztof R. Apt and Howard A. Blair†

Department of Computer Sciences
The University of Texas at Austin
Austin, Texas 78712-1188

TR-89-20 August 1989

*To appear in Fundamenta Informaticae. First version of this paper appeared in Proc. 5th International Conference on Logic Programming. The MIT Press, 1988.

†School of Computer and Information Science, 313 Link Hall, Syracuse University, Syracuse, NY 13244, USA.

Arithmetic Classification of Perfect Models of Stratified Programs

(Extended Version)*)

Krzysztof R. Apt
Centre for Mathematics and Computer Science
P.O. Box 4079, 1009 AB Amsterdam, The Netherlands
and
Department of Computer Sciences, University of Texas at Austin,
Austin, Texas 78712-1188, U.S.A.

Howard A. Blair
School of Computer and Information Science,
313 Link Hall,
Syracuse University, Syracuse, N.Y. 13244, U.S.A.

We study here the recursion theoretic complexity of the perfect (Herbrand) models of stratified logic programs. We show that these models lie arbitrarily high in the arithmetic hierarchy. As a byproduct we obtain a similar characterization of the recursion theoretic complexity of the set of consequences in a number of formalisms for non-monotonic reasoning. We show that under some circumstances this complexity can be brought down to recursiveness and recursive enumerability. To this purpose we study a class of recursion - free programs.

1. Introduction

A substantial amount of the recent research in logic programming concentrated on the "safe" use of negation. This research led to an identification of a subclass of general logic programs, called *stratified* programs, which restrict the ways in which recursion and negation can be combined. Intuitively, the use of negation is restricted by only applying it to already known relations. Thus, in defining a collection of relations some of them are first defined, perhaps recursively in terms of themselves, without the use of negation. New relations may then be defined in terms of themselves without using negation, and in terms of the previously defined relations and their negations. The process can be iterated until all of the relations in the collections have been defined.

Stratified programs were introduced in Apt, Blair and Walker [ABW88] and van Gelder [VG88]. They form a simple generalization of a class of database queries introduced in Chandra and Harel [CH85].

Stratified programs have a natural semantics associated with them in the form of a specific Herbrand model. The special character of these models was captured by Przymusinski [P88] who introduced the concept of *perfect* models. The designated model of a stratified program is its unique perfect Herbrand model. In this paper we study the recursion theoretic complexity of the perfect (Herbrand) models of stratified programs. We show that they lie arbitrarily high in the arithmetic hierarchy. We also show that under certain circumstances their complexity can be brought down to recursiveness and recursive enumerability. To this purpose we study a class of *recursion-free* programs. We prove that Clark's [C78] completions of recursion-free programs together with a first order domain

*) to appear in Fundamenta Informaticae. First version of this paper appeared in Proc. 5th International Conference on Logic Programming, The MIT Press, 1988.

Given an operator T, we define its *powers* by

$$T\uparrow 0(I) = I,$$
$$T\uparrow(n+1)(I) = T(T\uparrow n(I)) \cup T\uparrow n(I),$$
$$T\uparrow\omega(I) = \cup\{T\uparrow n(I) | n<\omega\}.$$

We call an operator T *finitary* if for every infinite sequence

$$I_0 \subseteq I_1 \subseteq \ldots,$$
$$T(\cup\{I_n | n<\omega\}) \subseteq \cup\{T(I_n) | n<\omega\}$$

holds.

We call an operator T *growing* if for all I,J,M

$$I \subseteq J \subseteq M \subseteq T\uparrow\omega(I)$$

implies

$$T(J) \subseteq T(M).$$

Thus "growing" is a restricted form of monotonicity. The following lemma will be needed in Section 5.

LEMMA 2.1. *Let T be a finitary and growing operator. For all A,I and $n \geq 1$,*

$$A \in T\uparrow n(I)$$

iff there exists a finitely branching tree of depth $\leq n$ such that

- *A is its root,*
- *for every node B with direct descendants $B_1, \ldots, B_k$, $k>0$, we have $B \in T(I \cup \{B_1, \ldots, B_k\})$,*
- *every leaf is an element of $T\uparrow 1(I)$.*

PROOF. For all I and $n \geq 1$, $T\uparrow n(I)$ is countable and includes I, so for some sequence $S_0 \subseteq S_1 \subseteq \ldots$ of finite subsets of $T\uparrow n(I)$

$$T\uparrow n(I) = \cup\{I \cup S_k | k<\omega\}.$$

Since T is finitary

$$T(T\uparrow n(I)) \subseteq \cup\{T(I \cup S_k) | k<\omega\}.$$

Also, for $k<\omega$ we have $I \subseteq I \cup S_k \subseteq T\uparrow n(I) \subseteq T\uparrow\omega(I)$, so since T is growing

$$\cup\{T(I \cup S_k) | k<\omega\} \subseteq T(T\uparrow n(I)).$$

Thus

$$T(T\uparrow n(I)) = \cup\{T(I \cup S_k) | k<\omega\}.$$

Hence for all A,I and $n \geq 1$,

$$A \in T(T\uparrow n(I))$$

iff for some $B_1, \ldots, B_k \in T\uparrow n(I)$, $k \geq 0$, we have $A \in T(I \cup \{B_1, \ldots, B_k\})$.

From this the claim follows by a simple induction on n. □

$$M_1 = \bigcap \{M \mid M \text{ is supported model of } P_1\},$$
$$M_2 = \bigcap \{M \mid M \text{ is supported model of } P_2 \text{ and } M \cap B_{P_1} = M_1\},$$
$$\vdots$$
$$M_n = \bigcap \{M \mid M \text{ is supported model of } P_n \text{ and } M \cap B_{P_1 \cup \ldots \cup P_{n-1}} = M_{n-1}\},$$
$$M_P = M_n.$$

iv) *M_P is a model of comp(P),* CLARK's [Cl78] completion of P.

v) *When P has no function symbols, there is a backchaining interpreter for P which combines negation as failure with loop checking to test for membership in M_P. On each inference cycle the interpreter fully instantiates a clause.* □

Other properties of stratified programs were proved in [VG88].

When P is a program, $M_P = T_P \uparrow \omega(\emptyset)$ and M_P coincides with the least Herbrand model of P introduced in VAN EMDEN and KOWALSKI [VEK76].

2.4. Perfect model semantics

Further characterization of the model M_P was provided by PRZYMUSINSKI [P88] who introduced the concept of *perfect* models. The essence of his approach can be summarized as follows.

Consider a general program P. Let $<$ be a well founded ordering on the Herbrand base B_P of P. If $A < B$ then we say that A has a *higher priority* than B.

Let M, N be interpretations of P. We call N *preferable to* M if $M \neq N$ and for every $B \in N \setminus M$ there exists $A \in M \setminus N$ such that $A < B$. We call a model of P *perfect* if no other model of P is preferable to it.

Intuitively, N is preferable to M if it is obtained from M by possibly adding/removing some atoms and an addition of an atom to N is always compensated by the simultaneous removal from M of an atom of higher priority. This reflects the fact that we are determined to minimize higher priority atoms even at the cost of adding atoms of lower priority.

The above definitions are parameterized by the well founded ordering $<$. We now consider a fixed stratified program P and a well founded ordering on B_P obtained by first, putting for two relation symbols

$p < q$ iff there is a path from q to p in D_P with a negative arc,

and then putting for two ground atoms A, B

$A < B$ iff $p < q$ where p appears in A and q appears in B.

Note that if $p < q$, then in any stratification of P, p is defined in a lower stratum than q is. Thus $<$ is well founded. This implies that the latter ordering $<$ is indeed a well founded ordering on B_P. In this ordering ground atoms with a relation symbol from a lower stratum have a higher priority.

The following theorem from [P88] characterizes the model M_P of P.

THEOREM 2.4. *Let P be a stratified program. Then M_P is the unique perfect model of P.* □

3.2. Computability over the Herbrand universe

Our task is to adapt the entire previous discussion of computability over the natural numbers to computability over Herbrand universes. Of course this can be done in one stroke by effectively identifying the ground terms with the natural numbers. However, if we want to characterize what general programs compute in recursion-theoretic terms, the correspondence between the Herbrand universe and $\mathbb{N}$ is delicate. This point can be brought out vividly by reflecting on the following task: write a program P such that for a ground term t, $\leftarrow r(t)$ succeeds iff t is a constant. Note that this cannot be done if, for example, the underlying Herbrand universe contains *infinitely many* constant symbols and infinitely many functions symbols. It follows that if the Herbrand universe is generated by an infinite alphabet then not every computable relation over such a Herbrand universe can be computed by a logic program.

We now analyse what logic programs compute in recursion-theoretic terms under the assumption that the underlying Herbrand universe is finitely generated. We assume a fixed finitely generated Herbrand universe U_L with at least one constant and one function symbol. All general programs P considered in this paper are such that their Herbrand universe U_P coincides with U_L.

A program P *computes* a relation R over U_L *using* a relation symbol r if for all sequences $\bar{t}$ of elements from U_L

$$\bar{t} \in R \text{ iff there exists an SLD-refutation of } P \cup \{\leftarrow r(\bar{t})\}.$$

A program P *defines* a relation R over U_L *using* a relation symbol r if for all sequences $\bar{t}$ of elements from U_L

$$\bar{t} \in R \text{ iff } P \models r(\bar{t}).$$

Here and elsewhere we assume that R and r have the same arity which also coincides with the length of the sequence $\bar{t}$.

The following theorem links computability and definability and the least Herbrand model of a program, and is fundamental in logic programming (cf APT and VAN EMDEN [AVE82]; see also Theorem 4.1 in APT [A]).

THEOREM 3.3. *Let P be a program, R a relation over U_L, and r a relation symbol. Then*

i) *P computes R using r iff P defines R using r.*

ii) *P defines R using r iff for all sequences $\bar{t}$ of elements from U_L,* □

$$\bar{t} \in R \text{ iff } r(\bar{t}) \in M_P.$$

This theorem allows us to identify computability with definability and reduce the latter to definability over the least Herbrand model. Note that this theorem also holds when U_L is finite and nonempty, which arises when U_L consists of a finite set of constants.

The identification of U_L with $\mathbb{N}$ is obtained via the next theorem.

THEOREM 3.4. (Enumeration Theorem) *A program* successor *which defines the successor relation on U_L using the binary relation symbol* succ *can be constructed. More precisely, an ordering $<$ on U_L of order-type ω can be constructed such that for all terms $s,t \in U_L$, t is an $<$- successor of s iff* successor $\models$succ(s,t). □

The enumeration theorem above is due to ANDREKA and NEMETI [AN78]. BLAIR [B86] gives a version in which the *successor* program satisfies additional semantic constraints related to finite failure of goals.

This theorem allows us to identify a finitely generated Herbrand universe U_L of the form assumed at the beginning of this section with natural numbers. This identification allows us to transfer the