Fundamenta Informaticae XIII (1990) 1-17
IOS Press

ARITHMETIC CLASSIFICATION OF PERFECT MODELS OF STRATIFIED PROGRAMS*

Krzysztof R. APT

Centre for Mathematics and Computer Science,
P.O. Box 4079, 1009 AB Amsterdam, Netherlands

Howard A. BLAIR

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

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 recursive enumerability.

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 [ABW87] and VAN GELDER [VG86]. 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 [P87] 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 recursive enumerability.

The recent rise of interest in non-monotonic reasoning led to intensive research of the relative strength and expressive power of the formalisms involved. In this paper we take advantage of this fact by indicating that the results obtained here directly translate into results concerning default logic of REITER [R80], pointwise circumscription of LIFSCHITZ [L86] and Iterated Closed World Assumption of GELFOND, PRZYMUSINSKA and PRZYMUSINSKI [GPP86]. This allows us to assess the recursion theoretic complexity of these formalisms, too.

Our results improve upon an observation of KOLAITIS [K87] who showed that the perfect models of stratified programs are Δ_1^1 relations. Similarly as [CH85], [K87] is mainly concerned with the complexity of perfect models of stratified programs, in the absence of function symbols.

2. PRELIMINARIES

In this section we review the basic results and definitions dealing with stratified programs which form a basis for this paper. All logic programming notation and terminology not defined in this paper may be found in LLOYD [Ll84].

Recall that by a *clause* we mean a construct of the form $A \leftarrow B_1,...,B_n$, where A, $B_1,...,B_n$ $(n \geqslant 0)$ are atoms. A *program* is a finite, non-empty set of clauses.

In turn, by a *general clause* we mean a construct of the form $A \leftarrow L_1,...,L_n$, where A is an atom and $L_1,...,L_n$ $(n \geqslant 0)$ are literals. A *general program* is a finite, non-empty set of general clauses.

2.1. Stratified programs

Given a general program P, we define its *dependency graph* D_P by putting for two relation symbols p,q

$(p,q) \in D_P$ iff there is a general clause in P using p in its head and q in its body.

The arc (p,q) is called *positive* (resp. *negative*) if there is a general clause in P such that p appears in its head and q appears in a positive (resp. negative) literal of its body. Note that an arc may be both positive and negative.

Now, a general program is called *stratified* if in its depending graph D_P there is no cycle with a negative arc.

We say that a relation symbol *occurs negatively* in a general program P, if it appears in a negative literal of a body of a general clause from P. By a *definition* of a relation symbol r (within P) we mean the set of all general clauses of P in whose heads r appears.

An alternative definition of a stratified program is as follows. A general program P is stratified if for some partition

$$P = P_1 \dot{\cup} \ldots \dot{\cup} P_n$$

the following two conditions hold for $i = 1,\ldots,n$:

i) if a relation symbol appears in a positive literal of a general clause from P_i, then its definition is contained within $\cup\{P_j | j \leq i\}$,

ii) if a relation symbol appears in a negative literal of a general clause from P_i, then its definition is contained within $\cup\{P_j | j < i\}$.

We allow P_1 to be empty. A head of a general clause is viewed here as one of its positive literals. We call each P_i a *stratum*. Note that the definition of any relation symbol is either empty or a subset of exactly one stratum.

To study the semantics of stratified programs we first discuss operators on complete lattices.

2.2. Finitary and growing operators

Consider an arbitrary but fixed, non-empty, countable set. We denote its elements by A,B. Its subsets form a complete lattice with the order relation $\subseteq$, the least upper bound operator $\cup$ and the greatest lower bound operator $\cap$. We denote its elements by I,J,M. We now consider operators on this lattice.

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 3.

LEMMA 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*

$B \in T(I \cup \{B_1,...,B_k\})$,

- *every leaf is an element of* $T\uparrow 1(I)$.

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

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

Since T is finitary and growing

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

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

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

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

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

2.3. Semantics of stratified programs

We now summarize the notions and results of [ABW87]. Given a general program P, we denote by ground (P) the set of all ground instances of general clauses of P. To avoid some uninteresting complications we assume that ground (P) is always non-empty. Consider now the complete lattice consisting of all subsets of the Herbrand base B_P of P. These subsets are in the sequel identified with Herbrand interpretations of P. We only study here Herbrand interpretations and models, so we drop the qualification "Herbrand".

Given a general program P and an interpretation M of P, we put

$$T_P(M) = \{A \mid \text{for some literals } L_1,...,L_n$$
$$A \leftarrow L_1,...,L_n \text{ is in ground } (P)$$
$$\text{and } M \models L_1 \wedge ... \wedge L_n\}.$$

We call a general program P *semi-positive,* if no relation symbol which appears in a head of a general clause of P, also appears negatively in P.

The following lemma summarizes the results we shall need in the sequel.

LEMMA 2:

i) *For a general program* P, T_P *is finitary.*

ii) *For a semi-positive program* P, T_P *is growing.*

iii) *A stratum of a stratified program is semi-positive.* □

Thus for a stratum P of a stratified program, we can use lemma 1 to characterize the sets $T_P\uparrow n(I)$.

Consider now a stratified program P with a stratification

$$P = P_1 \dot{\cup} ... \dot{\cup} P_n.$$

We assign to P a Herbrand model M_P by putting

$$M_1 = T_{P_1}\uparrow\omega(\varnothing),$$
$$M_2 = T_{P_2}\uparrow\omega(M_1),$$
$$\vdots$$
$$M_n = T_{P_n}\uparrow\omega(M_{n-1})$$

and letting

$$M_P = M_n.$$

M_P is called in [ABW87] the *standard model* of P.

Some general results on non-monotonic operators on complete lattices, like lemma 2, were established in [ABW87], to prove the properties of stratified programs and their standard models that are listed in the following theorem. In the theorem, a *supported* model M has the property that if ground atom A is true in M, then there is a ground instance $A \leftarrow L_1,\ldots,L_n$ of a clause in P such that $L_1 \wedge \ldots \wedge L_n$ is true in M. $L_1 \wedge \ldots \wedge L_n$ can then be viewed as an *explanation for* A. Thus in a supported model every true ground atom has an explanation.

THEOREM 3: *Let P be a stratified program. Then:*

i) *M_P is independent of the stratification of P.*

ii) *M_P is a minimal supported model of P.*

iii) *There is an alternative definition of M_P that uses iteratively smallest models as follows:*

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

iv) *M_P is a model of comp(P),* CLARK's [C178] 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 [VG86].

When P is a program, $M_P = T_P\uparrow\omega(\varnothing)$ 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 [P87] 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

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

and then putting for two ground atoms A,B

$$A < B \text{ iff } p < q \text{ where } p \text{ appears in } A \text{ and } q \text{ appears in } B.$$

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 [P87] characterizes the model M_P of P.

THEOREM 4: *Let P be a stratified program. Then M_P is the unique perfect model of P.* □

3. COMPUTABILITY

3.1. Preliminaries

The results given in the next section are based on a recursion-theoretic characterization of the relations computable by logic programs. We recall here the basic concepts of recursion theory. We assume the reader is familiar with the inductive definition of (total) recursive functions over the natural numbers, $\mathbb{N}$, obtained by closing a set of basic functions by composition and application of minimization under certain totality conditions; see, for example rules **R1, R2** and **R3** in SHOENFIELD [Sh67, chapter 6]. By removing the restriction on when minimization is applicable the partial recursive functions are obtained. A relation over $\mathbb{N}$ is recursive iff its characteristic function is recursive. (Our usage of the term *relation* differs from that of Shoenfield).

We can relativize the total recursive functions by adding new functions to the set of basic functions from which we previously obtained the rest of the recursive functions. If F is a set of functions, let $Rec(F)$ be set of functions obtained in this way. The functions in $Rec(F)$ are said to be *recursive in F*. A relation R is recursive in a set of relations C iff the characteristic function of R is recursive in the set of characteristic functions of relations in C. The arithmetic hierarchy is defined as follows.

Here and elsewhere $\overline{m}$ stands for a sequence of natural numbers. Similar convention is used for terms and variables.

Σ_0^0 is the of all relations whose characteristic functions are in $Rec(\varnothing)$, which is the set of all recursive relations.

Π_n^0 is the set of all relations whose complement (with respect to $\mathbb{N}$) is in Σ_n^0.

Σ_{n+1}^0 is the set of all relations R satisfying

$$\overline{m} \in R \text{ iff } \exists n[(\overline{m},n) \in Q] \qquad (\dagger)$$

for some Q in Π_n^0.

In general, if R is defined via an equivalence of the form given in (†) and Q is *recursive in* a set of relations C, then R is said to be *recursively enumerable in C*. Note that $\Pi_0^0 = \Sigma_0^0$, and that the familiar recursively enumerable relations are just those relations recursively enumerable in Π_0^0.

Now, a relation R is (*many-one*) *complete* for a class of relations C iff $R \in C$ and for each relation $Q \in C$ there is a total recursive function f such that

$$\overline{m} \in Q \text{ iff } f(\overline{m}) \in R.$$

Intuitively, R is representative of the hardest decision problem in C. (One may note that the distinction between Turing completeness and many-one completeness is immaterial for our results in the next section.)

LEMMA 5: *A relation R is in Σ_{n+1}^0 iff R is recursively enumerable in Π_n^0.* □

The preceding lemma is less trivial than it may seem since for R to be recursively enumerable in Π_n^0 there must be a relation Q which is *recursive in* Π_n^0 such that (†), but this does not mean that Q itself need be in Π_n^0. However Q is recursive in Π_n^0 iff Q is recursive in Σ_n^0. Thus

COROLLARY 6: *A relation R is in Σ_{n+1}^0 iff R is recursively enumerable in Σ_n^0.* □

3.2. Computability over 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 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 7: *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,*

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

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 8: (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 [B186] 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 notions of recursion theory from $\mathbb{N}$ to U_L, and subsequently from U_L to B_P. Our subsequent investigations rely on this transfer.

The following lemma due to ANDREKA and NEMETI [AN] (see Corollary 4.5 in [A]) connects the notion of definability by programs with the recursion theoretic concepts.

LEMMA 9: *A relation R on U_L is recursively enumerable iff some program P defines R using a relation symbol r.* □

3.3. Computability by programs

We start our investigations with the following lemma which strictly speaking, is not needed to prove our main results. However, it is interesting in itself.

LEMMA 10: *For a program P, the relation $\{(n,A) | A \in T_P \uparrow n(\varnothing), n < \omega\}$ is recursive.*

PROOF. Following WOLFRAM, MAHER and LASSEZ [WML], by a *BF-derivation* of $P' \cup \{N\}$ for a program P' and a goal N we mean a refinement of the usual SLD-derivation in which in each goal *all* atoms are selected. (*BF* stands for *B*readth-*F*irst.) If the last goal is empty, such a derivation is called a *refutation.*

Now,

$A \in T_P \uparrow n(\varnothing)$ iff there is a BF-refutation

of ground$(P) \cup \{\leftarrow A\}$

of length at most n.

However, by a lifting lemma for BF-resolution, proved in [WML], in fact the following equivalence holds:

$A \in T_P \uparrow n(\varnothing)$ iff there is a BF-refutation

of $P \cup \{\leftarrow A\}$

of length at most n.

But the relation

$$\{(A,n,\xi) | \xi \text{ is a BF-refutation of } P \cup \{\leftarrow A\} \text{ of length at most } n\}$$

is recursive. Moreover, ignoring the choice of variables in goals and mgu's, there are only finitely many BF-refutations of $P \cup \{\leftarrow A\}$ of length at most n. This proves the claim. □

COROLLARY 11: *For a program P, $T_P \uparrow \omega(\varnothing)$ is recursively enumerable.* □

Perhaps surprisingly, lemma 10 does *not* relativize. Indeed, for a program P, $T_P \uparrow n(M)$ is not recursive in M. To see this, note that $T_P(M)$ need not be recursive in M.

EXAMPLE 12: Let Q be a recursively enumerable, non-recursive, subset of U_L. For some recursive relation R

$$s \in Q \text{ iff } \exists t[(s,t) \in R].$$

Let P be the program

$$q(X) \leftarrow r(X,Y),$$

and let $M = \{r(s,t) | (s,t) \in R\}$. Then $T_P(M) = \{q(s) | s \in Q\}$. M is recursive; $T_P(M)$ is not. □

3.4. Computability by semi-positive programs

However, $T_P \uparrow n(M)$ is recursively enumerable in M. This holds for semi-positive programs, as well. We need this fact later; to establish it we first need the following observation. Here, $<B_1,...,B_k>$ stands for a natural number associated with the sequence of atoms $B_1,...,B_k$ in a standard way (see [Sh67, chapter 6]).

LEMMA 13: *For a general program P, the relation*

$$\{(A, <B_1,...,B_k>) | A \in T_P(M \cup \{B_1,...,B_k\})\}$$

is recursively enumerable in M.

PROOF. Direct, by the definition of T_P and the standard techniques of recursion theory. □

We can now prove the desired lemma.

LEMMA 14: *For a semi-positive program P, the relation $\{(n,A) | A \in T_P \uparrow n(M), n < \omega\}$ is recursively enumerable in M.*

PROOF. Thanks to lemma 2 we can use lemma 1 to characterize the relation in question. This characterization implies by lemma 13 and the standard techniques of recursion theory, that this relation is indeed recursively enumerable in M. □
The following generalizes corollary 11.

COROLLARY 15: *For a semi-positive program P, the relation $T_P\uparrow\omega(M)$ is recursively enumerable in M.* □

For an interpretation M and a relation symbol r, let

$$M|r = \{A | A \in M \text{ and the relation symbol of } A \text{ is } r\}.$$

We say that an interpretation M of P is *strongly recursively enumerable,* (or *strongly R.E.,* in short) if M is recursively enumerable and for each relation symbol r which appears negatively in P, $M|r$ is recursive.

We now show that under some circumstances the relations studied in lemmata 13 and 14 and corollary 15 can be characterized in a more precise way.

LEMMA 16: *Consider a general program P and an interpretation M. Suppose that M is strongly R.E. Then $T_P(M)$ is recursively enumerable.*

PROOF. We have for all ground atoms A

$$A \in T_P(M)$$

iff for some literals $L_1,...,L_n$
i) $A \leftarrow L_1,...,L_n$ is in ground (P),
ii) for every positive literal B from $L_1,...,L_n$ we have $B \in M$,
iii) for every negative literal $\neg B$ from $L_1,...,L_n$ whose relation symbol is r, we have $B \notin M|r$.

Now by the standard techniques of recursion theory, $T_P(M)$ is indeed recursively enumerable. □

LEMMA 17: *Consider a semi-positive program P and an interpretation M. Suppose that M is strongly R.E. Then the relation $\{(n,A) | A \in T_P\uparrow n(M), n < \omega\}$ is recursively enumerable.*

PROOF. Analogous to the proof of lemma 14 but using lemma 16 instead of lemma 13.

COROLLARY 18: *Consider a semi-positive program P and an interpretation M. Suppose that M is strongly R.E. Then the relation $T_P\uparrow\omega(M)$ is recursively enumerable.* □

4. Arithmetic classification of M_P

We are now ready to prove the main results of the paper.

THEOREM 19: *If P is a stratified program with n strata, then M_P is Σ^0_n.*

PROOF. We proceed by induction on n. If $n=1$, then P is a program and the theorem follows from corollary 11.

Now suppose the statement of the theorem holds for $n-1$, and P is stratified by $P_1 \dot\cup \ldots \dot\cup P_n$. We have $M_P = T_{P_n}\uparrow\omega(M_{P_1 \cup \ldots \cup P_{n-1}})$, so by corollary 15 and lemma 2 iii) M_P is recursively enumerable in $M_{P_1 \cup \ldots \cup P_{n-1}}$. By the induction hypothesis, $M_{P_1 \cup \ldots \cup P_{n-1}}$ is Σ^0_{n-1}. Therefore by corollary 6, M_P is Σ^0_n. □

THEOREM 20: *Let P be a stratified program. Suppose that for each relation symbol r which occurs negatively in P, $M_P|r$ is recursive. Then M_P is recursively enumerable.*

PROOF. Consider a stratification $P_1 \dot\cup \ldots \dot\cup P_n$ of P with the corresponding sequence of models $M_1,\ldots,M_n$ with $M_P = M_n$. We prove by induction on $i=1,\ldots,n$ that each M_i is recursively enumerable.

For $i=1$ it is the content of corollary 11. Assume the claim holds for some i, $1 \leqslant i < n$.

Consider a relation symbol r which occurs negatively in P_{i+1}. Then the definition of r is contained in $\cup\{P_j | j \leqslant i\}$, so $M_P|r = M_i|r$. By assumption, for every r which occurs negatively in P_{i+1}, $M_i|r$ is recursive. Thus by lemma 2 iii) and corollary 18 applied to P_{i+1} and M_i, M_{i+1} is recursively enumerable. □

Of course, it is in general not clear how to check that for a relation symbol r and an interpretation M, $M|r$ is recursive. However, in some situations this is obvious - when r is defined by enumeration, i.e. exclusively by a list of unit clauses. Then for every such r, $M_P|r$ is recursive.

Call a general program *strongly stratified* if each relation symbol which occurs negatively in P is defined exclusively by unit clauses. Obviously, every strongly stratified program is stratified. By the above observation and theorem 20 we have:

COROLLARY 21: *Let P be a strongly stratified program. Then M_P is recursively enumerable.* □

Finally, we prove the following:

THEOREM 22: *For each $n \geqslant 1$ there is a stratified program P with n strata for which M_P is Σ^0_n-complete.*

PROOF. We prove the following stronger claim from which the theorem

follows by choosing R to be Σ_n^0-complete: for each Σ_n^0 relation R over U_L we can find a stratified program P with n strata such that for some relation symbol r

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

We now proceed by induction on n.

For $n=1$ the claim is a consequence of lemma 9 and theorem 7 ii). Now assume the claim holds for a particular $n \geq 1$. Let R be a Σ_{n+1}^0 relation over U_L. For some Π_n^0 relation S over U_L

$$\bar{s} \in R \text{ iff } \exists t[(\bar{s},t) \in S].$$

Let Q be the complement of S in U_L. Q is Σ_n^0. By the induction hypothesis we can find a stratified program P with n strata such that for some relation symbol q

$$(\bar{s},t) \in Q \text{ iff } q(\bar{s},t) \in M_P.$$

We now add to P two clauses defining R in terms of S and S in terms of Q. Let P_{n+1} consist of the clauses

$$p_R(\bar{X}) \leftarrow p_S(\bar{X}, Y),$$
$$p_S(\bar{X}, Y) \leftarrow \neg q(\bar{X}, Y)$$

where p_R and p_S are relation symbols not occurring in P. Let $P' = P \cup P_{n+1}$. Then

$$\begin{aligned} M_{P'} &= M_P \cup \{p_R(\bar{s}) | \exists t[(\bar{s},t) \in S]\} \cup \{p_S(\bar{s},t) | (\bar{s},t) \notin Q\} \\ &= M_P \cup \{p_R(\bar{s}) | \bar{s} \in R\} \cup \{p_S(\bar{s},t) | (\bar{s},t) \in S\}. \end{aligned}$$

Thus,

$$\bar{s} \in R \text{ iff } p_R(\bar{s}) \in M_P. \quad \square$$

5. Applications to non-monotonic reasoning

We now relate our results to three formalisms commonly used in the area on non-monotonic reasoning. We follow here their description given in Przymusinski [P87].

5.1. Default logic

One of them is *default logic* introduced in [R80]. In default logic, apart of the usual rules of first order logic, also *default rules* are used. They have the form

$$\frac{B: MC_1,..,MC_n}{A}$$

where A, B, $C_1,..,C_n$ are first order formulas. Such a rule intuitively means: "if B holds and each of C_i-s can be (separately) consistently assumed, then

conclude A". The usual rules and the default rules induce a natural concept of an *extension* of a set of first order formulas. We omit here its formal definition. This extension, if it is unique, denotes the set of consequences of a set of formulas under the default rules.

PRZYMUSINSKA [Pa87] related general programs to default logic by noting that a general clause $A \leftarrow A_1,...,A_m, \neg B_1,..,\neg B_n$ where $n>0$ naturally translates into a default rule

$$\frac{A_1 \wedge ... \wedge A_m : M\neg B_1,..,M\neg B_n}{A}$$

Given a general program P, let T denote the set of (positive) clauses of P and let D_P denote the set of default rules obtained by the above translation. PRZYMUSINSKA [Pa87] showed that given a stratified program P, the default rules in D_P induce a unique extension $D_P(T)$ of T which coincides with the set of formulas true in the perfect model of P.

By theorems 19, 20 and 22 we immediately obtain

COROLLARY 23:

i) *Let P be a stratified program with n strata. Then $D_P(T)$ is Σ^0_n.*
ii) *Let P be a strongly stratified program Then $D_P(T)$ is Σ^0_1.*
iii) *For each $n \geqslant 1$ there is a default theory whose set of consequences is Σ^0_n-complete.* □

5.2. Circumscription

Another approach to non-monotonic reasoning is based on the circumscription method of McCarthy. We discuss here its variant called *prioritized circumscription* described in [MC86].

Let $\phi(R,Q)$ be a first order formula whose relation symbols appear in $R=\{r_1,...,r_m\}$ or $Q=\{q_1,...,q_n\}$, where $R \cap Q = \varnothing$, and let $R'=\{r'_1,...,r'_m\}$ and $Q'=\{q'_1,...,q'_n\}$ be sets of relation symbols of the same arities as those in R and Q, correspondingly. By a *parallel circumscription* of R in ϕ with variables Q we mean the following second order formula $CIRC(\phi;R;Q)$:

$$\phi(R,Q) \wedge \forall R',Q'[\phi(R',Q') \wedge (R' \rightarrow R) \rightarrow R'=R],$$

where $R' \rightarrow R$ stands for

$$\bigwedge_{i=1}^{m} \forall \bar{x}(r'_i(\bar{x}) \rightarrow r_i(\bar{x}))$$

and $R'=R$ stands for

$$\bigwedge_{i=1}^{m} \forall \bar{x}(r'_i(\bar{x}) \leftrightarrow r_i(\bar{x})).$$

Intuitively, $CIRC(\phi;R;Q)$ states that relation symbols from R are minimal under the assumption that $\exists Q'\phi(R,Q')$ holds and moreover, $\phi(R,Q)$ does hold.

Now, consider disjoint sets of relation symbols $R_1,...,R_n$. By a *prioritized*

circumscription of a second order formula ϕ with priorities $R_1 > ... > R_n$ we mean the following second order formula $CIRC(\phi, R_1 > ... > R_n)$:

$$CIRC(\phi; R_1; \{R_2 \cup ... \cup R_k\}) \wedge CIRC(\phi; R_2; \{R_3 \cup ... \cup R_n\}) \wedge ... \wedge CIRC(\phi; R_n; \varnothing)$$

Intuitively, this formula states that the relation symbols in $R_1,...,R_n$ are minimized in a particular order given by the priorities $R_1 > ... > R_n$.

Denote the set of first order formulas implied by a second formula ϕ by $Cn(\phi)$. Consider now a stratified program P with a stratification $P_1 \cup ... \cup P_n$. Let $R_1,...,R_n$ be the sets of relation symbols defined in $P_1,...,P_n$, respectively. After an identification of P with a conjunction of its general clauses, P can be viewed as a second order formula whose relation symbols are those in $R_1,...,R_k$.

LIFSCHITZ [L87] showed that the set of formulas $Cn(CIRC(P, R_1 > ... > R_k))$ coincides with the set of formulas true in the perfect model of P.

Again, by theorems 19, 20 and 22 we obtain

COROLLARY 24: *Let P be a stratified program with a stratification $P_1 \cup ... \cup P_n$. Let $R_1,...,R_n$ be the sets of relation symbols defined in $P_1,...,P_n$, respectively.*

i) *$Cn(CIRC(P, R_1 > ... > R_n))$ is Σ_n^0.*
ii) *If P is strongly stratified, then $Cn(CIRC(P, R_1 > ... > R_n))$ is Σ_1^0.*
iii) *For each $n \geqslant 1$ there is a second order formula ϕ with disjoint sets of relation symbols $Q_1,..,Q_n$ such that $Cn(CIRC(\phi, Q_1 > ... > Q_n))$ is Σ_n^0-complete.* □

5.3. Iterated closed world assumption

Finally, we consider the Iterated Closed World Assumption (ICWA) introduced in [GPP86]. ICWA is a generalization of the Closed World Assumption of REITER [R78] (CWA).

Given a set of (first order) formulas P we define first

$$CWA(P) = P \cup \{\neg A \mid A \text{ is a ground atom such that } P \not\vdash A \text{ does not hold}\}$$

[R78] showed that for a program P, $CWA(P)$ is consistent. Unfortunately, this result does not hold for a general program P. To resolve this problem [GPP86] concentrated on the case of stratified programs.

Consider a stratified program P with a stratification $P_1 \cup ... \cup P_n$. We define

$$ICWA(P_1) = CWA(P_1),$$
$$ICWA(P_{i+1}) = CWA(P_{i+1} \cup ICWA(P_i)) \quad \text{for } 1 \leqslant i < n,$$
$$ICWA(P) = ICWA(P_n).$$

[GPP86] showed that for a stratified program P, $ICWA(P)$ has exactly one

model, namely the perfect model of P.

By theorems 19, 20 and 22 we obtain

COROLLARY 25:

i) *Let P be a stratified program with n strata. Then $ICWA(P)$ is Σ_n^0.*

ii) *Let P be a strongly stratified program. Then $ICWA(P)$ is Σ_1^0.*

iii) *For each $n \geqslant 1$ there is a stratified program with n strata such that $ICWA(P)$ is Σ_n^0-complete.* □

For every reasoning method it is preferable from the logic point of view that the set of consequences obtained by it is decidable (recursive) or semi-decidable (recursively enumerable). We showed here that this is not the case for a majority of commonly used formalisms in the area of non-monotonic reasoning. However, we also indicated a reasonable restriction - to strongly stratified programs, which allows us to bring down this complexity to recursive enumerability.

ACKNOWLEDGEMENT

We would like to thank Marc Bezem for helpful comments and Ms Caroline Swagerman for speedy typing of the manuscript.

REFERENCES

[AN78] H. ANDREKA and I. NEMETI, The Generalised Completeness of Horn Predicate Logic as a Programming Language, Acta Cybernetica, vol. 4, no. 1, 1978, pp. 3-10.

[A87] K.R. APT, Introduction to Logic Programming, Centre for Mathematics and Computer Science, Amsterdam, Technical Report CS-R8741, 1987 (to appear in Handbook of Theoretical Computer Science (J. van Leeuwen, Managing Editor)).

[ABW87] K.R. APT, R. BLAIR and A. WALKER, Towards a Theory of Declarative Knowledge, in: *Foundations of Deductive Databases and Logic Programming,* (J. Minker, ed.), Morgan-Kaufmann, Los Altos, CA., 1987.

[AVE82] K.R. APT and M.H. VAN EMDEN, Contributions to the Theory of Logic Programming, *JACM, vol. 29, No. 3,* 1982, pp. 841-862.

[Bl86] H.A. BLAIR, Decidability in the Herbrand Base, *Workshop on Foundations of Deductive Databases and Logic Programming,* Washington, D.C., manuscript, 1986.

[CH85] A. CHANDRA and D. HAREL, Horn Clause Queries and Generalizations, Journal of Logic Programming, vol. 2, no. 1, 1985, pp. 1-15.

[Cl78] K.L. CLARK, Negation as Failure, in: *Logic and Databases,* (H. Gallaire and J. Minker, eds.), Plenum Press, New York, 1978, pp. 293-322.

[GPP86] M. GELFOND, T. PRZYMUSINSKI and H. PRZYMUSINSKA, On the Relationship between Circumscription and Negation as

Failure, to appear in Journal of Artificial Intelligence.

[K87] P.G. KOLAITIS, The Expressive Power of Stratified Logic Programs, manuscript, Nov. 1987.

[L87] V. LIFSCHITZ, On the Declarative Semantics of Logic Programs with Negation, in: *Foundations of Deductive Databases and Logic Programming* (J. Minker, ed.), Morgan Kaufmann, Los Altos, C.A., 1987.

[Ll84] J.W. LLOYD, *Foundations of Logic Programming,* Springer-Verlag, 1984.

[MC86] J. MCCARTHY, Applications of Circumscription to Formalizing Common Sense Knowledge, Journal of Artificial Intelligence, vol. 28, 1986, pp. 89-116.

[P86] T. PRZYMUSINSKI, On the Declarative and Procedural Semantics of Logic Programs, in: *Foundations of Deductive Databases and Logic Programming* (J. Minker, ed.), Morgan Kaufmann, Los Altos, C.A., 1987.

[P87] T. PRZYMUSINSKI, Non-monotonic Reasoning vs. Logic Programming: A New Perspective, to appear in Handbook on the Formal Foundations of A.I. (Y. Wilks and D. Patridge, eds.)

[Pa87] H. PRZYMUSINSKA, personal communication.

[R78] R. REITER, On Closed-World Data Bases, in: Logic and Databases, (H. Gallaire and J. Minker, eds.), Plenum Press, New York, 1978, pp. 55-76.

[R80] R. REITER, A Logic for Default Theory, Jornal of Artificial Intelligence, vol. 13, 1980, pp. 81-132.

[Sh67] J. SHOENFIELD, *Mathematical Logic,* Addison-Wesley, Reading, Mass. 1967.

[VEK76] M.H. VAN EMDEN and R.A. KOWALSKI, The Semantics of Predicate Logic as a Programming Language, *JACM,* vol. 23, no. 4, 1976, pp. 733-742.

[VG86] A. VAN GELDER, Negation as Failure using Tight Derivations for General Logic Programs in: *Proc. of the 3rd IEEE Symposium on Logic Programming,* Salt Lake City, Utah, 1986.

[WGM] D.A. WOLFRAM, M.J. MAHER and J.-L. LASSEZ, A Unified Treatment of Resolution Strategies for Logic Programs, in: *Proc. of the Second International Conference on Logic Programming, 1984, pp. 263-276.*