INFORMATION AND CONTROL 54, 25–47 (1982)

# The Recursion-Theoretic Complexity of the Semantics of Predicate Logic as a Programming Language*

HOWARD A. BLAIR

*Iowa State University, Ames, Iowa 50010*

## 1. INTRODUCTION

The coupling of resolution techniques for automatic theorem proving with what is termed the procedural interpretation of logic (Kowalski 1979) has resulted in efforts to implement predicate logic as a programming language. These efforts have already resulted in the language PROLOG. The semantics of predicate logic as a programming language have been formulated with an orientation toward the precedural interpretation of logic in van Emden and Kowalski (1976) and Apt and van Emden (1980). The predicate logic core of PROLOG is a restriction to universally quantified Horn sentences with atomic conclusions, so-called definite clauses, and therefore prevents the user from fully expressing logical negation. In the procedural interpretation of logic, a logic program is regarded as its "if and only if" version of which half is explicitly presented. The precise definition of this "if and only if" version is given in Clark (1980). The interpretations of a logic program are then restricted to the Herbrand models of the "if and only if" version of the program, and the formula $F$ is Herbrand valid in logic program $P$ iff $F$ is valid in all such Herbrand interpretations of $P$.

Although the user does not have negation available the control features of PROLOG do allow the user to have a useful but still limited form of negation: to infer $\neg A$ from a proof of the unprovability of $A$. This is sound, as we shall see, in the procedural interpretation of logic. Our purpose here is to use a connection between the semantics of predicate logic as a programming language and the well-studied theory of inductive definability to given a measure to the incompleteness of the negation as failure rule for proving Herbrand valid negations of formulas, and then to show that the negation as failure rule is very highly incomplete in the sense of the measure. Moreover, the general problem of deciding whether a formula is Herbrand

* This article is part of the NSF-sponsored Workshop on Recursion Theoretic Aspects of Computer Science held at Purdue University in May 1981.



0019-9958/82 $2.00

valid in logic program $P$ is of maximum intractability, being $\Pi_1^1$-complete. With little effort this generalizes to saying that the problem of deciding whether a formula is valid in all Herbrand models of a recursively axiomatized first order theory is $\Pi_1^1$-complete.

## 2. Preliminaries

A *closure* of a formula $A$ of predicate logic is a formula obtained from $A$ by prefixing to $A$ a sequence of universally quantified variables which have among them all of the free variables of $A$. A *clause* is a universal closure of a formula of the form

$$A_1 \& \cdots \& A_n \to B_1 \vee \cdots \vee B_m \tag{2.1}$$

for $m, n \geqslant 0$, where each $A_i$ and $B_j$ is atomic. If $n = 0$, (2.1) is simply a disjunction of atomic formulas. If $m = 0$, (2.1) is the negation of a conjunction of atomic formulas. We point out, since the mistake is frequently made, that the clause

$$\forall x_1 \cdots \forall x_k(A_1 \& \cdots \& A_n \to)$$

is not

$$\neg \forall x_1 \cdots \forall x_k(A_1 \& \cdots \& A_m).$$

It is, however, immediately equivalent to

$$\neg \exists x_1 \cdots \exists x_k(A_1 \& \cdots \& A_n).$$

If $m = n = 0$, (2.1) is the empty clause and is defined to be false. Equation (2.1) itself is the matrix of the corresponding clause, and it is the matrix of a *Horn clause* if $m \leqslant 1$, of a *definite clause* if $m = 1$, and of a *negative clause* if $n > 0$ and $m = 0$.

A *logic program* is a set of definite clauses. (See above.) When we write logic programs we omit the universal quantifiers, writing the programs in an abbreviated form. Let $P$ be a logic program. The language of $P$ is the smallest first order language without logical equality that contains the formulas of $P$. Denote the language of $P$ by $L(P)$. The *Herbrand universe* of $P$ is the set of all variable-free terms in $L(P)$. The *Herbrand base* of $P$ is the set of all variable-free atomic formulas in $L(P)$.

Now let $L$ be a first order language with, or without, logical equality. A *Herbrand structure* $I$ for $L$ is any subset of the set of all variable-free atomic formulas of $L$ in which the logical equality symbol does not occur. A Herbrand structure for $L$ is a structure $I$ for $L$ in the ordinary sense where each constant and function symbol is given its free interpretation, and the interpretations of the relation symbols are determined by membership in $I$.

Validity in $I$ is defined as validity in the corresponding ordinary structure. For example, if $I$ is a Herbrand structure for $L$, $B(x)$ is atomic, and $A$ is atomic and variable-free, then

$$I \models A \qquad \text{iff} \quad A \in I$$

and

$$I \models \exists x B(x) \qquad \text{if} \quad I \models B(t)$$

for some variable-free term $t$ in $L$. This last point about validity in Herbrand structures amounts to restricting the class of structures being considered to only those structures which only contain individuals named by some variable-free term. If $t_1 = t_2$ is a closed formula of $L$, then

$$I \models t_1 = t_2 \text{ if } t_1 \text{ is, identically, } t_2.$$

We restrict all considerations of logic programs to those programs which satisfy two technical assumptions. This does not result in any substantive restriction and has no bearing on the theorems of Section 5. We introduce these assumptions only because they are needed in the proofs of our statements of the lemmas proved in Apt and van Emden (1980).

(1) There is at least one constant symbol in $L(P)$.

(2) Every relation symbol which occurs in the hypothesis of some clause in $P$ also occurs in the conclusion of some clause in $P$.

In Clark (1980) a first order theory called the completed data base of $P$ is defined. This is the "if and only if" version of $P$ which we denote CDB($P$). The CDB($P$) can be explained as follows: In the procedural interpretation of a logic program each Horn clause is thought of as a definition of a procedure. That is, the Horn clause

$$A_1(u_1,..., u_k) \,\&\, \cdots \,\&\, A_m(u_1,..., u_k) \rightarrow B(u_1,..., u_k)$$

is interpreted as: to satisfy $B(u_1,..., u_k)$ try satisfying each $A_1(u_1,..., u_k)$,..., $A_1(u_1,..., u_k)$. It is well established in the logic programming literature (cf. van Emden and Kowalski, 1976) that this notion can be viewed as giving a definition of procedure $B(u_1,..., u_k)$ by the sequence of procedures $A_1(u_1,..., u_k)$,..., $A_1(u_1,..., u_k)$. (No commitment is made to a particular order of evaluation.) Now, of course it is possible, indeed likely, that in a logic program $P$ there are several Horn clauses with the same relation symbol in the conclusion. For example, suppose the following are all the Horn clauses in $P$ in which $B$ occurs in the conclusion.

$$\begin{aligned} A_{1,1} \,\&\, \cdots \,\&\, A_{1,n_1} &\to B(u_{1,1},\ldots, u_{1,k}) \\ &\vdots \\ A_{m,1} \,\&\, \cdots \,\&\, A_{m,n_m} &\to B(u_{m,1},\ldots, u_{m,k}). \end{aligned} \tag{2.2}$$

The procedural interpretation demands that $B$ be entirely defined by program $P$. Thus $P$ is thought of as being equipped with the extra assumption that the only way to satisfy $B(x_1,\ldots, x_k)$ is to satisfy one of the clauses in (2.2).

The CDB($P$) captures this idea in the following way: For each relation symbol $B$ (where $B$ is $m$-ary) occurring in a conclusion of some member of $P$, let (2.2) be the set of all clauses in $P$ in which $B$ occurs in the conclusion. Corresponding to $B$, including in CDB($P$) the following formula:

$$\forall x_1 \cdots \forall x_k [E_1 \vee \cdots \vee E_m \leftrightarrow B(x_1,\ldots, x_k)]$$

where each $E_i$ is of the form

$$\exists y_{i,1} \cdots \exists y_{i,p_i} [x_1 = u_{i,1} \,\&\, \cdots \,\&\, x_k = u_{i,k} \,\&\, A_{i,1} \,\&\, \cdots \,\&\, A_{i,n_i}],$$

where "=" is logical equality, the $y_{i,j}$ are the free variables occurring in $u_{i,1},\ldots, u_{i,k}$, and $x_1,\ldots, x_k$ are new. The CDB($P$) is the first order theory with these formulas as nonlogical axioms together with nonlogical axioms forcing a free interpretation on the terms in any structure which is a model of CDB($P$).

As an example, suppose $P$ is

$$\{A(y) \,\&\, B(f(y)) \to A(f(y)), B(y) \to A(g(y)), A(y) \to B(g(y))\}.$$

Then the nonlogical axioms obtained from $P$ in CDB($P$) are

$$\forall x [\exists y [x = f(y) \,\&\, A(y) \,\&\, B(f(y))] \vee \exists y [x = g(y) \,\&\, B(y)] \leftrightarrow A(x)]$$

and

$$\forall x [\exists y [x = g(y) \,\&\, A(y)] \leftrightarrow B(x)].$$

DEFINITION 2.1. Let $P$ be a logic program and let $A$ be a formula of $L(P)$. Formula $A$ is said to be Herbrand valid in $P$ if $A$ is valid in all Herbrand models of CDB($P$).

From logic program $P$, formulas of the form

$$\exists x_1 \cdots \exists x_k [A_1 \,\&\, \cdots \,\&\, A_m], \tag{2.3}$$

where the $A_i$ are atomic, can be proved by linear resolution. A far less efficient means of proving formulas of the form (2.3) is to completely instantiate the logic program in all possible ways over the Herbrand universe and

enumerate all possible sequences of inferences from these instantiations using only modus ponens and the rule: from $A_1,\ldots,A_n$ infer $A_1 \& \cdots \& A_n$. While this is not computationally appropriate this approach does provide a characterization of the Herbrand models of CDB($P$).

DEFINITION 2.2. Let $P$ be a logic program. Define $T_P$ on the Herbrand structures for $L(P)$ by

$$A \in T_P(I) \text{ iff there is a clause } B_1 \& \cdots \& B_n \to B_0 \text{ in } P, \text{ and a substitution } \Theta \text{ such that } A \text{ is } B_0\Theta \text{ and } B_1\Theta,\ldots,B_n\Theta \in I.$$

Definition 2.2 is introduced in van Emden and Kowalski (1976). Here $T_P$ corresponds to completely instantiating $P$ in all possible ways over the Herbrand universe of $P$, and applying modus ponens once to these instantiations and the conjunctions of formulas in $I$. The following three lemmas are proved in Apt and van Emden (1980):

LEMMA 2.1. *$T_P(I) = I$ iff $I$ is a model of* CDB($P$).

LEMMA 2.2. *$\bigcap \{I \mid T_P(I) = I\}$ is a fixed point of $T_P$.*

LEMMA 2.3. *$\bigcup \{I \mid T_P(I) = I\}$ is a fixed point of $T_P$.*

Lemmas 2.1 and 2.3 imply that if $A$ is a variable-free atomic formula, then $\neg A$ is Herbrand valid in $P$ iff $A \in H - \bigcup \{I \mid T_P(I) = I\}$, where $H$ is the Herbrand base of $P$. We mention in passing that for formulas $B$ of form (2.3), $B$ is Herbrand valid in $P$ iff $B$ is valid, in the ordinary sense, in all models of $P$. This is a fundamental property of the resolution technique (cf. Robinson, 1965). This equivalence of Herbrand validity with ordinary validity does not apply to negations of formulas.

Let $L$ be a complete lattice with bottom $\bot$, and top $\top$, and let $f$ be a monotonic function mapping $L$ to $L$. Let $e \in L$. Define

$$\begin{aligned} f\uparrow^0(e) &= e, & f\downarrow^0(e) &= e, \\ f\uparrow^\alpha(e) &= \bigcup \{f(f\uparrow^\beta(e)) \mid \beta < \alpha\}, & f\downarrow^\alpha(e) &= \bigcap \{f(f\downarrow^\beta(e)) \mid \beta < \alpha\}, \end{aligned} \tag{2.4}$$

for each ordinal $\alpha$.

Here $f$ has a unique maximal and a unique minimal fixed point. The minimal fixed point of $f$ is given by $f\uparrow^\alpha(\bot)$, where $\alpha$ is the least ordinal such that $f\uparrow^\alpha(\bot) = f\uparrow^{\alpha+1}(\bot)$, and the maximal fixed point of $f$ is given by $f\downarrow^\lambda(\top)$, where $\lambda$ is the least ordinal such that $f\downarrow^\lambda(\top) = f\downarrow^{\lambda+1}(\top)$. Let

$\|f\uparrow\|$ and $\|f\downarrow\|$ be these ordinals. These remarks apply when $f$ is the operator $T_P$ and $L$ is the power set of $H$, the Herbrand base of $P$.

In Apt and van Emden (1980) the finite-failure set of logic program $P$ is defined to be the set of all variable-free atomic formulas $A$ such that there is a linear resolution tree with $A$ as root which is finite and every path terminates in failure; that is, the finite failure set of $P$ is

$$\{A \mid A \text{ is atomic, variable-free, and } \neg(P \vdash A) \text{ is provable by the termination in failure of linear resolution}\}.$$

Linear resolution is the proof procedure employed by the pure definite clause part of PROLOG. Lemma 2.4 is proved in Apt and van Emden (1980).

LEMMA 2.4. *Let $H$ be the Herbrand base of logic program $P$, and let $F$ be the finite-failure set of $P$. Then $F = H - T_P\downarrow^{\omega}(H)$.*

Whether $\|T_P\downarrow\| < \omega$ or not, we have by (2.4)

$$T_P\downarrow^{\|T_P\downarrow\|}(H) = T_P\downarrow^{\omega+\alpha}(H), \text{ for some } \alpha.$$

By Lemma 2.4 we may regard the minimum such $\alpha$ as a measure of the incompleteness of the negation as failure rule where linear resolution is the underlying proof procedure.

Now $T_P$ is an enumeration operator. By Eq. (4.1) and a theorem of C. Spector (Hinman, 1978, p. 150) $\|T_P\downarrow\| \leqslant \omega_1$, where $\omega_1$ is the least nonconstructive ordinal. We shall show that, in fact, $\|T_P\| = \omega_1$ for certain $P$ that we will construct.

We assume the reader is familiar with the fundamental properties of the arithmetical and analytical hierarchies as well as the notions of 1–1 reducibility and recursive isomorphism. If $X$ is a $\Pi^1_1$ set of numbers for which every $\Pi^1_1$ set of numbers is 1–1 reducible to $X$, then $X$ is said to be $\Pi^1_1$-complete. It follows, therefore, that up to recursive isomorphism there is only one $\Pi^1_1$-complete set. Kleene's system of ordinal notations is a well-known example of such a set.

We shall employ Turing machines to establish the principal results of Section 3. We adopt most of the conventions of Rogers (1967) regarding Turing machines, with the following exceptions: We identify the equivalence class of instantaneous descriptions which represent a given tape-state configuration with the tape-state configuration itself. Our Turing machines are over the alphabet $\{M, B\}$ with $B$ representing "blank."

An instantaneous description is a description of a portion of the tape containing the nonblank portion of the tape with the state inserted immediately to the left of the scanned cell. Thus $xqy$, where $q \in \mathbb{N}$, and $x, y \in \{B, M\}^*$ is an instantaneous description. Thus two instantaneous

descriptions are equivalent if they only differ by the number of leading and trailing $B$'s. The states are nonnegative integers. In addition to the deterministic Turing machines which are bijectively assigned Gödel numbers, we also consider nondeterministic Turing machines which we do not assign Gödel numbers. If $F$ is a Turing machine, deterministic or not, $F$ halts on tape-state configuration $c$ if $F$ reaches a halting state on all computations proceeding from $c$. We assume that the notion of tape-state configuration $c'$ reachable from tape-state configuration $c$ in $n$ steps by Turing machine $F$ is sufficiently clear not to need further elaboration. The domain($F$) is the set of all tape-state configurations represented by

$$\underbrace{0M \cdots M}_{n+1}, \tag{2.5}$$

where $0 \in \mathbb{N}$, and $M \cdots M \in \{B, M\}^*$ for which $F$ halts. As is often the case in the discussion of Turing machines, (2.5) represents the input of the number $n$ to $F$. In particular, if $F$ is deterministic with Gödel number $z$, then domain$(F) = W_z$.

If $S$ is a set, $S^*$ is the set of all finite strings with elements in $S$. The symbol $\biguplus$ denotes disjoint union; $\langle x, y \rangle$ is the code number of $(x, y)$ as in Rogers (1967); $\langle \cdot, \cdot \rangle$ is bijective; $(\langle x, y \rangle)_0 = x$ and $(\langle x, y \rangle)_1 = y$; and $\phi_z$ is the partial recursive function computed by the deterministic Turing machine with Gödel number $z$. Our notation will be somewhat more compact with

DEFINITION 2.3. Let $H$ be the Herbrand base of logic program $P$, and let $R$ be a relation symbol. Now $H_R$ is the subset of $H$ consisting of all formulas in which the symbol $R$ occurs. If $R \notin L(P)$, $H_R$ is empty.

## 3. FINITE-FAILURE SETS

The principal result of this section is that the class of all finite-failure sets is the class of all r.e. sets under a suitable representation of nonnegative integers by variable-free terms. This result is neither deep nor surprising, but it does need a demonstration, and the constructions employed in the demonstration given here provide the groundwork for proving that Herbrand validity is $\Pi_1^1$-complete.

Linear resolution trees are finitely branched. Thus all finite initial segments of all linear resolution trees for logic programs $P$ proceeding from negative clause $A\rightarrow$ can be recursively enumerated. Thus, if $A$ is in the finite-failure set of $P$, the finite linear resolution tree which fails $A\rightarrow$ will appear in the enumeration. Consequently, each finite-failure set is r.e. It remains to

show that the class of all r.e. sets is 1–1 reducible to the class of finite-failure sets.

Turing machine quadruples can be represented as definite clauses expressing transitions between instantaneous descriptions. A string over the alphabet $\{M, B\}$ can be represented by a term in a logic program using the constant symbol $a$, and the letters $M$ and $B$ as unary function symbols. The singleton strings $M$ and $B$ are represented by $M(a)$ and $B(a)$, respectively. If $t$ is a term representing the string $\sigma$, then $M(t)$ and $B(t)$ represent the strings $M\sigma$ and $B\sigma$, respectively. An instantaneous description has the form

$$s_1 q s_2, \tag{3.1}$$

where $s_1, s_2 \in \{M, B\}^*$. Let $s_1^R$ be the result of reversing $s_1$ and let $t_1, t_2$ represent $s_1^R$ and $s_2$, respectively. Represent each $n \in \mathbb{N}$ by the term

$$\underbrace{M(\cdots M(a)\cdots)}_{n}.$$

We abbreviate this term by $M^n(a)$. Then the instantaneous description (3.1) can be represented by the atomic formula

$$ID(t_1, t_2, M^q(a)). \tag{3.2}$$

For eample, the instantaneous description

$$BMMMBMM3MBMMBMBBB$$

is represented by

$$ID(M(M(B(M(M(M(B(a)))))))), M(B(M(M(B(M(B(B(a))))))))),$$
$$M(M(M(a)))).$$

We now construct a logic program $PF$ corresponding to (possibly nondeterministic) Turing machine $F$. The definition is divided into three cases according to whether a quadruple in $F$ contains a symbol to write or a command to move left or right. The purpose of the additional clauses in each case is to control the proliferation of leading and trailing $B$'s. Since we do not have logical equality in a logic program we cannot include the clause $\rightarrow a = B$. An example follows the definition.

DEFINITION 3.1. (1) For each quadruple $\langle p, f, g, q\rangle$ in $F$ include in $PF$ the clause

(a) $ID(x, f(y), M^p(a)) \rightarrow ID(x, g(y), M^q(a))$.

In addition: if $f = B$ and $g = B$ include

(b) $ID(x, a, M^p(a)) \rightarrow ID(x, a, M^q(a))$;
if $f = B$ and $g = M$ include

(c) $ID(x, a, M^p(a)) \rightarrow ID(x, M(a), M^q(a))$;
if $f = M$ and $g = B$ include

(d) $ID(x, M(a), M^p(a)) \rightarrow ID(x, a, M^q(a))$.

(2) For each quadruple $\langle p, f, L, q \rangle$ in $F$ include in $PF$ the clauses

(a) $ID(B(x), f(y), M^p(a)) \rightarrow ID(x, B(f(y)), M^q(a))$

(b) $ID(M(x), f(y), M^p(a)) \rightarrow ID(x, M(f(y)), M^q(a))$.
In addition: include

(c) $ID(a, f(y), M^p(a)) \rightarrow ID(a, B(f(y)), M^q(a))$;
if $f = B$ include

(d) $ID(B(x), a, M^p(a)) \rightarrow ID(x, a, M^q(a))$;

(e) $ID(a, a, M^p(a)) \rightarrow ID(a, a, M^q(a))$;

(f) $ID(M(x), a, M^p(a)) \rightarrow ID(x, M(a), M^q(a))$.

(3) For each quadruple $\langle p, f, R, q \rangle$ in $F$, include in $PF$ the clause

(a) $ID(x, f(y), M^p(a)) \rightarrow ID(f(x), y, M^q(a))$.
In addition: if $f = B$ include

(b) $ID(x, a, M^p(a)) \rightarrow ID(B(x), a, M^q(a))$;

(c) $ID(a, a, M^p(a)) \rightarrow ID(a, a, M^q(a))$;

(d) $ID(a, B(y), M^p(a)) \rightarrow ID(a, y, M^q(a))$.

This completes the definition of $PF$.

As an example, consider the Turing machine $F$ defined by $\{\langle 0, M, B, 0\rangle, \langle 0, B, R, 1\rangle, \langle 1, M, B, 0\rangle\}$ which erases the $M$'s immediately to the right of the initially scanned cell when starting in state 0. $PF$ is given by

(1) $\{ID(x, M(y), a) \rightarrow ID(x, B(y), a)$,

(2) $ID(x, M(a), a) \rightarrow ID(x, a, a)$,

(3) $ID(x, B(y), a) \rightarrow ID(B(x), y, M(a))$,

(4) $ID(x, a, a) \rightarrow ID(B(x), a, M(a))$,

(5) $ID(a, a, a) \rightarrow ID(a, a, M(a))$,

(6) $ID(a, B(y), a) \rightarrow ID(a, y, M(a))$,

(7) $ID(x, M(y), M(a)) \rightarrow ID(x, B(y), a)$,

(8) $ID(x, M(a), M(a)) \rightarrow ID(x, a, a)\}$.

The behavior of $F$ is illustrated by the transitions $0MM \vdash 0BM \vdash BIM \vdash B0B \vdash BB1$. Correspondingly, from $ID(a, M(M(a)), a)$, one can derive

$ID(B(B(a)), a, M(a))$, using clauses (1), (3), and (7). The transitions of $F$ from $0MM$ are also given by $0MM \vdash 0BM \vdash IM \vdash 0 \vdash 1$. Correspondingly, from $ID(a, M(M(a)), a)$ one derives $ID(a, a, M(a))$ by clauses (1), (6), (8), and (5).

DEFINITION 3.2. If $d$ and $e$ are instantaneous descriptions of the same tape-state configuration, then $d$ and $e$ are equivalent. (Note that $d$ and $e$ differ only by the number of leading and trailing blanks.)

DEFINITION 3.3. Let $A$ be an instance of $ID(x, y, z)$ representing instantaneous description $d$. Then

$A^0 = \{B \mid B$ represents some instantaneous description equivalent to $d\}$.

If $S$ is a set of formulas, let $S^0 = \bigcup_{A \in S} A^0$. (Note that $A^0$ and the class of all instantaneous descriptions equivalent to $d$ are in one-to-one correspondence, and can be identified.)

LEMMA 3.1. $T_{PF}(A^0) = \bigcup_{A' \in A^0} [T_{PF}(\{A'\})]^0$.

DEFINITION 3.4. Let states $(F)$ be the set of states that occur in some quadruple in Turing machine $F$.

(1) For each $q$ in states $(F)$, select new states $q^L$, and $q^R$ such that all are distinct.

(2) For each $\langle p, x, y, q\rangle$ in $F$, where $y \in \{M, B\}$ include in reverse$(F)$ the quadruple $\langle q, y, x, p\rangle$.

(3) For each $\langle p, x, L, q\rangle$ in $F$ include in reverse$(F)$ the following quadruples: $\langle q, B, R, q^R\rangle$, $\langle q, M, R, q^R\rangle$, and $\langle q^R, x, x, p\rangle$.

(4) For each $\langle p, x, R, q\rangle$ in $F$ include in reverse$(F)$ the following quadruples: $\langle q, B, L, q^L\rangle$, $\langle q, M, L, q^L\rangle$, and $\langle q^L, x, x, p\rangle$.

(5) There are no quadruples in reverse$(F)$ other than those that are included by (1)–(4).

We shall show that given r.e. set $W_z$, a Turing machine $G$ can be constructed such that the finite-failure set of the corresponding logic program $PG$ is $W_z$. This is accomplished by constructing $G$ to simulate $F$ running backwards, where domain$(F) = W_z$.

LEMMA 3.2. *Let $c_1$ and $c_2$ by tape-state configurations whose states occur in the quadruples of $F$. Then*

(i) *if $c_2$ can be reached from $c_1$ by $F$ in $n$ steps, then there exists $m$ such that $\frac{1}{2}m \leqslant n \leqslant m$ and $c_1$ can be reached from $c_2$ by reverse($F$) in $m$ steps.*

(ii) *if $c_1$ can be reached from $c_2$ by reverse($F$) in $m$ steps, then there exists $n$ such that $\frac{1}{2}m \leqslant n \leqslant m$ and $c_2$ can be reached from $c_1$ by $F$ in $n$ steps.*

Now, by Lemma 3.1, for $n > 0$,

$$T_{PF}\downarrow^n(H) = \{A \mid \text{for some } B \in H,\ A^0 \text{ can be reached from } B^0 \text{ in exactly } n \text{ steps by } F\}.$$

Consequently, since $T_{PF}\downarrow^\alpha(H)$ is nested as $\alpha$ increases,

$$H\text{–}\!\!\neg\, T_{PF}\downarrow^\omega(H) = \{A \mid \text{for some } n_A \in \mathbb{N}, \text{ for every } B \in H, \text{ and every } n \geqslant n_A,\ A^0 \text{ cannot be reached from } B^0 \text{ in exactly } n \text{ steps by } F\}$$

$$(\text{by Lemma 3.2}) = \{A \in H \mid \text{reverse}(F) \text{ halts starting in } A^0 \text{ if the state of } A^0 \text{ is a state of } F\}.$$

It also follows from Lemma 3.2 that if $c$ is a tape-state configuration whose state is a state occurring in the quadruples of $F$, then $F$ halts starting in $c$ iff reverse (reverse($F$)) halts starting in $c$. Consequently, we have

LEMMA 3.3. *Let*

$$H^F = \{A \mid \textit{the state of } A^0 \textit{ is a state occurring in the quadruples in } F\}.$$

*Then,*

$$H^F \cap (H\text{–}\!\!\neg\, T_{P\,\text{reverse}(F)}\downarrow^\omega(H)) = \{A \in H^F \mid F \textit{ halts starting in } A^0\}.$$

Let nonnegative integer $n$ be represented by the tape-state configuration $ID(a, M^{n+1}(a), a)^0$. Let $B \in A^0$. Then by Lemma 3.1

$$B \in H^F \cap (H\text{–}\!\!\neg\, T_{P\,\text{reverse}(F)}\downarrow^\omega(H)) \quad \text{iff} \quad A \in H^F \cap (H\text{–}\!\!\neg\, T_{P\,\text{reverse}(F)}\downarrow^\omega(H)). \tag{3.3}$$

Lemma (3.3) implies

$$n \in \text{domain}(F) \quad \text{iff} \quad ID(a, M^{n+1}(a), a) \in H\text{–}\!\!\neg\, T_{P\,\text{reverse}(F)}\downarrow^\omega(H). \tag{3.4}$$

It is in this sense that every r.e. set can be represented by a finite-failure set. Equation (3.4) gives a 1–1 reducibility of the class of r.e. sets into the class of finite-failure sets.

Let $F$ be a Turing machine. Obtain $PNF$ from $PF$ by including the following new clauses:

$$ID(x, y, a) \& V(x) \& V(y) \to W(x, y)$$
$$\to V(M(a))$$
$$V(x) \to V(M(x)).$$

Let $H$ be the Herbrand base of $PNF$ and let $H_{ID}$ be the Herbrand base of $PF$. Then $H_{ID} \cap T_{PNF}\downarrow^{\omega}(H) = T_{PF}\downarrow^{\omega}(H_{ID})$. Also

$$\begin{aligned} &W(t_1, t_2) \in T_{PNF}\downarrow^{\omega}(H) \\ &\quad \text{iff} \quad ID(t_1, t_2, a),\ V(t_1),\ V(t_2) \in T_{PNF}\downarrow^{\omega}(H) \\ &\quad \text{iff} \quad ID(t_1, t_2, a) \in T_{PF}\downarrow^{\omega}(H_{ID}) \text{ and } t_1 \text{ and } t_2 \text{ are of the form } M^{n_1}(a) \\ &\qquad \text{and } M^{n_2}(a), \text{ respectively, for some } n_1, n_2 \geqslant 1. \end{aligned} \tag{3.5}$$

Associated with each Turing machine $F$ there is a Turing machine $F'$ which halts when started in the tape-state configuration represented by

$$\underbrace{M \cdots M}_{n_1} 0 \underbrace{M \cdots M}_{n_2} \qquad (n_1, n_2 \geqslant 1)$$

iff $F$ halts when starting in the tape-state configuration represented by

$$\underbrace{0M \cdots M.}_{\langle n_1 - 1, n_2 - 1\rangle + 1}$$

Constructing $F'$ from $F$ is a straightforward, although tedious, exercise which we leave to the reader.

By (3.5) and Lemma 3.3, we have

LEMMA 3.4.

$$\begin{aligned} &W(t_1, t_2) \in T_{PN\,\mathrm{reverse}(F')}\downarrow^{\omega}(H) \\ &\quad \textit{iff for some } n_1, n_2 \geqslant 1,\ t_1 \textit{ is } M^{n_1}(a),\ t_2 \textit{ is } M^{n_2}(a), \\ &\quad \textit{and } \langle n_1 - 1, n_2 - 1\rangle \textit{ is not in } \mathrm{domain}(F). \end{aligned}$$

One more lemma will put us in a position to study the range of descending closure ordinals of the various $T_P$.

LEMMA 3.5. *For each Turing machine* $F$, $\| T_{PNF}\downarrow \| = \omega$.

*Proof.* It suffices to show

$$T_{PNF}(T_{PNF}\downarrow^{\omega}(H)) = T_{PNF}\downarrow^{\omega}(H),$$

where $H$ is the Herbrand base of $PNF$. $H = H_{ID} \cup H_V \cup H_W$, (cf. Definition 2.3). Now,

$$ID(t_1, t_2, t_3) \in T_{PNF}(T_{PNF}\downarrow^{\omega}(H))$$
iff for some $ID(t_1', t_2', t_3') \in T_{PNF}\downarrow^{\omega}(H)$,
$ID(t_1', t_2', t_3') \to ID(t_1, t_2, t_3)$
is a variable-free instance of the matrix of a clause in $PF$. (3.6)

Also,

$$ID(t_1, t_2, t_3) \in \bigcap_{n<\omega} T_{PNF}\downarrow^{n}(H)$$
iff for each $n < \omega$, there is a formula $ID(t_{1,n}, t_{2,n}t_{3,n})$
in $T_{PNF}\downarrow^{n}(H)$ and $ID(t_{1,n}, t_{2,n}, t_{3,n}) \to ID(t_1, t_2, t_3)$
is a variable-free istance of the matrix of a clause in $PF$
iff (since $PF$ is finite) Eq. (3.6). ∎

## 4. Lemmas Concerning Enumeration Operators

Definition 4.1. Let $S \subseteq \mathbb{N}$. Define the operator

$$Z_S: P(\mathbb{N}) \to P(\mathbb{N})$$

by

$$Z_S(X) = \{y \mid \exists u[\langle y, u\rangle \in S \,\&\, u \in X]\}.$$

In case $S$ is r.e., $Z_S$ is an enumeration operator, although not all enumeration operators are given by the various $Z_S$ (Rogers, 1967).

Definition 4.2. Suppose $\Gamma: P(\mathbb{N}) \to P(\mathbb{N})$. Let $\Gamma^d$ (read $\Gamma$-dual) be defined by

$$\Gamma^d(X) = \mathbb{N} - \Gamma(\mathbb{N} - X).$$

The following are easy to verify.

$$(\Gamma^d)^d = \Gamma. \tag{4.1}$$

$$\|\Gamma^d\uparrow\| = \|\Gamma\downarrow\|. \tag{4.2}$$

$$\|\Gamma^d\downarrow\| = \|\Gamma\uparrow\|. \tag{4.3}$$

$$Z_S^d(X) = \{y \mid \forall u[\langle y, u\rangle \in S \to u \in X]\}. \tag{4.4}$$

Also, for any ordinal $\alpha$,

$$\Gamma\downarrow^{\alpha}(\mathbb{N}) = \mathbb{N} - \Gamma^{d}\uparrow^{\alpha}(\varnothing). \tag{4.5}$$

For any hyperarithmetical set $S$, both $Z_S$ and $Z_S^d$ are $\Pi_1^1$ operators. It follows that (Hinman, 1978)

$$Z_S^d\uparrow^{\|Z_S^d\uparrow\|}(\varnothing) \text{ is } \Pi_1^1, \text{ and } \|Z_S^d\uparrow\| \leqslant \omega_1, \text{ if } S \text{ is hyperarithmetical.} \tag{4.6}$$

This section is concerned with establishing two technical results, as we shall see in the next section, that apply to the operators $T_P$ associated with logic programs. We state both results now, and then present their proofs below.

THEOREM 4.1. *For each ordinal $\alpha \leqslant \omega_1$, there is a recursive set $S$ such that*

$$\|Z_S^d\uparrow\| = \alpha.$$

THEOREM 4.2. *There is a recursive set $S$ such that*

(i) $\|Z_S^d\uparrow\| = \omega_1$ *and*

(ii) $Z_S^d\uparrow^{\omega_1}(\varnothing)$ *is $\Pi_1^1$-complete.*

The proof of Theorem 4.2 to be presented is independent of the proof of Theorem 4.1. The case $\alpha = \omega_1$ in Theorem 4.1 is implied by Theorem 4.2.

*Proof of Theorem* 4.1. It suffices to show that Theorem 4.1 holds for each $\alpha < \omega_1$. If $0 \leqslant \alpha < \omega$ the theorem is easy to verify.

Suppose $\omega \leqslant \alpha < \omega_1$. There is a recursive binary relation $R'$ which is a strict well ordering (order-isomorphic to $\alpha$) of some set of integers $\hat{R}'$; $\hat{R}'$ is infinite and r.e. Thus, $R'$ and $\hat{R}'$ can be replaced by a recursive $R$, order isomorphic to $\alpha$, with $\hat{R} = \mathbb{N}$ (Rogers, 1967). There is a 1–1 correspondence between $\hat{R}$ and $\{\beta \mid \beta < \alpha\}$ via the order isomorphism. Let $|\beta|$ be the element of $\mathbb{N}$ corresponding to $\beta$. Define

$$\langle y, x\rangle \in S \leftrightarrow R(x, y)$$

and

$${}^{R}y = \{x \mid R(x, y)\}$$

Thus, for $\beta < \alpha$, $R_{|\beta|}$ is order isomorphic to $\beta$. Define $R_{|\alpha|} = \mathbb{N}$, which is order isomorphic to $\alpha$.

By a straightforward transfinite induction using the fact that

$$Z_S^d({}^{R}y) = {}^{R}y \cup \{y\}$$

we have

$$\forall(\beta \leqslant \alpha)[Z_S^d \uparrow^\beta (\varnothing) = {}^R|\beta|].$$

Consequently,

$$Z_S^d \uparrow^{\alpha+1} (\varnothing) = Z_S^d(\mathbb{N}) = \mathbb{N}.$$

Therefore,

$$\| Z_S^d \uparrow \| = \alpha.$$

This completes the proof except for the case $\alpha = \omega_1$, which is implied by Theorem 4.2. ∎

DEFINITION 4.3. Let $\mathrm{Dom}(X) = \{ y \mid W_y \subseteq X\}$, where $X \subseteq \mathbb{N}$.

If $f\colon \mathbb{N} \to \mathbb{N}$, then $f$ can be regarded as an operator mapping $P(\mathbb{N})$ into $P(\mathbb{N})$ in the obvious way by

$$f(X) = \{ y \mid \exists x[x \in X \,\&\, y = f(x)]\}.$$

For any set $S \subseteq \mathbb{N}$, let $\bar{S} = \mathbb{N} - S$. Also, for operators $F$ and $G$ let

$$FG(X) = F(G(X)).$$

Lemma 4.1 will provide a means of choosing a recursive $S$ to satisfy the requirements of Theorem 4.2.

LEMMA 4.1. *There exist one-to-one total recursive functions $f$ and $\bar{f}$ such that*

(i) *$W_{f(x)}$ is recursive, and $W_{\bar{f}(x)} = \overline{W_{f(x)}}$, for every $x$.*

(ii) $\mathrm{Dom} \uparrow^\alpha (\varnothing) = (f^{-1}\,\mathrm{Dom}) \uparrow^\alpha (\varnothing)$.

*Proof.* Let $d\colon \mathbb{N} \times \mathbb{N} \to \mathbb{N}$ be a total, recursive, one-to-one function satisfying

$$n < d(n, 0) < d(n, 1) < \cdots \tag{4.7}$$

and

$$W_n = W_{d(n,0)} = W_{d(n,1)} = \cdots. \tag{4.8}$$

Let $T$ be the Kleene normal form predicate satisfying

$$\phi_z(x) \simeq (\mu y T(z, x, y))_0, \tag{4.9}$$

where $\simeq$ means both sides are defined and equal to each other, or both sides are undefined. Then

$$m \in W_z \leftrightarrow \exists\, y T(z, m, y). \tag{4.10}$$

Define $h$ by

$$h(z, 0) = d(0, 0);$$

$$\begin{aligned} h(z, n+1) &= d(x, k+1), \quad \text{if } h(z, n) = d(x, k) \text{ and } \neg T(z, (n)_0, (n)_1), \\ &= d((n)_0, \mu y[d((n)_0, y) > h(z, n)]), \quad \text{if} \\ &\qquad T(z, (n)_0, (n)_1) \text{ and } (n)_0 \neq 0, \\ &= d(d((n)_0, 0), \mu y[d(d((n)_0, 0), y) > h(z, n)]), \quad \text{if} \\ &\qquad T(z, (n)_0, (n)_1) \text{ and } (n)_0 = 0. \end{aligned}$$

*Claim* 1. (i) $h$ is total, and $\text{range}(h) \subseteq \text{range}(d)$.

(ii) For each $z$, $h(z, 0) < h(z, 1) < h(z, 2) < \cdots$.

To prove Claim 1, one proves simultaneously by ordinary induction that

(i) $h(z, 0), \ldots, h(z, n)$ are all defined and in $\text{range}(d)$, and

(ii) $h(z, 0) < h(z, 1) < \cdots < h(z, n)$. ∎

Let $Q = \{d(0, n) \mid n \in \mathbb{N}\}$; $Q$ is recursive by (4.7, 4.8). Let

$$B_z = \text{range}(\lambda n(h(z, n))) \cap (\mathbb{N} - Q). \tag{4.11}$$

By Claim 1, $B_z$ is recursive, and a decision procedure for $B_z$ can be determined uniformly from $z$. Therefore, there are total recursive functions $f$ and $\bar{f}$ such that

$$W_{f(z)} = B_z \tag{4.12}$$

and

$$W_{\bar{f}(z)} = \overline{B_z} = \overline{W_{f(z)}}. \tag{4.13}$$

Moreover, we can choose $f$ and $\bar{f}$ one-to-one. This establishes Lemma 4.1(i).

*Claim* 2. $\text{Dom}(\varnothing) = f^{-1}\,\text{Dom}(\varnothing)$.

*Proof.* Suppose $z \in \text{Dom}(\varnothing)$, then $W_z = \varnothing$. By (4.10) $\forall n \neg T(z, (n)_0, (n)_1)$. By definition of $h$, and (4.11) $W_{f(z)} = \{d(0, 0), d(0, 1), \ldots\} \cap \overline{Q} = \varnothing$. Thus

$$f(z) \in \text{Dom}(\varnothing) \tag{4.14}$$

and

$$z \in f^{-1}\,\text{Dom}(\varnothing). \tag{4.15}$$

Conversely, suppose (4.15). Then (4.14). Suppose $W_z \neq \emptyset$. For some $y$, and $m \in W_z$, $T(z, m, y)$. Let $n = \langle m, y \rangle$. Then

$$\begin{aligned} h(z, n+1) &= d((n_0, \mu y[d((n)_0, y) > h(z, n)], && \text{if} \quad (n)_0 \neq 0, \\ &= d(d((n)_0, 0), \mu y[d(d((n)_0, 0), y) > h(z, n)]), && \text{if} \quad (n)_0 = 0. \end{aligned}$$

Therefore, since $d$ is 1–1, $h(z, n+1) \notin Q$.

By (4.11), $h(z, n+1) \in W_{f(z)}$ which is empty. The contradiction implies $W_z = \emptyset$, and hence $z \in \text{domain}(\emptyset)$. This completes the proof of Claim 2.

*Claim* 3. For all $\alpha$, $\text{Dom} \uparrow^\alpha (\emptyset) = (f^{-1}\,\text{Dom}) \uparrow^\alpha (\emptyset)$.

*Proof.* We proceed by transfinite induction.

If $\alpha = 0$, then the claim is trivial. Suppose $\alpha > 0$. Then

$$\begin{aligned} &(f^{-1}\,\text{Dom}) \uparrow^\alpha (\emptyset) \\ &\quad = \bigcup_{\beta<\alpha} (f^{-1}\,\text{Dom})((f^{-1}\,\text{Dom})((f^{-1}\,\text{Dom}) \uparrow^\beta (\emptyset)) \\ &\quad = \bigcup_{\beta<\alpha} (f^{-1}\,\text{Dom})(\text{Dom} \uparrow^\beta (\emptyset)) \qquad \text{(by induction)} \\ &\quad = \bigcup_{\beta<\alpha} f^{-1}(\text{Dom} \uparrow^{\beta+1} (\emptyset)). \end{aligned}$$

It remains to prove

$$f^{-1}(\text{Dom} \uparrow^{\beta+1} (\emptyset)) = \text{Dom} \uparrow^{\beta+1} (\emptyset). \tag{4.16}$$

We show this directly, without transfinite induction. If $\beta = 0$, then (4.16) is Claim 2. Assume $\beta > 0$. It must be shown that

$$W_{f(z)} \subseteq \text{Dom} \uparrow^\beta (\emptyset) \qquad \text{iff} \quad W_z \subseteq \text{Dom} \uparrow^\beta (\emptyset). \tag{4.17}$$

By Claim 2 we can assume that neither $W_{f(z)}$ nor $W_z$ is empty. Let $m \in W_z$. As in the proof of the previous claim there is some $y$ such that

$$T(z, m, y),$$

and letting $n = \langle m, y \rangle$ we have by (4.11)

$$h(z, n+1) \in W_{f(z)}.$$

Thus:

if $m \in W_z$, then $h(z, n+1) \in W_{f(z)}$ for some $n$ such that $(n)_0 = m$. (4.18)

Suppose $h(z, n+1) \in W_{f(z)}$. For some $r, s \in \mathbb{N}$ $h(z, n+1) = d(r, s)$ $r \neq 0$, otherwise $h(z, n+1) \in Q$, and hence $h(z, n+1) \notin W_{f(z)}$. Let $u =$ the least $v$ such that $d(r, v) \in \text{range}(\lambda n h(z, n))$. Now,

$$W_{h(z,n+1)} = W_{d(r,s)} = W_{d(r,x)} = W_r \qquad \text{for any } x,$$

and for some $t$, $h(z, t) = d(r, u)$. If $t = 0$, then $h(z, t) = d(0, 0)$ and hence $r = 0$. But $r \neq 0$, so $t \neq 0$.

Suppose

$$\neg T(z, (t-1)_0, (t-1)_1).$$

Then $h(z, t) = d(x, k+1)$, where $h(z, t-1) = d(x, k)$. So $r = x$, and $u = k + 1$. Therefore

$$d(r, u-1) \in \text{range}(\lambda n h(z, n))$$

which contradicts the choice of $u$. Therefore we must have

$$T(z, (t-1)_0, (t-1)_1). \tag{4.19}$$

From (4.19) and the definition of $h$ we have

$$W_{h(z,n+1)} = W_{(t-1)_0} \qquad \text{if} \quad h(z, n+1) \in W_{f(z)} \tag{4.20}$$

as we have supposed. Also by (4.19)

$$(t-1)_0 \in W_z.$$

Therefore, if $W_z \subseteq \text{Dom} \uparrow^\beta (\varnothing)$, then $(t-1)_0 \in \text{Dom} \uparrow^\beta (\varnothing)$. By (4.20)

$$h(z, n+1) \in \text{Dom} \uparrow^\beta (\varnothing)$$

for any $n$ such that $h(z, n+1) \in W_{f(z)}$. Every element of $W_{f(z)}$ is of the form $h(z, n+1)$ for some $n$. Thus $W_{f(z)} \subseteq \text{Dom} \uparrow^\beta (\varnothing)$.

Conversely, if $W_{f(z)} \subseteq \text{Dom} \uparrow^\beta (\varnothing)$, by (4.18) for each element $m$ in $W_z$, for some $n$

$$W_m = W_{h(z,n+1)},$$

where $h(z, n+1) \in W_{f(z)}$. Consequently, $W_z \subseteq \text{Dom} \uparrow^\beta (\varnothing)$. This establishes (4.17) and completes the proofs of Claim 3 and Lemma 4.1. ∎

Let $S$ be defined by

$$x \in S \leftrightarrow (x)_1 \in W_{f((x)_0)}.$$

$S$ is recursive by Lemma 4.1(i), and $Z_S^d(X) = f^{-1}\,\mathrm{Dom}(X)$. By Lemma 4.1(ii)

$$Z_S^d \uparrow^\alpha (\varnothing) = (f^{-1}\,\mathrm{Dom}) \uparrow^\alpha (\varnothing) = \mathrm{Dom} \uparrow^\alpha (\varnothing).$$

Thus

$$Z_S^d \uparrow^{\omega_1} (\varnothing) = \mathrm{Dom} \uparrow^{\omega_1} (\varnothing),$$

and

$$\| Z_S^d \uparrow \| = \| \mathrm{Dom} \uparrow \| = \omega_1.$$

It is known that $\| \mathrm{Dom} \uparrow \| = \omega_1$ and that $\mathrm{Dom} \uparrow^{\omega_1} (\varnothing)$, which is recursively isomorphic to Kleene's system of ordinal notations 0, is the (productive) center of the identity function, and is $\Pi_1^1$-complete (Rogers, 1967). This completes the proof of Theorem 4.2. ∎

## 5. Applications to $T_P$

For each logic program *PNF* augment *PNF* to (*PNF*+) by adding the clauses

$$N(x) \to N(s(x)), \tag{5.1}$$

$$N(x) \to A(y), \tag{5.2}$$

$$W(y, x) \,\&\, A(x) \to A(y), \tag{5.3}$$

where $N$ and $A$ are new relation symbols.

Lemma 5.1. *Let $G$ be* reverse($F'$). *Let $H$ be the Herbrand base of* (*PNG*)+. *Let $S =$* domain($F$). *Then for each $\alpha > 0$,*

$$H_A \cap T_{(PNG)+} \downarrow^{\omega+\alpha} (H) = \{A(M^{n+1}(a)) \mid n \in Z_{\bar{S}} \downarrow^\alpha (\mathbb{N})\}.$$

*Proof.* The proof is by transfinite induction. Suppose $\alpha = 1$.

$$n \in Z_{\bar{S}}(\mathbb{N}) \leftrightarrow \exists u[\langle n, u\rangle \in \bar{S}] \leftrightarrow \qquad \text{(by Lemma 3.4)}$$

$$\exists u[W(M^{n+1}(a), M^{u+1}(a)) \in T_{PNG} \downarrow^\omega (H \neg (H_A \cup H_N))] \leftrightarrow$$

$$\exists u[W(M^{n+1}(a), M^{u+1}(a)) \in T_{(PNG)+} \downarrow^\omega (H)] \leftrightarrow$$

$$A(M^{n+1}(a)) \in T_{(PNG)+} \downarrow^{\omega+1} (H).$$

The last equivalence follows from

$$H_A \cap T_{(PNG)+} \downarrow^{\omega} (H) = H_A.$$

Note that clause (5.1) forces relation $N$ to be nonempty at each finite level of descending iteration, and empty at every transfinite level. Consequently, clause (5.2) forces $A$ to consist of the entire Herbrand universe at each finite level of descending iteration, and has no effect on $A$ at each transfinite level.

Suppose $\alpha > 1$. For any $\beta \geqslant 0$,

$$\begin{aligned} A(s) \in T_{(PNG)+}(T_{(PNG)+} \downarrow^{\omega+\beta} (H)) \\ \text{iff} \quad \exists t[W(s,t) \in T_{(PNG)+} \downarrow^{\omega+\beta} (H) \,\&\, A(t) \in T_{(PNG)+} \downarrow^{\omega+\beta}(H)] \\ \text{iff} \quad \exists t[W(s,t) \in T_{PNG} \downarrow^{\omega+\beta} (H) \,\&\, A(t) \in T_{(PNG)+} \downarrow^{\omega+\beta} (H)] \end{aligned} \tag{5.4}$$

(by Lemmas 3.5, 3.4) iff

$$\begin{aligned} &\text{for some } n_1, n_2 \in \mathbb{N}, s \text{ is } M^{n_1+1}(a) \\ &\text{and } \langle n_1, n_2 \rangle \in \bar{S}, \text{ and } A(M^{n_2+1}(a)) \in T_{(PNG)+} \downarrow^{\omega+\beta} (H). \end{aligned} \tag{5.5}$$

The remainder of the proof is now straightforward using the equivalence of (5.4) and (5.5). ∎

Let $F$ be a Turing machine with domain$(F) = S$, where $S$ is a recursive set. Fix a large, countable first order language $L$ suitable for expressing up to a suitable isomorphism, any logic program. Augment $L$ to $L'$ by including logical equality. Fix a Gödel numbering of $L'$ which assigns a Gödel number to each term and formula of $L'$. Identify terms and formulas with their Gödel numbers. We assume that decidable sets of syntactic objects in $L$ encode into recursive sets of Gödel numbers, and conversely. The details are left to the reader. Let $G$ and $H$ be as in Lemma 5.1.

In the following lemma and theorem, $L_0$ is either $L$ or $L'$.

THEOREM 5.1. *For each $\alpha \leqslant \omega_1$ there is a logic program $P$ for which*

$$\| T_P \downarrow \| = \alpha.$$

*Proof.* It is easy to obtain $P$ if $\alpha$ is finite. If $\alpha \geqslant \omega$, then let $P = (PNG)+$, where domain$(F) = S$, and $\bar{S}$ is the recursive set guaranteed by Theorem 4.1 for $\beta$ such that $\omega + \beta = \alpha$, and $G$ is as in Lemma 5.1. ∎

THEOREM 5.2. *There is a logic program $P$ such that*

(i) $\|T_P\downarrow\| = \omega_1$,

(ii) $H - T_P\downarrow^{\omega_1}(H)$ *is* $\Pi^1_1$*-complete, where* $H$ *is the Herbrand base of* $P$.

*Proof.* Let $S$ be the complement of the recursive set guaranteed by Theorem 4.2. Note that $\omega + \omega_1 = \omega_1$ and choose $P$ as in the proof of Theorem 5.1. By Lemma 5.1

$$H_A \cap (H - T_{(PNG)+}\downarrow^{\omega_1}(H)) \tag{5.6}$$

is recursively isomorphic to $Z_{\bar{S}}\uparrow^{\omega_1}(\varnothing)$. Further,

$$(H - H_A) \cap (H - T_{(PNG)+}\downarrow^{\omega}(H)) \tag{5.7}$$

is r.e. Thus

$H - T_{(PNG)+}\downarrow^{\omega_1}(H)$ is the disjoint union of a $\Pi^1_1$-complete set,

namely, (5.6) and an r.e. set, namely, (5.7), each contained in complementary recursive sets, and is therefore itself, $\Pi^1_1$-complete. ∎

LEMMA 5.2. *Define* $R_0$ *by*

$R_0(\langle c, t\rangle) \leftrightarrow$ (i) *There is a recursively axiomatizable theory* $T$ *with language* $L_0$, *with index* $t$ (*i.e., the axioms of* $T$ *constitute an* r.e. *set with index* $t$),

*and* (ii) $c$ *is a formula of* $L_0$,

*and* (iii) $c$ *is valid in every Herbrand model of* $T$.

*Then* $R_0$ *is* $\Pi^1_1$.

*Proof.* (i) is $\Pi^0_1$.

(ii) is recursive.

Define $V$ by

$$V(\alpha, x) \leftrightarrow \alpha \vDash x,$$

where $\alpha \vDash x$ means that if $x$ is a formula of $L_0$, then $x$ is valid in the Herbrand structure for $L_0$ determined by

$$\{n \mid \alpha(n) = 1 \;\&\; n \text{ is a closed atomic formula of } L_0\}.$$

Define $V_x$ by

$$V_x(\alpha) \leftrightarrow V(\alpha, x).$$

If $x$ is closed and quantifier free, $V_x$ is recursive. Clearly, then $V_x$ is arithmetical for each $x$. Hence $V$ is hyperarithmetical.

(iii) is $\forall\alpha[\alpha \vDash T \to V(\alpha, c)]$.

Thus (iii), and $R_0$ is $\Pi^1_1$.

∎

THEOREM 5.3. *Let $\Gamma$ be any $\Pi^1_1$ set of formulas containing all negations of variable-free atomic formulas in $L_0$. Define*

$$R(\langle c, P\rangle) \leftrightarrow \text{(i) } P \text{ is a logic program (logic programs are finite),}$$
$$\text{and (ii) } c \in \Gamma,$$
$$\text{and (iii) } c \text{ is Herbrand valid in } P.$$

*Then $R$ is $\Pi^1_1$-complete.*

*Proof.* CDB($P$) (cf. Definition 2.1) is recursively axiomatized. By Lemma 5.2, $R$ is $\Pi^1_1$. Moreover, $H_A$, which is recursive, maps bijectively onto

$$\{\langle \neg B, (PNG)+\rangle \mid B \in H_A\},$$

for each fixed $P$. Choose $P_0$ by Theorem 5.2. Then

$$B \in (5.6) \leftrightarrow R(\langle \neg B, (P_0 DG)+\rangle).$$

Consequently,

$$\text{Eq. } (5.6) \leqslant_1 R. \quad ∎$$

COROLLARY. *Let $R_0$ be as in Lemma 5.2. $R_0$ is $\Pi^1_1$-complete.*

*Proof.* Fix $t$ to be an index of CDB($P_0$), where $P_0$ is chosen as in the previous theorem. Then

$$R_0(\langle c, t\rangle) \leftrightarrow R(\langle c, P_0\rangle),$$

hence

$$\{c \mid R(\langle c, P_0\rangle)\} \leqslant_1 R_0.$$

So $R_0$ is $\Pi^1_1$-hard. By Lemma 5.2 $R_0$ is $\Pi^1_1$-complete. ∎

## ACKNOWLEDGMENTS

The author has benefited from a private cmmunication from M. van Emden. Special thanks are due to K. Bowen for valuable discussions. Thanks are due also to the referees for suggestions on presenting the motivations for the results.

## REFERENCES

APT K. R., AND VAN EMDEN, M. H. (1980), Contributions to the Theory of Logic Programming, Research Report CS-80-72, University of Waterloo, Dept. of Computer Science, Waterloo, Ontario.

CLARK, K. L. (1980), Negation as failure, *in* "Logic and Data Bases" (Gallaire and Minker, Eds.), pp. 293–322, Plenum, New York.

HINMAN, P. G. (1978), "Recursion-Theoretic Hierarchies," Springer-Verlag, Berlin.

KOWALSKI, R. A. (1979), "Algorithm = Logic + Control," *Comm. ACM* 424–436.

ROBINSON, J. A. (1965), "A Machine-Oriented Logic Based on the Resolution Principle," *J. Assoc. Comput. Mach.* 23–41.

ROGERS, H. (1967), "Theory of Recursive Functions and Effective Computability," McGraw–Hill, New York.

VAN EMDEN, M. G., AND KOWALSKI, R. A. (1976), "The Semantics of Predicate Logic as a Programming Language," *J. Assoc. Comput. Mach.* 733–742.