The Expressiveness of Locally Stratified Programs

Howard A. Blair[1]
School of Computer and Information Science
Syracuse University
Syracuse, New York 13244-4100
blair@top.cis.syr.edu

Wiktor Marek[2]
Department of Computer Science
University of Kentucky
Lexington, KY 40506
marek@ms.uky.edu

John S. Schlipf[3]
Department of Computer Science
University of Cincinnati
Cincinnati, Ohio 45221-0008
John.Schlipf@uc.edu

Abstract

This paper completes an investigation of the logical expressibility of finite, locally stratified, general logic programs. We show that every hyperarithmetic set can be defined by a suitably chosen locally stratified logic program (as a set of values of a predicate over its perfect model). This is an optimal result, since the perfect model of a locally stratified program is itself an implicitly definable hyperarithmetic set (under a recursive coding of the Herbrand base); hence to

[1]Research partially supported by the Xerox Corporation, Webster Research Center, Systems Sciences Laboratory and partially supported by the U.S. Army Research Office through the Mathematical Sciences Institute of Cornell University.

[2]Research partially supported by NSF Grant RII 8610671 and partially supported by Army Research Office DAAL03-89-K-0124.

[3]Research partially supported by NSF grant IRI-8905166 and partially supported by the U.S. Army Research Office through the Mathematical Sciences Institute of Cornell University.

obtain all hyperarithmetic sets requires something new, in this case selecting one predicate from the model.

We find that the expressive power of programs does not increase when one considers the programs which have a unique stable model or a total well-founded model. This shows that all these classes of structures (perfect models of locally stratified logic programs, well-founded models which turn out to be total, and stable models of programs possessing a unique stable model) are all closely connected with Kleene's hyperarithmetical hierarchy. Thus, for general logic programming, negation with respect to two-valued logic is related to the hyperarithmetic hierarchy in the same way as Horn logic is to the class of recursively enumerable sets. In particular, a set is definable in the well-founded semantics by a program P whose well-founded partial model is total iff it is hyperarithmetic.

1 Introduction

In this paper we investigate the complexity of the perfect model semantics of locally stratified logic programs. Our results extend the results of:

Smullyan [Smu61] and Andreka and Nemeti [AN78] on the expressive power of Horn programs, and

Apt and Blair [AB90] results on the expressive power of stratified logic programs.

The results of Apt and Blair state that the Σ_n^0 sets of natural numbers are precisely defined by the stratified programs with n strata. This result (as we show) extends into the transfinite with the locally stratified logic programs.

Specifically, we show that the locally stratified logic programs defined precisely the hyperarithmetic sets (that is Δ_1^1 sets). This shows that the class of predicates definable by the locally stratified logic programs coincides with the class of predicates definable by programs with the unique stable models (see [MNR90]) as well as programs with the two-valued well-founded model [**?**, **?**].

As a consequence, we show that the minimum lengths of (local) stratification of logic programs are arbitrary large constructive ordinals.

We also observe that the well-founded semantics, up to any constructive ordinal, can be uniformly simulated by locally stratified programs.

2 Preliminaries

We assume the reader is familiar with the basics of logic programming and the Herbrand semantics of logic programs. For the interested reader, greater detail on preliminary technical matters can be found in a variety of sources, [Ll87] and [ABW88], in particular. The following definitions and facts are useful to recall.

The definition of an Herbrand interpretation, and the one-step deduction operator $\mathbf{T}_P$ associated with a logic program P is the usual one [Ll87, Apt90].

The operator $\mathbf{T}_P$ can be iterated, transfinitely. There are several useful ways to do this, but we will confine ourselves to the way that is most efficiently suited to our current purposes. The limit ordinal and successor ordinal cases do not need to be distinguished.

Definition 2.1:

$$\begin{aligned} \mathbf{T}_P \Uparrow 0(I) &= I \\ \mathbf{T}_P \Uparrow \alpha(I) &= \bigcup_{\beta<\alpha} \mathbf{T}_P(\mathbf{T}_P \Uparrow \beta(I)) \end{aligned}$$

When P is a definite clause program, $\mathbf{T}_P \Uparrow \omega(\emptyset)$ is the least Herbrand model of P.

Hereafter, L is a fixed, countable, language. Locally stratified programs were originally introduced by Przymusinski, [Pr88]. Following the presentation in Apt and Bezem, [ABe90] we define the locally stratified logic programs via

Definition 2.2 *A program P is* locally stratified *if there exists a mapping* `stratum`*, which we call a* partitioning*, from B_L to the countable ordinals such that for every ground instance $A \leftarrow L_1 \,\&\, \ldots \,\&\, L_n$ of a clause in P the following conditions hold for each $1 \leq i \leq n$:*

(i) if L_i is B then $\texttt{stratum}(A) \geq \texttt{stratum}(B)$*;*
(ii) if L_i is $\neg B$ then $\texttt{stratum}(A) > \texttt{stratum}(B)$*, and,*
so that we don't waste ordinals,
(iii) the range of `stratum` *is closed under initial segments;*
i.e. *if $\alpha \in$ range(*`stratum`*) and $\beta < \alpha$, then $\beta \in$ range(*`stratum`*).*

The mapping `stratum` determines a transfinite partition of B_L. Let $H_\alpha = \texttt{stratum}^{-1}(\alpha)$.

We say that a clause of the form

$$A \;\leftarrow\; B_1 \& \;\ldots\; \& \, B_m \,\&\, \neg C_1 \,\&\; \ldots \;\&\, \neg C_n$$

is a clause in *normal order* and is *a normal order* of any clause resulting from a permutation of the literals in its body.

Let $\mathrm{grd}_L(P)$ be the set of all ground (*i.e.* variable-free) instances of normal orders of the clauses of program P with respect to the language L. The fact that there are in general multiple normal orders of program clauses in P is immaterial.

Definition 2.3 *Let P be locally stratified, and let* `stratum` *be the associated partitioning. Let γ be the least ordinal not in the range of* `stratum`*. A* local stratification *of a normal program P is a partition of $grd_L(P)$ given by*

$$grd_L(P) = \dot{\bigcup_{\alpha<\gamma}} P_\alpha$$

such that each clause $A \leftarrow L_1 \,\&\, \ldots \,\&\, L_n$ in $grd_L(P)$ is in P_α if, and only if, $\texttt{stratum}(A) = \alpha$.

As a notational convenience let

$$\bar{P}_\delta = \dot{\bigcup_{\alpha<\delta}} P_\alpha\,.$$

We now define $M(P)$ and $M(\bar{P}_\delta)$, which we will see, are the unique stable models of P and $\bar{P}_\delta$, respectively.

Definition 2.4: *Let*

$$grd_L(P) = \dot{\bigcup_{\alpha<\gamma}} P_\alpha\,.$$

be a local stratification of P. Put

$$\begin{array}{rcl} M(\bar{P}_0) & = & \emptyset \\ M(\bar{P}_{\alpha+1}) & = & \mathbf{T}_P \Uparrow \omega M(\bar{P}_\alpha),\ \textit{for each } \alpha < \gamma \\ M(\bar{P}_\lambda) & = & \bigcup_{\alpha<\lambda} M(\bar{P}_\alpha),\ \textit{for each limit ordinal } \lambda \leq \gamma \\ \multicolumn{3}{c}{\textit{and}} \\ M(P) & = & M(\bar{P}_\gamma) \end{array}$$

For P a locally stratified logic program, the ground instantiation of P is r.e. Hence, the dependency graph is also r.e. Thus a locally stratified partition of the Herbrand base of P can be constructed as follows: Repeatedly select the set of all minimal elements from the dependency graph, which constitutes a co-r.e. set, and let the next stratum be this set. Delete these elements from the graph to obtain an r.e. set. We can see that the procedure must terminate, by a cardinality argument, and in fact by standard recursion-theoretic techniques can be shown to terminate at a stage γ strictly below ω_1^{ck}.

Recall the notion of *stable model*, introduced by Gelfond and Lifschitz, [GL88].

We have the following result that precisely characterizes the complexity of the *class* of all stable models of P.

Proposition 2.1: Marek, Nerode and Remmel, [MNR91] *The class $St(P)$ of stable models of a normal logic program P is Π^0_2.* ■

Marek, Nerode and Remmel showed that the set of stable models of a normal program is, up to a 1 : 1 recursive renaming, set of infinite paths through a recursive tree, and hence forms a Π^0_2 class of sets. This is the idea behind the preceding proposition. By a basic recursion-theoretic observation, it follows that if a program is fortunate enough to have a unique stable model, then that model is hyperarithmetic, since the unique element of a singleton arithmetic class is necessarily hyperarithmetic. This is the situation with locally stratified programs as we see from the next proposition.

Proposition 2.2: Marek and Subrahmanian, [MS89] *Let*

$$grd_L(P) = \bigcup_{\alpha<\gamma}^{\bullet} P_\alpha$$

be a local stratification of P. Then $M(\bar{P}_\delta)$ is the unique stable model of P, for each $\delta \leq \gamma$. ■

The following two corollaries of the preceding result, which can be obtained by standard recursion-theoretic techniques, have not previously been observed.

Corollary 2.3: *$M(P)$ is hyperarithmetic.* ■

Corollary 2.4: *$\gamma < \omega_1^{\mathrm{CK}}$ if γ is the least ordinal for which a locally stratified partition of $grd_L(P)$ can be obtained.*

Proof: If γ is as in the hypothesis, then $M(\bar{P}_\delta)$ is sufficiently uniform for $M(\bar{P}_\gamma)$ not to be hyperarithmetic if $\gamma = \omega_1^{\mathrm{CK}}$.

■

The preceding proposition immediately raises the question of whether it has a suitable converse; namely, is every hyperarithmetic set given by the unique stable model of a locally stratified program? As we remarked in the introduction this is indeed the case, but one must be careful here. More precisely, one must say *every hyperarithmetic set is given by the interpretation of a predicate symbol in the unique stable model of a locally stratified program.* This in not merely an artifact of our technical definitions; rather, it is forced by results of recursion theory that show that not every hyperarithmetic set is *implicitly definable*, *i.e.* is not the member of a singleton arithmetic class [Kol91].

3 Representing Ordinal Notations

Following [Ro67] let W_z be the domain of the z^{th} unary partial recursive with respect to a suitable indexing, and define

$$Dom\colon 2^\omega \longrightarrow 2^\omega$$

by

$$Dom(X) = \{\, z : W_z \subseteq X \,\}\,.$$

Moreover, let

$$\begin{aligned} Dom\!\uparrow\!0(X) &= X \\ Dom\!\uparrow\!\alpha(X) &= \textstyle\bigcup_{\beta<\alpha} Dom(Dom\!\uparrow\!\beta(X)) \end{aligned}$$

Concerning this operator we have the following facts, [Ro67].

1. $Dom\!\uparrow\!\omega_1^{\mathrm{CK}}(\emptyset)$ is Π^1_1-complete.
2. $x \in Dom\!\uparrow\!\omega_1^{\mathrm{CK}}(\emptyset)$ iff $W_x \subseteq Dom\!\uparrow\!\omega_1^{\mathrm{CK}}(\emptyset)$.
3. $Dom \in \Pi^0_1$.
4. $Dom\!\uparrow\!\alpha(\emptyset)$ is hyperarithmetic for all $\alpha < \omega_1^{\mathrm{CK}}$. (*cf.* Theorem 3.13, [Hi78]).

The elements of $Dom\!\uparrow\!\omega_1^{\mathrm{CK}}(\emptyset)$ can be thought of as notations for constructive ordinals. For $z \in Dom\!\uparrow\!\omega_1^{\mathrm{CK}}(\emptyset)$ let $\|z\|$ be the least ordinal α such that $z \in Dom\!\uparrow\!\alpha+1(\emptyset)$. ("+1" since if $W_z = \emptyset$ then $\|z\| = 0$; also this permits us to have notations for limit ordinals.) Note that if $z \in Dom\!\uparrow\!\omega_1^{\mathrm{CK}}(\emptyset)$ then $z \notin Dom\!\uparrow\!\|z\|(\emptyset)$ but $W_z \subseteq Dom\!\uparrow\!\|z\|(\emptyset)$.

In the remainder of the paper we will assume that the only function symbols of the language L are the constant 0 and the unary function symbol s. Thus the Herbrand universe of L is $\{0, s(0), s(s(0)), \ldots, s^n(0), \ldots\}$. We further assume that L has sufficient predicate symbols to supply the predicate symbols required for the development in the remainder of the paper. It should be noted that only a fixed finite number of predicate symbols will be required. We will also need notation for substitutions. $E\{X \mapsto t\}$ denotes the expression that results from syntactically substituting term t for each occurrence of variable X in E. We will not have to substitute for more than one variable at a time, and we will not need to be concerned about substituting into the scope of bound variables. We regard variable occurrences in the presentations of programs as free. Thus, for example, when instantiate the variable Z in the programs below to a term t, we instantiate all occurrences of Z in P to t.

The second of the facts concerning Dom is important in particular for showing that for each $\alpha < \omega_1^{\mathrm{CK}}$ there is a locally stratified logic program P with predicate symbol p such

that

$$p(s^n(0)) \in M(P) \text{ iff } n \in Dom\uparrow\alpha\,(\emptyset)\,.$$

Moreover, we have the stronger result that P can be given uniformly in terms of α. Specifically:

Theorem 3.1: *A finite logic program can be found with a clause*

$$p(X,Z) \quad \leftarrow \quad q(X,Y,Z)$$

such that for each $z \in \mathbf{N}$*, if* P_z^+ *is* $P\{Z \mapsto s^z(0)\}$*, then*

$$P_z^+ \text{ is locally stratified iff } z \in Dom\uparrow\omega_1^{\mathrm{CK}}\,(\emptyset)$$

and whenever $z \in Dom\uparrow\omega_1^{\mathrm{CK}}\,(\emptyset)$

$$p(s^n(0), s^z(0)) \in M(P_z^+) \text{ iff } n \in Dom\uparrow\|k\|(\emptyset) \text{ for some } k \in \mathbf{N} \text{ such that } \|k\| < \|z\|\,.$$

The proof of the theorem depends critically on the ability to replace the bodies of clauses by clauses with only one literal in a way that strongly relates dependencies in a program to the program's semantics, *cf.* the next section. A similar approach could be taken by adapting the logic program representations of register machines devloped by Sheperdson [She91] and Nerode and Shore [NS93].

4 Binary Logic Programs

The dependency relation induced by a given program among ground atoms is not generally thought to be closely tied to the program's semantics because the dependency relation remains unchanged when conjunctions between literals in normal clause bodies are replaced by disjunctions. Nevertheless, we are able to present a technique that allows for a tight relationship between a program's semantics and its ground dependency relation, and use this as a basis for the construction that we use to establish the main results.

Definition 4.1 *A* binary *logic program is a definite clause program with at most one atom in any clause body.*

We assume from here on that logic programs are written over a first-order language whose Herbrand universe is generated by the constant symbol `0` and unary function symbol `s`. We adopt the following syntactic abbreviations. $\mathtt{s}^0\mathtt{(0)}$ stands for `0` and $\mathtt{s}^{n+1}\mathtt{(0)}$ stands for $\mathtt{s(s}^n\mathtt{(0))}$.

Definition 4.2: *Let P be a* normal *logic program* (i.e. *a program in which positive as well as negative literals may occur in the bodies of the program's clauses*), *and let ground(P) be the set of ground instances of the program's clauses with respect to the Herbrand universe of the language of P. Ground atom A* refers positively to *ground atom B (in P) if there is a clause in ground(P) of the form*

$$A \leftarrow L_1 \,\&\, \ldots \,\&\, B \,\&\, \ldots \,\&\, L_n$$

A refers negatively to *B if there is a clause in ground(P) of the form*

$$A \leftarrow L_1 \,\&\, \ldots \,\&\, \neg B \,\&\, \ldots \,\&\, L_n$$

A refers to *B if A refers to B either positively or negatively. (Note that A may both positively and negatively refer to B.) A* depends on *B if (A, B) is in the transitive closure of the* refers to *relation. A* positively depends on *B if (A, B) is in the transitive closure of the* refers positively to *relation. A* negatively depends on *B if there are atoms A' and B' such that A depends on A' or is A', B' depends on B or is B, and A' refers negatively to B'. We say that the pair (A, B) is in the* negative dependency *relation if A negatively depends on B. Atom A negatively depends* directly *on atom B if there is an atom A' such that A depends positively on A' or is A' and A' refers negatively to B. We say that the pair (A, B) is in the* direct negative dependency *relation if A negatively depends directly on B.*

Note that the direct negative dependency relation is well-founded if, and only if, the negative dependency relation is well-founded.

The following basic lemma relates well-foundedness of negative dependency to local stratification [Pr88].

Lemma 4.1: *Normal program P is locally stratified if, and only if, the negative dependency relation of P is well-founded.*

■

We next introduce the fundamental idea linking a program's dependency relation to its semantics.

Definition 4.3 *Let ground atom A depend positively on ground atom B with respect to program P. Then the pair (A, B) is said to be a* logical dependency *iff $P \cup \{B\} \models A$. A program is* dependency sound *if every pair of ground atoms in the positive dependency relation of P is a logical dependency.* ■

Proposition 4.2 *Every binary program is dependency sound.* ■

The next definition rigorously sets out the notion of two programs having equivalent least models with respect to certain predicates.

Definition 4.4 *Let L be a first order language and let P_1, P_2 be definite clause logic programs over L. Let L' be a sublanguage of L and suppose the restrictions of the least models of P_1 and P_2 to the Herbrand base of L' are the same. Then P_1 and P_2 are said to be* extensionally equivalent *with respect to L'.* ■

We are now in a position to usefully introduce the construction of a binary program that is extensionally equivalent to a given program.

Definition 4.5 *Let P be a definite clause program. Extend L to a language L' by adjoining a new function symbol f_p for each predicate symbol p in L other than $=$. f_p has the same arity as p. Corresponding to each atom $p(t_1, \ldots, p_n)$ of L, the* translation, *$f_p(t_1, \ldots, p_n)$ is a term of L'. In general, for each atom A of L, let t_A denote the translation of A. Corresponding to P the* binary extensional equivalent *Q of P is defined as follows. Extend L' by adjoining a new binary function symbol* `stack`*, a new binary function symbol* `cons` *and a new constant symbol* `nil`*. Corresponding to each program clause*

$$A \leftarrow B_1, \ldots, B_n$$

of P, form the clause

$$\texttt{stack}(\texttt{cons}(t_A, Y), Z)) \leftarrow \texttt{stack}(\texttt{cons}(t_{B_1}, \texttt{cons}(t_{B_2}, \ldots, cons(t_{B_n}, Y)\ldots)), Z)\,.$$

Q also contains a bridging clause *for each predicate symbol p:*

$$p(X_1, \ldots, X_n) \leftarrow \texttt{stack}(\texttt{cons}(f_p(X_1, \ldots, X_n), \texttt{nil}), f_p(X_1, \ldots, X_n))$$

Finally, Q contains the terminating *clause*

$$\texttt{stack}(\texttt{nil}, Z) \leftarrow .$$

We assume below that as needed we may uniformly rename the predicate symbols and the associated function symbols in binary extensional equivalent programs to ensure that distinct programs have have no predicate symbols in common whenever this device is needed. ■

Proposition 4.3 *The binary extensional equivalent of P is extensionally equivalent to P, with respect to the language of P.* ■

For the proof that we give of Theorem 3.1 in the next section, we will need the following proposition.

Proposition 4.4 *Let Q be the binary extensional equivalent of P and let A and B be ground atoms in the language of P. Then*

$$A \textit{ depends on } \mathtt{stack}(\mathtt{nil}, t_B) \textit{ iff } Q \models A \textit{ and } B \textit{ is } A\,.$$

■

5 Representing Hyperarithmetic Sets with Locally Stratified Programs

We are continuing the proof of Theorem 3.1. Observe that, given a constructive ordinal α, the following holds:

$$\forall x[x \in Dom\uparrow\alpha\,(\emptyset) \Leftrightarrow \exists\beta[W_x \subseteq Dom\uparrow\beta\,(\emptyset)\ \&\ \beta < \alpha]]$$

As is familiar from the usual logic programming representation we can represent this as a general program clause where the expression of subset containment is just an abbreviation of a universal formula:

$$x \in Dom\uparrow\alpha\,(\emptyset) \leftarrow W_x \subseteq Dom\uparrow\beta\,(\emptyset)\ \&\ \beta < \alpha$$

Next, we replace α and β by the elements m' and m of $Dom \uparrow \omega_1^{\mathrm{CK}}\,(\emptyset))$ which serve as notations for α and β. Specifically, $\|\,m\,\|{+}1$ is the least ordinal α such that $m \in Dom\uparrow\alpha\,(\emptyset)$. Thus:

$$x \in Dom\uparrow\|\,m'\,\|(\emptyset)\ \leftarrow W_x \subseteq Dom\uparrow\|\,m\,\|(\emptyset)\ \&\ \ \|\,m\,\|{<}\|\,m'\,\| \tag{1}$$

is a "definition" of $Dom\uparrow\alpha\,(\emptyset)$.

If we want to inductively construct $Dom\uparrow\alpha\,(\emptyset)$ for a constructive ordinal α, we need only to allow m, m' to range over a subset of $Dom\uparrow(\alpha+1)\,(\emptyset)$ sufficient for $\|\,m\,\|, \|\,m'\,\|$ to range over all ordinals up through α. Choose an r.e. set S so that

$$\exists m[\|\,m\,\|{<}\|\,m'\,\|\ \&\ m, m' \in S]$$

is also an r.e. set. To show that such an r.e. set exists (see also, e.g., [Ro67]). let

$$\Phi(X) = \{y : \exists x(y \in W_x\ \&\ x \in X)\}$$

Φ is a monotonic enumeration operator. Its role is similar to the union operator in set theory. In the same way as union, which after iteration ω times generates the transitive closure, iterated Φ provides us with all the notations needed to produce notations of all ordinals below $\|z\|$, given $z \in Dom\uparrow\omega_1^{CK}(\emptyset)$. Define the iteration sequence:

$$\Phi^0(X) = X$$
$$\Phi^{n+1}(X) = \Phi(\Phi^n(X)), \quad n \in \omega$$

and

$$\Phi^\omega(X) = \bigcup_{n\in\omega} \Phi^n(X)\,.$$

If $X \in \Sigma_1^0$, so is $\Phi^\omega(X)$ (cf. [Ro67]). Suppose $z \in Dom\uparrow\omega_1^{\mathrm{CK}}(\emptyset)$. Then, since Φ is monotone

$$\Phi^\omega(\{z\}) \subseteq Dom\uparrow\omega_1^{\mathrm{CK}}(\emptyset)$$

and

$$\{\|y\| : y \in \Phi^\omega(\{z\}))\} =_{\|z\|}$$

So now, for a constructive ordinal α we select $z \in Dom\uparrow(\alpha+1)(\emptyset)$. We now have:

$$\exists m[\|m\| < \|m'\| \ \&\ m, m' \in \Phi^\omega(\{z\})] \leftrightarrow \exists m[m \in \Phi^\omega(\{m'\}) \ \&\ m' \in \Phi^\omega(\{z\})]$$

The right side of the above is obviously Σ_1^0. Hence we can now rewrite the formula (1) as:

$$x \in Dom\uparrow\|m'\|(\emptyset) \ \leftarrow W_x \subseteq Dom\uparrow\|m\|(\emptyset) \ \&\ m \in \Phi^\omega(\{m'\}) \ \&\ m' \in \Phi^\omega(\{z\}) \qquad (2)$$

which holds if $z \in Dom\uparrow\omega_1^{\mathrm{CK}}(\emptyset)$.

We are introducing now an auxiliary program $\hat{P}_z$ as follows. By the results of [AN78, She91], for each k-ary recursively enumerable relation $R \subseteq \mathbf{N}^k$ we can find a definite clause program P_R with the predicate symbol p_R so that:

$$p_R(s^{a_1}(0), \ldots s^{a_k}(0)) \in \mathbf{T}_{P_R}\uparrow\omega(\emptyset) \text{ if and only if } R(a_1, \ldots, a_k).$$

Let R_1, R_2 be the relations given by

$$R_1(m, m') \Leftrightarrow m \in \Phi^\omega(\{m'\})$$

$$R_2(y, x) \Leftrightarrow y \in W_x$$

Finally, let $q(x, m)$ abbreviate "$x \in Dom\uparrow\|m\|(\emptyset)$". $\hat{P}_z$ is, then, a general program consisting of general program clause:

$$q(X, M') \leftarrow \forall Y[p_{R_2}(Y, X) \rightarrow q(Y, M)] \ \&\ p_{R_1}(M, M') \ \&\ p_{R_1}(M', s^z(0)) \qquad (3)$$

together with the definite clauses of the programs P_{R_i}, $i = 1, 2$.

We now transform the clause (3) to a set of normal program clauses by an operational technique for eliminating universal quantifiers in general program clauses bodies. We replace clause (3) with two clauses:

$$q(X, M') \leftarrow \neg w(X, M) \ \& \ p_{R_1}(M, M') \ \& \ p_{R_1}(M', s^z(0)), \tag{4}$$

$$w(X, M) \leftarrow p_{R_2}(Y, X) \ \& \ \neg q(Y, M)$$

and let Q_z be a normal program consisting of clauses (4) and the definite clauses of P_{R_1} and P_{R_2}. We can also assume (by renaming predicate symbols if necessary) that P_{R_1} and P_{R_2} have no predicate symbols in common.

The programs Q_z are *not* locally stratified. Indeed, $q(0, 0)$ refers negatively to $w(0, 0)$ which in turn refers negatively $q(0, 0)$ by setting X, M, M', and Y to 0 in the clauses (4). To remedy this we will employ binary extensional equivalent programs.

Consider the recursively enumerable relation R_0 defined by

$$R_0(x, m, m', z) \text{ iff } R_1(m, m') \ \& \ R_1(m', z)$$

(The role of x will be apparent momentarily.) Let P_{R_0} be a definite clause program that computes the r.e. relation R_0. Thus, for some predicate symbol p_{R_0} in the language of P_{R_0},

$$P_{R_0} \models p_{R_0}(s^x(0), s^m(0), s^{m'}(0), s^z(0)) \text{ iff } R_0(x, m, m', z)\,.$$

Let Q_{R_0} be the binary extensional equivalent of P_{R_0}, but where the program's terminating unit clause

$$\texttt{stack}(\texttt{nil}, Z) \leftarrow$$

is replaced by the clause

$$\texttt{stack}(\texttt{nil}, f_{p_{R_0}}(X, M, M', Z)) \leftarrow \neg w(X, M, Z) \tag{5}$$

Similarly, let $P_{R'_2}$ be a definite clause program that computes the r.e. relation R'_2 defined by

$$R'_2(x, y, m, z) \text{ iff } R_2(y, z)\,,$$

and let $Q_{R'_2}$ be the binary extensional equivalent of $P_{R'_2}$, but where the program's terminating clause is replaced by

$$\texttt{stack}(\texttt{nil}, f_{p_{R'_2}}(X, Y, M, Z)) \leftarrow \neg q(Y, M, Z)\,. \tag{6}$$

We assume that by renaming predicate symbols as necessary that Q_{R_0} and $Q_{R'_2}$ have no predicate symbols in common other than w and q. For each $z \in \mathbf{N}$ form the program P_z as follows. The clauses of P_z are:

$$q(X, M', s^z(0)) \leftarrow p_{R_0}(X, M, M', s^z(0)) \tag{7}$$

$$w(X, M, s^z(0)) \leftarrow p_{R'_2}(X, M, Y, s^z(0)) \tag{8}$$

together with the clauses for P_{R_0} and $P_{P'_2}$ with $s^z(0)$ substituted for Z in clauses 5 and 6. We may assume that the variable Z is not used in any of the other clauses in programs P_{R_0} and $P_{R'_2}$. Notice that the predicate $q(\cdot, \cdot, \cdot)$ is a variation of the predicate q of the program Q_z. There we could not control enough the dependence relation and so it was not a locally stratified program. Here, the situation changes. We have now the following crucial fact.

Proposition 5.1 *$z \in Dom \uparrow \omega_1^{\mathrm{CK}}(\emptyset)$ if, and only if, the program P_z is locally stratified.*

Proof: It suffices to show that the negative dependency relation of P_z is well-founded whenever $z \in Dom \uparrow \omega_1^{\mathrm{CK}}(\emptyset)$.

Suppose that there is a sequence $A_0, A_1, \ldots$ of ground atoms of the language of program P_z such that for every $k \geq 1$

$$A_k \textit{ negatively} \text{ depends on } A_{k-1}.$$

Notice that in the graph of the *refers to* relation all negative edges arise only via ground instances of clauses 5 and 6. Thus this sequence yields a sequence of ground atoms

$$q(s^{x_0}(0), s^{m_0}(0), s^z(0)), \quad q(s^{x_1}(0), s^{m_1}(0), s^z(0)), \ \ldots$$

such that

$$q(s^{x_k}(0), s^{m_k}(0), s^z(0))$$

negatively depends directly on

$$w(s^{x_k}(0), s^{m_{k+1}}(0), s^z(0))$$

which negatively depends directly on

$$q(s^{x_{k+1}}(0), s^{m_{k+1}}(0), s^z(0))$$

Thus, for all $k \in \mathbf{N}$,

$$R_1(m_{k+1}, m_k) \ \& \ R_1(m_k, z)$$

and hence, since $z \in Dom \uparrow \omega_1^{\mathrm{CK}}(\emptyset)$,

$$\| m_{k+1} \| < \| m_k \| < \| z \| .$$

This yields an infinite descending chain of constructive ordinals and hence a contradiction.

For the converse, note that if $z \notin Dom \uparrow \omega_1^{\mathrm{CK}}(\emptyset)$, then there is a sequence

$$z,\ z_1,\ z_2,\ \ldots$$

of natural numbers such that $z_{k+1} \in W_{z_k}$, for all $k \in \mathbf{N}$. It follows that

$$q(s^{z_k}(0), s^{z_k}(0), s^z(0))$$

negatively depends directly on

$$q(s^{z_{k+1}}(0), s^{z_{k+1}}(0), s^z(0)) \text{ for all } k \geq 1.$$ ■

In proposition 5.1 we established that the program P_z is locally stratified (whenever $z \in Dom \uparrow \omega_1^{\mathrm{CK}}(\emptyset)$). Therefore, for such z, P_z possesses the perfect model, $M(P_z)$. We shall now investigate the extension of predicate q in those models.

Proposition 5.2 *Let $z \in Dom \uparrow \omega_1^{\mathrm{CK}}(\emptyset)$). The following are equivalent:*
(a) $q(s^x(0), s^m(0), s^z(0))$ belongs to $M(P_z)$
(b) $m \in \Phi^\omega(\{z\})$ and $x \in Dom \uparrow \| m \|(\emptyset)$

Proof: First observe

$$q(s^x(0), s^{m'}(0), s^z(0)) \in M(P_z)$$

if and only if

$$\text{for some } m \in \mathbf{N},\ p_{R_0}(s^x(0), s^m(0), s^{m'}(0), s^z(0)) \in M(P_z)$$

if and only if

for some $m \in \mathbf{N}$, $\mathtt{stack}(\mathtt{nil}, f_{p_{R_0}}(s^x(0), s^m(0), s^z(0))) \in M(P_z)$
and $R_1(m, m')$ and $R_1(m', z)$.

if and only if

for some $m \in \mathbf{N}$, $M(P_z) \models \neg w(s^x(0), s^m(0), s^z(0))$ and
$m \in \Phi^\omega(\{m'\})$ and $m' \in \Phi^\omega(\{z\})$.

But

$$w(s^x(0), s^m(0), s^z(0)) \in M(P_z)$$

if and only if

for some $y \in \mathbf{N}$, $\mathtt{stack}(\mathtt{nil}, f_{p_{R_0}}(s^x(0), s^m(0), s^y(0), s^z(0))) \in M(P_z)$

if and only if

for some $y \in \mathbf{N}$, $\mathtt{stack}(\mathtt{nil}, f_{p_{R_0}}(s^y(0), s^m(0), s^z(0))) \in M(P_z)$
and $y \in W_x$

if and only if

for some $y \in \mathbf{N}$, $y \in W_x$ *and* $M(P_z) \models \neg q(s^y(0), s^m(0), s^z(0))$.

Therefore, $q(s^x(0), s^{m'}(0), s^z(0)) \in M(P_z)$

if and only if

$$\begin{aligned} &\textit{for some } m \in \mathbf{N} : \ \textit{for all } y \in \mathbf{N} : \ m \in \Phi^\omega(\{m'\}) \textit{ and } m' \in \Phi^\omega(\{z\}) \\ &\qquad\qquad \text{and } (y \notin W_x \textit{ or } M(P_z) \models q(s^y(0), s^m(0), s^z(0)). \end{aligned} \tag{9}$$

We now proceed by transfinite induction.
Case 1. $\| z \| \models 0$. Then $W_z = \emptyset$, so by the immediately preceding equivalence neither of the conditions $m' \in \Phi^\omega(\{z\})$ nor $q(s^x(0), s^{m'}(0), s^z(0)) \in M(P_z)$ can hold, for all $m', x \in \mathbf{N}$.

Case 2. $\| z \| \rhd 0$. Suppose $q(s^x(0), s^{m'}(0), s^z(0)) \in M(P_z)$. The by property (9) and the induction hypothesis:

for some $m \in \mathbf{N}$, for all $y \in \mathbf{N}$, $m \in \Phi^\omega(\{m'\})$ and $m' \in \Phi^\omega(\{z\})$ and $y \in W_x \rightarrow$
$y \in Dom \uparrow \| m \| (\emptyset)$,

which can be restated as

for some $m \in \mathbf{N}$, $m \in \Phi^\omega(\{m'\})$ and $m' \in \Phi^\omega(\{z\})$ and $W_x \subseteq Dom \uparrow \| m \| (\emptyset)$.

Hence,

for some $m \in \mathbf{N}$: $m \in \Phi^\omega(\{m'\})$ and $m' \in \Phi^\omega(\{z\})$
and $x \in Dom \uparrow (\| m \| + 1) (\emptyset) \subseteq Dom \uparrow \| m' \| (\emptyset)$.

Thus

$q(s^x(0), s^{m'}(0), s^z(0)) \in M(P_z)$ is equivalent to $x \in Dom \uparrow \| m' \| (\emptyset)$ and $m' \in \Phi^\omega(\{z\})$. ■

We are finally in the position to complete the proof of Theorem 1. To obtain P include three clauses:

$$p(X, Z) \leftarrow q(X, Y, Z) \tag{10}$$

and

$$q(X, M', Z) \leftarrow p_{R_0}(X, M, M', Z)$$

$$w(X, M, Z) \leftarrow p_{R'_2}(X, M, Y, Z)$$

together with the clauses for P_{R_0} and $P_{R'_2}$.

If we instantiate P to $P\{Z \mapsto s^z(0)\}$, (i.e. equivalently, we instantiate clauses (10) using the substitution $\{Z \mapsto s^z(0)\}$ and add the clauses of the program P_z), the resulting program, which we denote P_z^+, remains locally stratified by Proposition 5.2, provided z is in $Dom \uparrow \omega_1^{\mathrm{CK}}(\emptyset)$. Moreover, by Proposition 5.2:

$$p(s^x(0), s^z(0)) \in M(P)$$

if and only if

$$\textit{for some } m \in \mathbf{N},\ q(s^x(0), s^m(0), s^z(0)) \in M(P_z)$$

if and only if

$$\textit{for some } m \in \mathbf{N},\ m \in \Phi^\omega(\{z\}) \textit{ and } x \in Dom \uparrow \| m \|(\emptyset)$$

if and only if

$$\textit{for some } m \in \mathbf{N},\ \| m \| < \| z \| \textit{ and } x \in Dom \uparrow \| m \|(\emptyset)$$

If z is not in $Dom \uparrow \omega_1^{\mathrm{CK}}(\emptyset)$ then, by proposition 5.1, P_z and hence P_z^+ is not locally stratified. ■

The following corollary to theorem 1 is immediate.

Corollary 5.3 *A finite logic program P can be found with a clause*

$$p(X, Y) \quad \leftarrow \quad q(X, Y, Z)$$

such that for $z \in Dom \uparrow \omega_1^{\mathrm{CK}}(\emptyset)$ where $\| z \|$ is a limit ordinal,

$$p(s^x(0), s^z(0)) \in M(P\{Z \mapsto s^z(0)\}) \textit{ if, and only if, } x \in Dom \uparrow \| z \|(\emptyset).$$

■

By adding a few more clauses which comprise a program for computing a certain partial recursive function we have the following variation of the main theorem.

Corollary 5.4 *A program Q can be found with binary predicate symbol r such that*

$$z \in Dom\!\uparrow\!\omega_1^{\mathrm{CK}}(\emptyset) \text{ if, and only if, } Q\{Z \mapsto s^z(0)\} \text{ is locally stratified,}$$

and whenever $z \in Dom\!\uparrow\!\omega_1^{\mathrm{CK}}(\emptyset)$

$$r(s^x(0), s^z(0)) \in M(Q\{Z \mapsto s^z(0)\}) \text{ iff } x \in Dom\!\uparrow\!\|z\|+1(\emptyset).$$

Proof: Obtain Q be adding a set of definite clauses to program P given by the main theorem with a new binary predicate symbol r (i.e. r does not occur in P) so that

$$r(s^x(0), s^z(0)) \in M(Q) \text{ iff } p(s^x(0), s^{\varphi(z)}(0)) \in M(P)$$

where φ is a partial recursive function chosen so that $W_{\varphi(n)} = \{n\}$, for all $n \in \mathbf{N}$. It follows that $\|\varphi(z)\| = \|z\|+1$ whenever $z \in Dom\!\uparrow\!\omega_1^{\mathrm{CK}}(\emptyset)$. ■

The proof of Proposition 5.1 permits us to read off an explicit local stratification of P_z^+, whenever $z \in Dom\!\uparrow\!\omega_1^{\mathrm{CK}}(\emptyset)$. We partition the Herbrand base B_P as follows:

For each ground atom $q(t_1, t_2, t_3)$, if $t_3 = s^z(0)$ and $t_2 = s^{m'}(0)$ for some $m' \in \Phi^\omega(\{z\})$, then $\|m'\| = \lambda + k$ for some limit ordinal λ and finite k. We put $q(t_1, t_2, t_3)$ in the stratum $H_{\lambda+2k}$; otherwise $q(t_1, t_2, t_3)$ is placed in H_0. Notice that if $t_3 = s^z(0)$ then the stratum in which $q(t_1, t_2, t_3)$ is placed is determined by t_2.

Suppose an atom A negatively depends directly on $q(t_1, t_2, t_3)$. Then

A is $w(s^x(0), t_2, t_3)$ for some $x \in \mathbf{N}$, or
A is $p_{R_2'}(s^x(0), t_1, t_2, t_3)$ for some $x \in \mathbf{N}$, or
A is $\mathtt{stacks}, f_{p_{R_2'}}(s^y(0), t_1, t_2, t_3))$ for some $x \in \mathbf{N}$.

Thus all atoms $q(t_1, t_2, t_3)$ on which A negatively depends directly are in the same stratum, if A is of one of the three forms immediately above. Thus, if A negatively depends directly on $q(t_1, t_2, t_3)$ and $q(t_1, t_2, t_3)$ is in stratum H_α then place A in stratum $H_{\alpha+1}$.

In particular, all ground $w(u_1, u_2, u_3)$ that do not negatively depend directly on any $q(t_1, t_2, t_3)$ are placed in H_0.

In this way we obtain a partition

$$B_P \setminus \{p(s^x(0), p^z(0)) : x \in \mathbf{N}\} = \bigcup_{\alpha<\gamma} H_\alpha$$

(where γ is the supremum of ordinals used in the construction).

Finally define

$$H_\gamma = \{p(s^x(0), s^z(0)) : x \in \mathbf{N}\}$$

So

$$B_P = \bigcup_{\alpha<\gamma+1} H_\alpha$$

We can now verify that the conditions of local stratification are met by the partition, straightforwardly.

When we trace the ordinals used in the construction, we see that two cases may occur: either $\| z \|$ is limit, or for a unique limit λ and $k \in \mathbf{N}$, $\| z \| = \lambda + k + 1$. In the first case $\gamma = \lambda$, in the second $\gamma = \lambda + 2k$. In the either case

Theorem 5.5

$$ground(P_z^+) = \bigcup_{\alpha\leq\gamma+1} \overline{P_\alpha}$$

where $\gamma = lub\{\lambda + 2k : \lambda + k < \| z \|$, λ *limit,* k *finite* $\}$. ■

By standard techniques of recursion theory on can show that for every hyperarithmetic set A these is a constructive ordinal α such that A is 1:1 reducible to $Dom \uparrow \alpha\,(\emptyset)$. In outline, this is shown by 1:1 reducing A to a well-founded recursion-theoretic *tree* $\mathcal{T}_\beta$ [*cf.* theorem XXIIa in [Ro67], chapter 16] and then reducing $\mathcal{T}_\beta$ to a subset of $Dom \uparrow \alpha\,(\emptyset)$, for some constructive α whose size can be estimated in terms of β.

Theorem 5.6 *For every hyperarithmetic set A there exists a locally stratified program P_A, with a predicate symbol r such that*

$$r(s^x(0)) \in M(P_A) \text{ iff } x \in A\,.$$

Proof: Let A be hyperarithmetic. Then for some $\alpha < \omega_1^{\mathrm{CK}}$, $A \leq_1 Dom \uparrow \alpha\,(\emptyset)$. Let f be a recursive function used in this reduction. Let P_f be a Horn program which computes the graph of f. We assume that the set of predicate symbols of P_f is disjoint with that of our program P whose instantiations P_z^+ define $Dom \uparrow \| z \|(\emptyset)$. Moreover we assume that the predicate $gr(\cdot,\cdot)$ computes the graph of f.

Now, we select z such that $\| z \| = \alpha$ and a new unary predicate symbol q, and write a program P_A which is the union of two programs:

P_z
P_f

and the clause

$q(X) \leftarrow gr(X, Y),\ p(Y, s^z(0))$.

(where p is the predicate symbol defining $Dom\uparrow\alpha(\emptyset)$ in P_z^+.)

Notice that P_A is locally stratified. The local stratification of the part of the language corresponding to P_z^+ is preserved, atoms of the form $q(t)$ are put in a new stratum above the strata of p, and the atoms of the form $gr(t_1, t_2)$ and other ground atoms of the language of P_f can be put anywhere, as long as they are put in a single, existing, stratum.

Next, notice that $q(s^n(0)) \in M(P_A)$ if, and only if, for some m, $gr(s^n(0), s^m(0)) \in M(P_A)$ and $p(s^m(0), s^z(0)) \in M(P_A)$. But this is equivalent to

$f(n) = m\ \&\ m \in Dom\uparrow\alpha(\emptyset))$, i.e. $n \in A$. ■

6 The Well-Founded Semantics

The well-founded partial model for a program **P** is defined by transfinite induction. One very commonly used construction uses the alternating fixed point operator of Van Gelder [VG89]; we follow more closely the presentation of [BS91]. For any logic program P an r.e. operator $\mathbf{F}_P$ is defined (the Gelfond-Lifschitz operator for the stable semantics). Now $\mathbf{F}_P$ is anti-monotonic, so $\mathbf{F}_P^2$ is monotonic. The well-founded semantics infers a ground atom β iff β is in the least fixed point of $\mathbf{F}_P^2$, and it infers $\neg\beta$ iff β is not in the greatest fixed point of $\mathbf{F}_P^2$. These fixed points can be constructed by transfinite recursion; here L is the language of P:

$$\begin{aligned}
\mathbf{W}_P^+\Uparrow < \eta &= \bigcup_{\nu<\eta} \mathbf{W}_P^+\Uparrow\nu. \qquad (\text{Hence } \mathbf{W}_P^+\Uparrow < 0 = \emptyset)\\
\mathbf{W}_P^+\Uparrow\eta &= \mathbf{F}_P^2(\mathbf{W}_P^+\Uparrow\eta)\\
\mathbf{W}_P^-\Uparrow < \eta &= \bigcap_{\nu<\eta} \mathbf{W}_P^-\Uparrow\nu. \qquad (\text{Hence } \mathbf{W}_P^-\Uparrow < 0 = B_L)\\
\mathbf{W}_P^-\Uparrow\eta &= \mathbf{F}_P^2(\mathbf{W}_P^-\Uparrow\eta).
\end{aligned}$$

Now $x \in \mathbf{F}_P(I)$ can be defined over the natural numbers by a first order formula

$$\exists y_1, \ldots, y_k, z(\phi(x, y_1, \ldots, y_k, z) \wedge y_1 \notin I \wedge \cdots \wedge y_k \notin I$$

where I does not appear in ϕ [Sch90]. Substitution gives a first order formula defining $x \in \mathbf{F}_P^2(I)$ in which I appears only positively. Hence both inductions reach fixed points in $\leq \omega_1^{ck}$ steps; call the fixed points $\mathbf{W}_P^+\Uparrow\infty$ and $\mathbf{W}_P^-\Uparrow\infty$. Then $\mathbf{W}_P^+\Uparrow\infty$ is the least fixed point of $\mathbf{F}_P^2$, and $\mathbf{W}_P^-\Uparrow\infty$ is the greatest fixed point. (In fact, it is well-known that both reach fixed points in the same number of steps.) Also, $\mathbf{W}_P^+\Uparrow\infty$ is Π_1^1 definable, and $\mathbf{W}_P^-\Uparrow\infty$ is Σ_1^1 definable.

The well-founded partial model of P is total if $\mathbf{W}_P^+ \Uparrow \infty = \mathbf{W}_P^- \Uparrow \infty$, that is, if for each ground atom β, either β or $\neg\beta$ is inferred.

Theorem 6.1 *A set is definable in the well-founded semantics by a program P whose well-founded partial model is total iff it is hyperarithmetic.*

Proof: If the well-founded partial model of P is total, then $\mathbf{W}_P^+ \Uparrow \infty = \mathbf{W}_P^- \Uparrow \infty$ is, by the above discussion, both Π_1^1 and Σ_1^1 definable, i.e., it is hyperarithmetic. On the other hand, if a set is hyperarithmetic definable, then it is definable by a locally stratified program P, by Theorem 5.6, and, by [VGRS91], the well-founded partial model of P is total and $\mathbf{W}_P^+ \Uparrow \infty$ is the perfect model.

Theorem 6.2 *Suppose the well-founded partial model of a program P is total. Then there is a locally stratified program Q where, for each ground atom β of P, L is in the well-founded partial model of P iff β is in the perfect model of Q.*

Proof: Since the well-founded partial model is total, the set of ground atoms true in it is hyperarithmetic. By Theorem 5.6, such a program Q exists.

Moreover, using the methods of the previous section, we can simulate the well-founded semantics up to any constructive ordinal stage in the inductive construction – uniformly in a notation for the ordinal. Recall that the well-founded partial model is not necessarily total. Thus, to represent with a locally stratified program, we must represent positive and negative literals separately; we shall add relation symbols p^+ and p^- for each relation symbol p of P.

Theorem 6.3 *Let P be a logic program. Then there is a logic program Q with an extra variable Z so that, when Z is instantiated to $s^z(0)$ for $z \in Dom \Uparrow \omega_1^{ck}$,*

- *the instantiated program is locally stratified, and*
- *for any ground atom $p(\vec{t})$ of P, $p(\vec{t}) \in \mathbf{W}_P^+ \Uparrow < \|z\|$ iff $p^+(\vec{t})$ is in the perfect model of the instantiated program, and*
- *for any ground atom $p(\vec{t})$ of P, $p(t) \notin \mathbf{W}_P^- \Uparrow < \|z\|$ iff $p^-(\vec{t})$ is in the perfect model of the stratified program.*

References

[AN78] Andreka H. and Nemeti I., "The Generalized Completeness of Horn Predicate Logic as a Programming Language", *Acta Cybernetica*, vol. 4, pp.3–10, 1978

[Apt90] Apt, K.R., "Logic Programming" *Handbook of Theoretical Computer Science*. J. van Leeuven, (ed.) Elsevier Science Publishers, pp.495–574, 1990.

[ABe90] Apt, K.R. and Bezem, M. "Acyclic Programs". Centre for Mathematics and Computer Science Technical Report CS-R9010, Amsterdam, 1990.

[ABW88] Apt, K. R., Blair, H. A., & Walker, A. "Towards a Theory of Declarative Knowledge," in *Foundations of Deductive Databases and Logic Programming,* Jack Minker, ed. Morgan-Kaufmann, Los Altos, CA. 1988, pp. 89–148.

[AB90] Apt, K.R. and Blair, H.A., "Arithmetic Classification of Perfect Models of Stratified Programs", *Fundamenta Informaticae*, XIII, pp. 1–17, 1990.

[AB91] Apt, K.R. and Blair, H.A., "Arithmetic Classification of Perfect Models of Stratified Programs - addendum", *Fundamenta Informaticae*, XIV(3), pp. 339–344, 1991.

[BS91] Baral, C. and Subrahmanian, V. S. "Dualities between alternative semantics for logic programming and non-monotonic reasoning." In *Logic Programming and Non-monotonic Reasoning: Proc. of the First International Workshop*, Nerode, A., Marek, W. and Subrahmanian, V. S., editors. MIT Press, Cambridge, Massachusetts, pp. 69-87, 1991.

[BF88] Bidoit, N. and Froixdevaux, Ch., "Negation by Default and Non Stratifiable Logic Programs", Internal Report 437, Universite Paris-Sud, 1988.

[Bl82] Blair, H. A. "The Recursion-Theoretic Complexity of the Semantics of Predicate Logic as a Programming Language." *Information and Control*, July-August, 1982, pp. 25–47.

[CB91] Cholak, C. and Blair, H. "The Complexity of Local Stratification." Manuscript, 1991. (Submitted).

[vEK76] van Emden M.H. and Kowalski, R.A. "The Semantics of Predicate Logic as a Programming Language", *Journal of the ACM*, vol. 23(4), pp. 733–742, 1976

[Ge89] Gelfond, M. "Autoepistemic Logic and Formalization of Commonsense Reasoning", *Non-Monotonic Reasoning*, M. Reinfrank, J. de Kleer, M.L. Ginsberg

and E. Sandewall (eds.), Lecture Notes in Artificial Intelligence, 346. Springer-Verlag, pp.176–186, 1989.

[GL88] Gelfond, M. and Lifschitz, V. "The Stable Model Semantics for Logic Programming," *Proc. ICLP/SLP-5*, 1988, pp.1070–1080.

[Hi78] Hinman, P.G. *Recursion-Theoretic Hierarchies*. Springer-Verlag, Perspectives in Mathematical Logic, 1978.

[Kol87] Kolaitis, P.G. "The Expressive Power of Stratified Logic Programs." Manuscript, Nov. 1987.

[Kol91] *Private Communication.*

[Kow74] Kowalski, R.A. "Predicate Logic as a Programming Language." *Proceedings IFIP Congress, 1974*. North-Holland, Amsterdam, 1974. pp 569–574.

[Li88] Lifschitz, V. "On the Delcarative Semantics of Logic Programs with Negation." *Foundations of Deductive Databases and Logic Programming,* Jack Minker, ed. Morgan-Kaufmann, Los Altos, CA. 1988, pp. 172-192.

[Ll87] Lloyd, J.W. *Foundations of Logic Programming*, (2nd. ed.) Springer-Verlag, 1987.

[MNR90] Marek, W., Nerode, A. and Remmel, J. *A Theory of Nonmonotonic Rule Systems*, MSI Technical Report 90-31, Mathematical Sciences Institute, Cornell University.

[MNR91] Marek, W., Nerode, A. and Remmel, J., *How Complicated Is the Set of Stable Models of a General Logic Program?*, Technical Report, Mathematical Sciences Institute, Cornell University, 1991.

[MS89] Marek, W. and Subrahmanian, V.S. "The Relationship Between Logic Program Semantics and Non-monotonic reasoning", in *Proceedings of the 6th International Conference on Logic Programming.* G. Levi and M. Martelli (eds.), MIT Press, pp.600–617, 1989.

[MT89] Marek, W. and Truszczyński, M. "Stable Semantics for Logic Programs and Default Theories", *Proceedings of the North American Conference on Logic Programming*, MIT Press, Cambridge, MA. pp. 243–256, 1989.

[NS93] Nerode, Anil & Shore, Richard, *Logic for Applications*, Springer-Verlag, 1993.

[Pr88] Przymusinski, T. "On the Declarative Semantics of Deductive Databases and Logic Programs," in *Foundations of Deductive Databases and Logic Programming,* Jack Minker, ed. Morgan-Kaufmann, Los Altos, CA. 1988,

[Pr89] Przymusinski, T. "Every Logic Program Has a Natural Stratification and an Iterated Fixed Point Model," in *Eighth ACM Symposium on Principles of Database Systems*, pages 11-21, 1989.

[Ro67] Rogers, H. *Theory of Recursive Functions and Effective Computability*, McGraw-Hill, 1967.

[Sch90] Schlipf, J. "The Expressive Powers of the Logic Programming Semantics," to appear in *JCSS*. A preliminary version appeared in *Ninth ACM Symposium on Principles of Database Systems*, 1990.

[She91] Shepherdson, J. C. *Unsolvable Problems for SLDNF-Resolution*, *J. of Logic Programming*, 10(1), Jan. 1991, pp. 19–22.

[SS63] Shepherdson, J. C. and Sturgis, H. E. "Computability of Recursive Functions," *JACM*, **10**, 217–255, 1963.

[Smu61] Smullyan, R.M., *Theory of Formal Systems*, Princeton University Press, Princeton, NJ 1961.

[VG88] Van Gelder, A. "Negation as Failure Using Tight Derivations for General Logic Programs," in *Foundations of Deductive Databases and Logic Programming,* Jack Minker, ed. Morgan-Kaufmann, Los Altos, CA. 1988, pp. 149–176.

[VG89] Van Gelder, A. "The Alternating Fixpoint of Logic Programs with Negation," in *Eighth ACM Symposium on Principles of Database Systems*, pages 1-10, 1989.

[VGRS91] Van Gelder, A., Ross, K., and Schlipf, J. "The Well-Founded Semantics for General Logic Programs." *Journal of the ACM 38*, 1991, pp. 620-650.