Locally Determined Logic Programs and Recursive Stable Models

A. Douglas Cenzer (cenzer@ufl.edu)
University of Florida

B. Jeffrey B. Remmel (jremmel@ucsd.edu)
University of California at San Diego

C. Amy Vanderbilt (vanderbilt@xu.edu)
Xavier University

Abstract. In general, the set of stable models of a recursive logic program can be quite complex. For example, it follows from results of Marek, Nerode, and Remme [17] that there exists finite predicate logic programs and recursive propositional logic programs which have stable models but no hyperarithmetic stable models. In this paper, we shall define several conditions which ensure that a recursive propositional logic program P has a stable model which is of low complexity, that is, a recursive stable model, a polynomial time stable model, or a stable model which lies in a low level of the polynomial time hierarchy.

Keywords: nonmonotonic logic, logic programming

68T27: 03B70

1. Introduction

The stable model semantics of logic programs has been extensively studied. Unfortunately, the set of stable models of a recursive propositional logic program with negation or even a finite predicate logic program with negation can be quite complex. For example, in [18], Marek, Nerode, and Remmel showed that for any recursive propositional logic program P, there is an infinite branching recursive tree T such that there is an effective one-to-one degree preserving correspondence between the set of stable models of P and the set of infinite paths through T. Marek, Nerode, and Remmel [17] also showed that given any infinite branching recursive tree T, there exists a recursive propositional logic program P_T such that there is an effective one-to-one correspondence between the set of infinite paths through T and the set of stable models of P. Moreover, in [17], it is shown that the same results hold if we replace recursive propositional logic programs by finite predicate logic programs. It follows that the set of possible degrees of the stable models of a recursive propositional logic program or a finite predicate logic program are precisely the set of possible degrees of the infinite paths through some recursive infinitely branching tree. These results imply that the set of stable models of a recursive propositional logic program or a finite predicate logic program can be extremely complex. For example, it follows from these results that there a finite predicate logic program which has a stable model but which has no stable model that is hyperarithmetic.

The main motivation of the paper is to find conditions on recursive propositional logic programs P which guarantee the existence of well behaved stable for P, i.e. a stable model of P which is recursive, polynomial time, or in some low level of the polynomial time hierarchy. Since we shall deal mainly with propositional logic programs, we shall refer to propositional logic programs as simply logic programs. We should note that are a number of conditions in the literature which guarantee that a recursive logic program has a stable model of relatively low complexity with respect to the arithmetic hierarchy. Clearly, the first such condition is to consider recursive Horn logic programs. In that case, it is implicit in [24] and explicitly proved in [1] that the least model of a recursive Horn program is a recursively enumerable (r.e.) set and that every r.e. set can appear as the least model of recursive Horn Program. Another important class of logic programs is stratified logic programs [3], where one can single out a particular model called the perfect model which is the unique stable model of the program. In that case, Apt and Blair [4] showed that recursive logic program with n strata must have a perfect model which is Σ^0_n and that there is a recursive logic program P with n strata such that the perfect model of P is Σ^0_n complete. In [17], Marek, Nerode, and Remmel showed that imposing the conditions which say that (i) each atom of the Herbrand Base of P has at most finitely many minimal derivations from P (P is *locally finite* in the language of Marek, Nerode, and Remmel), and (ii) one can effectively find these possible derivations (P is an *rsp program* in the language of Marek, Nerode, and Remmel), ensures that there is highly recursive tree T such that there is an effective one-to-one correspondence between the set of stable models of P and the set of infinite paths through T. Here a tree T is *highly recursive*, if T is recursive, finitely branching, and there is a effective procedure which given any node $\eta \in T$, produces the set of nodes in T which immediately extend η. One consequence of this fact is that a recursive rps logic program which has a stable model always has a stable model M whose jump is recursive in $\mathbf{0}'$. In addition, Marek, Nerode, and Remmel [20] generalized Reiter's concept of normal default theories to logic programs (*FC-normal logic programs* in the language of [20]) and showed that FC-normal logic programs always have a stable model which is r.e. in $\mathbf{0}'$.

There are several key concepts introduced in this paper. One of the main reasons that the set of stable models of a recursive logic program can be so complex is that the analogue of the compactness theorem which holds for propositional logic or first order predicate logic fails badly for logic programs, see [21]. There are several ways to recover from this failure and to introduce conditions which ensure that a kind of compactness result holds for a logic program. One example of this type of condition is the notion of locally finite logic programs described above. In this paper, we introduce another such condition called *locally determined logic program.* Essentially, a logic program P with Herbrand base $H_P = \{a_0 < a_1 < \ldots\} \subseteq \omega$ is locally determined if for any $a_i \in H_P$, there exists an $n_i \geq i$ such that the question of whether any a_j with $j \leq n_i$ lies in a stable model E is determined only by the clauses of P which involve elements from $\{a_0, \ldots, a_{n_i}\}$. Such an n_i is called a *level* of P. We say that P is effectively locally determined if one can effectively find n_i from i. We then show that any recursive FC-normal effectively locally determined logic program always has a recursive stable model. Another way in which the existence of a recursive stable model can be guaranteed is to introduce a condition which ensures that the lexicographically least stable model of P is recursive. This idea leads to our second key concept which defines what we call *logic programs with effective witnesses.*

The outline of this paper is as follows. In Section 2, we shall establish our notation. In addition we shall define the concept of proof schemes and of FC-normal logic programs which will crucial for our later developments. In Section 3, we shall introduce the new notion of a locally determined logic program. Given a recursive logic program P and an effective listing of the atoms of the Herbrand base, H_P, of P, $H_P = \{a_0, a_1, \ldots\}$, we say that n is a *level* of P if, roughly, whenever there is a proof scheme p for a sentence a_i with $i \leq n$, then there exists a proof scheme q only involving elements from $\{a_0, \ldots, a_n\}$ and such that the restraint set of q is contained in the restraint set of p. We then say that P is locally determined if for every $k \geq 0$, there is an $n_k \geq k$ such that n_k is a level of P. We say that P is effectively locally determined if one can effectively find such an n_k from k. We shall show in Section 3, that if P is an effectively locally determined recursive logic program, then there is a highly recursive tree T such that there is an effective one-to-one correspondence between the stable models of P and the set of infinite paths through T. Thus being effectively locally determined is another condition much like the rps property which reduces the complexity of the set of stable models of P. In Section 4, we shall introduce several strengthenings of local determinedness which will ensure that a recursive logic program always has a recursive stable model. In Section 5, we shall show how one can define polynomial time versions of effectively locally determined recursive logic programs and recursive logic programs with effective witnesses which will ensure that a logic program P will have model which are NP, EXPTIME, etc. In Section 6, we characterize the set of stable models of a locally determined logic program.

The authors would like to thank Victor Marek for helpful comments on this paper.

2. Logic programs, proof schemes, and normality

In this section, we shall introduce several key notions which will be referred to in later section. In particular, we shall carefully define the notion of recursive logic programs. Then we shall define the notion of proof schemes which will lead to the definitions of locally finite programs and locally finite programs with recursive proof structures. Finally we shall describe the extension of the Reiter's concept of normal default theories to recursive logic programs following [20].

2.1. Basic Definitions

A *program clause* is an expression of the form

$$C = p \leftarrow q_1, \ldots, q_n, \neg r_1, \ldots, \neg r_m \tag{1}$$

where $p, q_1, \ldots, q_n, r_1, \ldots, r_m$ are atomic formulas possibly with variables in some first order language $\mathcal{L}$. A *program* is a set of clauses of the form (1). A clause C is called a *Horn clause* if $m = 0$. We let $Horn(P)$ denote the set of all Horn clauses of P. H_P is the Herbrand base of P, that is, the set of all ground atomic formulas of the language of P. We let $ground(P)$ denote the logic program that consists of the set of ground Herbrand substitutions of clauses in P. For the rest of this paper, we shall identify P with $ground(P)$. If

$$C = p \leftarrow q_1, \ldots, q_n, \neg r_1, \ldots, \neg r_m \tag{2}$$

is a clause of $ground(P)$, we let $h(C) = p$ denote the head of p and we call the set $\{r_1, \ldots, r_m\}$ the set of constraints of C and denote it by $cons(C)$.

If P is a program and $M \subseteq H_P$ is a subset of the Herbrand base, define the operator $T_{P,M} \colon \mathcal{P}(H_P) \to \mathcal{P}(H_P)$ where $T_{P,M}(I)$ is the set of all p such that there exists a clause $C = p \leftarrow q_1, \ldots, q_n, \neg r_1, \ldots, \neg r_m$ in P such that $q_1 \in I, \ldots, q_n \in I$ and $\{r_1, \ldots, r_m\} \cap M = \emptyset$.

With fixed M, the operator $T_{P,M}$ is a monotonic finitizable operator, see [2], and hence possesses a least fixpoint $F_{P,M}$. Given program P and $M \subseteq H_P$, the *Gelfond-Lifschitz reduct* of P is defined as follows. For every clause C of P, execute the following operation: if some atom a belongs to M and its negation $\neg a$ appears in C, then eliminate C altogether. In the remaining clauses that have not been eliminated by the operation above, eliminate all the negated atoms. The resulting program P_M^{GL} is a Horn propositional program (possibly infinite). The program P_M^{GL} possesses a least Herbrand model. If that least model of P_M^{GL} coincides with M, then M is called a *stable model* for P. The set of stable models of P will be denoted by $\mathcal{S}(P)$. Gelfond and Lifschitz [11] proved that every stable model of P is a minimal model of P and that M is stable model of P if and only if $M = F_{P,M}$.

The following was also proved in [11].

Proposition 2.1. Let P be a logic program. The set of stable models of P form an antichain. That is, if M_1 and M_2 are stable models of P such that $M_1 \subseteq M_2$, then $M_1 = M_2$.

Having characterized stable models as fixpoints of (parametrized) operators, consider the form of elements belonging to $F_{P,M}$. A P, M-*derivation* of an atom p is a sequence $\langle p_1, \ldots, p_s \rangle$ such that (i) $p_s = p$ and (ii) for every $i \leq s$, either there is a clause "$C = p_i \leftarrow \neg r_1, \ldots, \neg r_m$" where $\{r_1, \ldots, r_m\} \cap M = \emptyset$ is a member of P or there is a clause $D =$ "$p_i \leftarrow q_1, \ldots, q_n, \neg r_1, \ldots, \neg r_m$" such that $D \in P$, $q_1, \ldots, q_n \in \{p_1, \ldots, p_{i-1}\}$ and $\{r_1, \ldots, r_m\} \cap M = \emptyset$. It is easy to show that $F_{P,M}$ is the set of all atoms possessing a P, M-derivation. Thus M is a stable model of the program P if and only if M consists exactly of those atoms which possess a P, M-derivation.

The property that a sequence $\langle p_1, \ldots, p_s \rangle$ is a P, M-derivation of an atom p does not depend on the whole set M but only on the behavior of M on a finite set of atoms. In order that the sequence $\langle p_1, \ldots, p_s \rangle$ be a P, M-derivation of an atom p_s, some atoms must be left out of the set M. Each derivation depends on a finite number of such omitted atoms. In other words, if we classify the atoms according to whether they are "in" or "out" of M, the property that a sequence $\langle p_1, \ldots, p_s \rangle$ is a P, M-derivation depends only on whether a finite number of elements are out of M. The notion of a proof scheme formalizes this idea.

A *(P-)proof scheme* for an atom p is a sequence $\psi = \langle \langle p_i, C_i, U_i \rangle \rangle_{i=1}^s$ of triples where for all i, $p_i \in H_P$, $C_i \in P$ is a clause with the head p_i and U_i is a finite subset of H_P such that (1) $p_s = p$ and
(2) for every i, $C_i = p_i \leftarrow q_1, \ldots, q_n, \neg r_1, \ldots, \neg r_m$, where $\{q_1, \ldots, q_n\} \subseteq \{p_1, \ldots p_{i-1}\}$ and $U_i = U_{i-1} \cup \{r_1, \ldots, r_m\}$. Here by definition, $U_{-1} = \emptyset$. We call p the *conclusion* of ψ, written $p = cln(\psi)$, and the set U_m the

support of ψ, written *supp*(ψ). We say that a subset $M \subseteq H_P$ *admits* a proof scheme $\psi = \langle\langle p_i, C_i, U_i\rangle\rangle_{i=1}^{s}$ if $M \cap U_s = \emptyset$.

The following proposition due to Marek, Nerode,and Remmel in [18] characterizes of stability in terms of the existence of proof schemes.

Proposition 2.2. Let $M \subseteq H_P$. Then M is a stable model of P if and only if
(1) for every $p \in M$, there is a proof scheme ψ for p such that M admits ψ, and
(2) for every $p \notin M$, there is no proof scheme ψ for p such that M admits ψ.

As stated in the introduction, restrictions on the number of proof schemes greatly reduce the possible complexity of the set of stable models of a recursive logic program P. But how many derivation schemes for an atom p can there be? If we allow P to be infinite, then it is easy to construct an example with infinitely many derivations of a single atom. Moreover, given two proof schemes, one can insert one into the other (increasing appropriately the sets U_i in this process, with obvious restrictions). Thus various clauses C_i may be immaterial to the purpose of deriving p. This leads us to introduce a natural relation $\prec$ on proof schemes using a well-known device from proof theory. Namely, we define $S_1 \prec S_2$ if S_1, S_2 have the same conclusion and if every clause appearing in S_1 also appears in S_2. Then a *minimal* proof scheme for p is defined to be a proof scheme S for p such that whenever S' is a proof scheme for p and $S' \prec S$, then $S \prec S'$. Note that $\prec$ is reflexive and transitive, but $\prec$ is not antisymmetric. However it is wellfounded. That is, given any proof scheme S , there is an S' such that $S' \prec S$ and for every S'', if $S'' \prec S'$ then $S' \prec S''$. Moreover, the associated equivalence relation, $S \equiv S'$, defined by $S \prec S'$ and $S' \prec S$, has finite equivalence classes.

Example 2.3. Let P_1 be the following program:
C_1: $p(0) \leftarrow \neg q(Y)$.
C_2: $nat(0) \leftarrow$.
C_3: $nat(s(X)) \leftarrow nat(X)$.
Then atom $p(0)$ possesses infinitely many minimal proof schemes. For instance, each one-element sequence

$$\psi_i = \langle\langle p(0), C_1\Theta_i, \{q(s^i(0))\}\rangle\rangle$$

where Θ_i is the operation of substituting $s^i(0)$ for Y, is a minimal proof scheme for $p(0)$.

Example 2.4. Let P_2 be the following program:
C_1: $q(s(Y)) \leftarrow \neg q(Y)$.

C_2: $nat(0) \leftarrow .$
C_3: $nat(s(X)) \leftarrow nat(X)$.
It is easy to check that for P_2, each atom possesses only finitely many minimal proof schemes.

We shall call a program P *locally finite* if for every atom p, there are only finitely many different supports of *minimal* proof schemes with conclusion p. If P is locally finite and $p \in H_P$, we let D_p denote the set of all supports of minimal proof schemes of p. Clearly for any $M \subseteq H_P$, the question of whether p has a P, M-derivation depends only on $M \cap S$ for $S \in D_p$. This implies that if P is locally finite, when we attempt to construct a subset $M \subseteq H_P$ which is a stable model for P, we can apply a straightforward (although still infinite) tree construction to produce such an M, if such an M exists at all.

2.2. Recursive Logic Programs

Next, we need to make the notion of a recursive program precise. First, assume that we have a Gödel numbering of the elements of the Herbrand base H_P. Thus, we can think of each element of the Herbrand base as a natural number. If $p \in H_P$, write $c(p)$ for the code or Gödel number of p. Let $\omega = \{0, 1, 2, \ldots\}$. Assume $[,]$ is a fixed recursive pairing function which maps $\omega \times \omega$ onto ω and has recursive projection functions π_1 and π_2, defined by $\pi_i([x_1, x_2]) = x_i$ for all x_1 and x_2 and $i \in \{0, 1\}$. Code a finite sequence $\langle x_1, \ldots, x_n \rangle$ for $n \geq 3$ by the usual inductive definition $[x_1, \ldots, x_n] = [x_1, [x_2, \ldots, x_n]]$. Next, code finite subsets of ω via "canonical indices". The *canonical index* of the empty set, $\emptyset$, is the number 0 and the canonical index of a nonempty set $\{x_0, \ldots, x_n\}$, where $x_0 < \ldots < x_n$, is $\sum_{j=0}^{n} 2^{x_j}$. Let F_k denote the finite set whose canonical index is k. Once finite sets and sequences of natural numbers have been coded, we can code more complex objects such as clauses, proof schemes, etc. as follows. Let the code $c(C)$ of a clause $C = p \leftarrow q_1, \ldots, q_n, \neg r_1, \ldots, \neg r_m$ be $[c(p), k, l]$, where k is the canonical index of the finite set $\{c(q_1), \ldots, c(q_n)\}$, and l is the canonical index of the finite set $\{c(r_1), \ldots, c(r_m)\}$. Similarly, let the code $c(\psi)$ of a proof scheme

$$\psi = \langle \langle p_i, C_i, U_i \rangle \rangle_{i=1}^{s}$$

be $[s, [[c(p_1), c(C_1), c(U_1)], \ldots, [c(p_s), c(C_s), c(U_s)]]]$, where for each i, $c(U_i)$ is the canonical index of the finite set of codes of the elements of U_i. The first coordinate of the code of a proof scheme is the length of the proof scheme. Once we have defined the codes of proof schemes, then for locally finite programs, we can define the code of the set $c(D_p)$ consisting of codes of the supports of all minimal proof schemes for

P. Finally we need codes for recursive sets of natural numbers. Let $\phi_0, \phi_1, \ldots$ be an effective list of all partial recursive functions where ϕ_e is the partial recursive function computed by the e-th Turing machine. Similarly, if $A \subseteq \omega$, we let $\phi_0^A, \phi_1^A, \ldots$ be an effective list of all A-partial recursive functions where ϕ_e^A is the partial recursive function computed by the e-th oracle Turing machine with oracle A. By definition, a (recursive) index of a recursive set R is an e such that ϕ_e is the characteristic function of R. Call a program P *recursive* if the set of codes of the Herbrand universe H_P is recursive and the set of codes of the clauses of the program P is recursive. If P is a recursive program, then by an index of P we mean the code of a pair $[u, p]$ where u is an index of the recursive set of all codes of elements in H_P and p is an index of the recursive set of the codes of all clauses in P.

For the rest of this paper we shall identify an object with its code as described above. This means that we shall think of the Herbrand universe of a program, and the program itself, as subsets of ω and clauses, proof schemes, etc. as elements of ω.

We also need to define various types of recursive trees and Π^0_1 classes. Let $\omega^{<\omega}$ be the set of all finite sequences from ω and let $2^{<\omega}$ be the set of all finite sequences of 0's and 1's. Given $\alpha = \langle \alpha_1, \ldots, \alpha_n \rangle$ and $\beta = \langle \beta_1, \ldots, \beta_k \rangle$ in $\omega^{<\omega}$, write $\alpha \sqsubseteq \beta$ if α is initial segment of β, i.e. , if $n \leq k$ and $\alpha_i = \beta_i$ for $i \leq n$. In this paper, we identify each finite sequence $\alpha = \langle \alpha_1, \ldots, \alpha_n \rangle$ with its code $c(\alpha) = [n, [\alpha_1, \ldots, \alpha_n]]$ in ω. Let 0 be the code of the empty sequence $\emptyset$. When we say that a set $S \subseteq \omega^{<\omega}$ is recursive, recursively enumerable, etc., what we mean is that the set $\{c(\alpha) \colon \alpha \in S\}$ is recursive, recursively enumerable, etc. We say that a nonempty subset T of $\omega^{<\omega}$ is a *tree* if $\emptyset \in T$ and T is closed under initial segments. Call a function $f \colon \omega \to \omega$ an infinite *path* through T provided that for all n, $\langle f(0), \ldots, f(n) \rangle \in T$. Let $[T]$ be the set of all infinite paths through T. Call a set A of functions a Π^0_1-class if there exists a recursive predicate R such that $A = \{f \colon \omega \to \omega : \forall n (R(n, [f(0), \ldots, f(n)])\}$. Say that a tree $T \subseteq \omega^{<\omega}$ is *highly recursive* if T is a recursive finitely branching tree and also there is a recursive procedure which, applied to $\alpha = \langle \alpha_1, \ldots, \alpha_n \rangle$ in T, produces a canonical index of the set of immediate successors of α in T. Call a Π^0_1-class A *recursively bounded* if there exists a recursive function $g \colon \omega \to \omega$ such that $(\forall f \in A) \forall n (f(n) \leq g(n))$. It is not difficult to see that if A is a Π^0_1-class, then $A = [T]$ for some recursive tree $T \subseteq \omega^{<\omega}$. Then if A is a recursively bounded Π^0_1-class, it is easy to show that $A = [T]$ for some highly recursive tree $T \subseteq \omega^{<\omega}$, see [7]. We say that a tree T is *decidable* if T is a recursive tree and the set of nodes $\eta \in T$ such that η is an initial segment of some path $\pi \in [T]$ is recursive. A Π^0_1 class C is *decidable* if $C = [T]$ for some decidable tree. For any set

$A \subseteq \omega$, let $A' = \{e\colon \{e\}^A(e) \text{ is defined}\}$ be the jump of A. Let $\mathbf{0}'$ denote the jump of the empty set $\emptyset$. We write $A \leq_T B$ if A is Turing reducible to B and $A \equiv_T B$ if $A \leq_T B$ and $B \leq_T A$.

Formally, we say that there is an effective, one-to-one degree preserving correspondence between the set of stable models $\mathcal{S}(P)$ of a recursive logic program P and the set of infinite paths $[T]$ through a recursive tree T if there are indices e_1 and e_2 of oracle Turing machines such that
(i) $\forall f \in [T](\{e_1\}^{gr(f)} = E_f \in \mathcal{S}(P))$,
(ii) $\forall S \in \mathcal{S}(P)(\{e_2\}^S = f_S \in [T])$, and
(iii) $\forall f \in [T] \forall S \in \mathcal{S}(P)(\{e_1\}^{gr(f)} = S \text{ if and only if } \{e_2\}^S = f)$.
where $\{e\}^B$ denotes the function computed by the e^{th} oracle machine with oracle B. Also, write $\{e\}^B = A$ for a set A if $\{e\}^B$ is a characteristic function of A, and for a function $f\colon \omega \to \omega$, $gr(f) = \{[x, f(x)]\colon x \in \omega\}$. Condition (i) says that the branches of the tree T uniformly produce stable models via an algorithm with index e_1. Condition (ii) says that stable models of $\mathcal{S}$ uniformly produce branches of the tree T via an algorithm with index e_2. Condition (iii) asserts that if $\{e_1\}^{gr(f)} = E_f$, then f is Turing equivalent to E_f. In the sequel we shall not explicitly construct the indices e_1 and e_2, but it will be clear that such indices can be constructed in each case.

Now suppose that P is a locally finite program such that $H_P \subseteq \omega$. There is no guarantee that the global behavior of the function $p \mapsto D_p$, mapping ω into ω, has any sort of effective properties. Thus we are led to define the following.

Definition 2.5. We say that a locally finite recursive program P possesses a **recursive proof structure** (rps) if (1) P is locally finite, and (2) the function $p \mapsto c(D_p)$ is recursive. A locally finite recursive program with an rps is called an *rps program.*

If P is a finite predicate logic program, then we say that P is a locally finite (rps, etc.) if $ground(P)$ is a locally finite (rps, etc.) logic program.

This given, we can now state some basic results from [18, 17, 20] on the complexity of the stable models of recursive logic programs.

Theorem 2.6. For any rps logic program P, there is a highly recursive tree T_P such that there is an effective one-to-one degree preserving correspondence between $[T_P]$ and $\mathcal{S}(P)$. Vice versa, for any highly recursive tree T, there is a rps logic program P_T such that there is an effective one-to-one degree preserving correspondence between $[T]$ and $\mathcal{S}(P_T)$.

Theorem 2.7. For any locally finite recursive logic program P, there is a tree T_P which is highly recursive in $0'$ such that there is an effective one-to-one degree preserving correspondence between $[T_P]$ and $\mathcal{S}(P)$. Vice versa, for any highly recursive tree T in $0'$, there is a locally finite recursive logic program P_T such that there is an effective one-to-one degree preserving correspondence between $[T]$ and $\mathcal{S}(P_T)$.

Theorem 2.8. For any recursive logic program P, there is a recursive tree T_P such that there is an effective one-to-one degree preserving correspondence between $[T_P]$ and $\mathcal{S}(P)$. Vice versa, for any recursive tree T, there is a recursive logic P such that there is an effective one-to-one degree preserving correspondence between $[T]$ and $\mathcal{S}(P_T)$.

We note that in [18], Marek, Nerode, and Remmel show that in Theorem 2.6, 2.7, 2.8, one can replace recursive logic program by finite predicate logic program.

Because the set of degrees of paths through trees have been extensively studied in the literature, we can immediately derive a number of corollaries about the degrees of stable models of recursive logic program. We shall give a few of these corollaries below. We begin with some consequences of Theorem 2.6. First there are some basic results which guarantee that there are stable models of a rps logic program which are not too complex. Call a set $A \subseteq \omega$ *low* if $A' \equiv_T 0'$. This means that A is called low provided that the jump of A is as small as possible with respect to Turing degrees. The following corollary is an immediate consequence of Theorem 4.1 and the work of Jockusch and Soare [13].

Corollary 2.9. Let P be a rps logic program such that $\mathcal{S}(P) \neq \emptyset$. Then
(i) There exists a stable model S of $\mathcal{S}(P)$ such that P is low.
(ii) If P has only finitely many stable models, then every stable S of $\mathcal{S}(P)$ is recursive.

In the other direction, there are a number of corollaries of the Theorem 2.6 which allow us to show that there are rps programs P such that the set of degrees realized by elements of $\mathcal{S}(P)$ are quite complex. Again all these corollaries follow by transferring results of Jockusch and Soare [13, 12].

Corollary 2.10.
1. There is a rps program P such that P has $2^{\aleph_0}$ stable models but no recursive stable model.
2. There is a rps program P such that P has $2^{\aleph_0}$ stable models and any two stable models $S_1 \neq S_2$ of P are Turing incomparable.

3. If $\mathbf{a}$ is any Turing degree such that $\mathbf{0} <_T \mathbf{a} \leq_T \mathbf{0}'$, then there is a rps program P such that P has $2^{\aleph_0}$ stable models but no recursive stable models and P has an stable model of degree $\mathbf{a}$.

4. If $\mathbf{a}$ is any Turing degree such that $\mathbf{0} <_T \mathbf{a} \leq_T \mathbf{0}'$, then there is a rps program P such that P has $\aleph_0$ stable models, P has an stable model S of degree $\mathbf{a}$ and if $S' \neq S$ is an stable model of P, then S' is recursive.

5. There is a rps program P such that P has $2^{\aleph_0}$ stable models and if $\mathbf{a}$ is the degree of any stable model S of P and $\mathbf{b}$ is any recursively enumerable degree such that $\mathbf{a} <_T \mathbf{b}$, then $\mathbf{b} \equiv_T \mathbf{0}'$.

6. If $\mathbf{a}$ is any recursively enumerable Turing degree, then there is a rps program P such that P has $2^{\aleph_0}$ stable models and the set of recursively enumerable degrees $\mathbf{b}$ which contain an stable model of P is precisely the set of all recursively enumerable degrees $\mathbf{b} \geq_T \mathbf{a}$.

The situation for locally finite recursive logic programs P is very similar to the situation for rps logic program except that the degrees of stable models may be more complex. Essentially every result in Corollaries 4.4 and 4.5 hold for locally finite logic programs where each statement is taken relative to an $\mathbf{0}'$ oracle. See [18] for further details. However, the situation for recursive logic programs is quite different. That is, for recursive logic programs the set $\mathcal{S}(P)$ of stable models of P may be extremely complex. For example, all we can say in the positive direction is the following.

Corollary 2.11. 1. Every recursive logic program P which has a stable model has a stable model S such that $S \leq_T B$ where B is a complete Π^1_1-set.

2. If P is a recursive logic program with a unique stable model S, then S is hyperarithmetic.

In the opposite direction we have the following results, see [18].

Corollary 2.12. 1. There a recursive logic program P such that P has a stable model but P has no stable model which is hyperarithmetic.

2. For each recursive ordinal α, there exists a recursive logic program P possessing a unique stable model S such that $S \equiv_T \mathbf{0}^{(\alpha)}$.

2.3. FC-normal logic programs

We end this section with the notion of FC-normal logic programs as defined by Marek, Nerode, and Remmel [20].

Recall that $Horn(P)$ is the set of Horn clauses of a logic program P. We let $T_{Horn(P)}$ denote the one-step provability operator associated with $Horn(P)$, see [2]. That is, if $Q \subseteq H_P$, then $T_{Horn(P)}(Q)$ equals

$$\{p \in H_P : (\exists C = p \leftarrow q_1, \dots q_m \in Horn(P))\ (q_1, \dots, q_m \in Q)\}.$$

Call a family of subsets of H_P, Con, a **consistency property** over P if it satisfies the following conditions:

1. $\emptyset \in Con$.
2. If $A \subseteq B$ and $B \in Con$, then $A \in Con$.
3. Con is closed under directed unions.
4. If $A \in Con$ then $A \cup T_{Horn(P)}(A) \in Con$.

Conditions (1)-(3) are Scott's conditions for information systems. Condition (4) connects "consistent" sets of atoms to the Horn part of the program; if A is consistent then adding atoms provable from A preserves "consistency". The following fact is easy to prove.

Proposition 2.13. If Con is a consistency property with respect to P and $A \in Con$, then $T_{Horn(P)} \Uparrow \omega(A) \in Con$.

Here, for a Horn program Q, $T_Q \Uparrow \omega(A)$ is the cumulative fixpoint of T_Q over A. Proposition 2.13 says that our condition (4) in the definition of consistency property implies that the cumulative closure of a "consistent" set of atoms under $T_{H(P)}$ is still "consistent".

Given a consistency property, we define the concept of an **FC-normal** program with respect to that property. Here FC stands for "Forward Chaining".

Definition 2.14. (a) Let P be a program, let Con be a consistency property with respect to P. Call P **FC-normal with respect to** Con if for every clause $C = p \leftarrow q_1, \dots, q_n, \neg r_1, \dots, \neg r_m$ such that $C \in ground(P) - ground(Horn(P))$ and every consistent fixpoint A of $T_{Horn(P)}$ such that $q_1, \dots, q_n \in A$ and $p, r_1, \dots, r_m \notin A$ we have
(1) $A \cup \{p\} \in Con$ and
(2) $A \cup \{p, r_i\} \notin Con$ for all $1 \leq i \leq m$.

(b) P is called **FC-normal** if there exists a consistency property Con such that P is FC-normal with respect to Con.

Example 2.15. Let the Herbrand base consist of atoms a, b, c, d, e, f. Let the consistency property be defined by the following condition: $A \notin Con$ if and only if either $\{c, d\} \subseteq A$ or $\{e, f\} \subseteq A$.

Now consider the following program.

1) $a \leftarrow$
2) $b \leftarrow c$
3) $c \leftarrow b$
4) $c \leftarrow a, \neg d$
5) $e \leftarrow c, \neg f$

It is not difficult to check that this program is FC-normal with respect to the consistency property described above. Moreover, one can easily check that P possesses a unique stable model $M = \{a, b, c, e\}$.
If we add to this program the clause $f \leftarrow c, \neg e$, the resulting program is still FC-normal but now there are two stable models, $M_1 = \{a, b, c, e\}$ and $M_2 = \{a, b, c, f\}$.

Marek, Nerode, and Remmel [20] showed that FC-normal normal programs have many of the properties that are possessed by normal default theories.

Theorem 2.16. If P is an FC-normal program, then P possesses a stable model.

Theorem 2.17. If P is an FC-normal program with respect to the consistency property Con and $I \in Con$, then P possesses a stable model I' such that $I \subseteq I'$.

Marek, Nerode, and Remmel proved Theorem 2.16 and 2.17 via a generally forward chaining algorithm which can be applied to FC-normal programs of any cardinality. Since in our case, we are dealing with only recursive and hence countable programs, we shall give only the countable version of their forward chaining construction. That is, suppose we fix some well-ordering $\prec$ of $ground(P) - ground(H(P))$ of order type ω. Thus, the well-ordering $\prec$ determines some listing of the clauses of $ground(P) - ground(H(P))$, $\{c_n : n \in \omega\}$. Their forward chaining construction then defines an increasing sequence of sets $\{T_n^{\prec}\}_{n \in \omega}$ in stages.

The countable forward chaining construction of $T^{\prec} = \bigcup_{n \in \omega} T_n^{\prec}$.

Stage 0. Let $T_0^{\prec} = T_{Horn(P)} \Uparrow \omega(\emptyset)$.

Stage $n+1$. Let $\ell(n+1)$ be the least $s \in \omega$ such that
$c_s = \varphi \leftarrow \alpha_1, \ldots, \alpha_k, \neg\beta_1, \ldots, \neg\beta_m$ where $\alpha_1, \ldots, \alpha_k \in T_n^{\prec}$ and

$\beta_1, \ldots, \beta_m, \varphi \notin T_n^{\prec}$. If there is no such $\ell(n+1)$, let $T_{n+1}^{\prec} = T_n^{\prec}$. Otherwise let

$$T_{n+1}^{\prec} = T_{Horn(P)} \Uparrow \omega(T_n^{\prec} \cup \{p_{\ell(n+1)}\})$$

where $p_{\ell(n+1)}$ is the head of $c_{\ell(n+1)}$.

Example 2.18. If we consider the final extended program of Example 2.15, it is easy to check that any ordering $\prec_1$ in which the clause $C_1 = e \leftarrow c, \neg f$ precedes the clause $C_2 = f \leftarrow c, \neg e$ will have $T^{\prec_1} = M_1$ while any ordering $\prec_2$ in which C_2 precedes C_1 will have $T^{\prec_2} = M_2$.

This given, Marek, Nerode, and Remmel proved the following results.

Theorem 2.19. If P is a countable FC-normal program and $\prec$ is *any* well-ordering of $ground(P) - ground(H(P))$ of order type ω, then :
(1) $T^{\prec}$ is a stable model of P where $T^{\prec}$ is constructed via the countable forward chaining algorithm.
(2) (completeness of the construction). Every stable model model of P is of the form $T^{\prec}$ for a suitably chosen ordering $\prec$ of $ground(P) - ground(H(P))$ of order type ω where $T^{\prec}$ is constructed via the countable forward chaining algorithm.

Theorem 2.20. If P is an FC-normal logic program with respect to Con, then every stable model M of P is in Con.

Theorem 2.21. Let P be an FC-normal logic program with respect to a consistency property Con. Then if E_1 and E_2 are two distinct stable models of P, then $E_1 \cup E_2 \notin Con$.

Given a logic program P and a stable model M, we let $NG(M, P)$, the set of non-Horn generating clauses of P be equal to the set of all clauses $c = \varphi \leftarrow \alpha_1, \ldots, \alpha_k, \neg\beta_1, \ldots, \neg\beta_m$ in $ground(P)$ such that $\alpha_1, \ldots, \alpha_k \in M$ and $\beta_1, \ldots, \beta_m \notin M$.

FC-normal programs possess the following key "semi-monotonicity" property.

Theorem 2.22. Let P_1, P_2 be two programs such that $P_1 \subseteq P_2$ but $H(P_1) = H(P_2)$. Assume, in addition, that both are FC-normal with respect to the same consistency property. Then for every stable model M_1 of P_1, there is a stable model M_2 of P_2 such that (1)$M_1 \subseteq M_2$ and (2) $NG(M_1, P_1) \subseteq NG(M_2, P_2)$.

As mentioned in the introduction, a recursive FC-normal logic program P is guaranteed to have at least one relatively well behaved stable model which is in great contrast to Corollary 2.12.

Theorem 2.23. Suppose that P is a recursive logic program and P is FC-normal. Then P has a stable model S such that S is r.e. in $\mathbf{0}'$ and hence $E \leq_T \mathbf{0}''$.

We note that Theorem 2.23 is in some sense the best possible. That is, results from [20] show that the following holds. Given sets $A, B \subseteq \omega$, let $A \oplus B = \{2x : x \in A\} \cup \{2x+1 : x \in B\}$.

Theorem 2.24. Let A be any r.e. set and B be any set which is r.e. in A, i.e. $B = \{x : \phi_e^A(x) \downarrow\}$. Then there is a recursive FC-normal logic program P such P has a unique stable model S and $S \equiv_T A \oplus B$. In particular, if B is any set which is r.e. in $\mathbf{0}'$ and $B \geq_T \mathbf{0}'$, then there is an FC-normal recursive logic program P such that P has a unique stable model S and $S \equiv_T B$.

However if either $Horn(P)$ or $P - Horn(P)$ is finite, then one can improve on Theorem 2.23. That is, the following was proved in [20].

Theorem 2.25. Let P be a FC-normal recursive logic program such that $P - Horn(P)$ is finite, then every stable model of $\mathcal{S}$ is r.e..

We say that a recursive logic program P is **monotonically decidable** if for any finite set $F \subseteq \mathcal{H}(P)$, $T_{Horn(P)} \Uparrow \omega(F)$ is recursive and there is a uniform effective procedure to go from a canonical index of a finite set F to a recursive index of the $T_{Horn(P)} \Uparrow \omega(F)$, i.e. if there is a recursive function f such that for all k, $\phi_{f(k)}$ is the characteristic function of $T_{Horn(P)} \Uparrow \omega(D_k)$. It is easy to see that if $Horn(P)$ is finite, then the recursive program P is automatically monotonically decidable.

Theorem 2.26. Let P be a recursive logic program such that P is FC-normal and monotonically decidable, then P has an stable model which is r.e.

We end this section by giving complexity results for finite FC-normal logic program where the forward chaining algorithm runs in polynomial time.

For complexity considerations, we shall assume that the elements of H_P are coded by strings over some finite alphabet Σ. Thus every $a \in H_P$ will have some length which we denote by $||a||$. Next, for a clause

$$r = c \leftarrow a_1, \ldots, a_n, \neg b_1, \ldots, \neg b_m,$$

we define $||r|| = (\sum_{i \leq n} ||a_i||) + (\sum_{i \leq m} ||b_j||) + ||c||$. Finally, for a set Q of clauses, we define

$$||Q|| = \sum_{r \in Q} ||r||.$$

Theorem 2.27. Suppose P is a finite FC-normal logic program and $\prec$ is some well-ordering of $P - Horn(P)$. Then $E^{\prec}$ as constructed via our forward chaining algorithm can be computed in time $O(||Horn(P)|| \cdot ||P - Horn(P)|| + ||P - Horn(P)||^2)$.

We note that none of the theorems above make any explicit assumptions that the underlying consistency property of a recursive FC-normal logic P is in any way effective. Indeed none of the above results require that the underlying consistency property has any effective properties.

Finally Marek, Nerode, and Remmel [20] proved the following result about recursive FC-normal logic programs.

Theorem 2.28. Let T be a recursive tree in $2^{<\omega}$ such that $[T] \neq \emptyset$. Then there is a FC-normal recursive logic program P such that there is an effective one-to-one degree preserving correspondence between $[T]$ and $\mathcal{S}(P)$.

Reiter ([22]) proved that there is a recursive normal default theory with no recursive extension. Theorem 2.28, which was originally proved for nonmonotonic rule systems of which logic programs and default theories are special cases, contains Reiter's result as special case. In addition, it gives much finer information even for recursive normal default theories since the set of degrees of paths through highly recursive trees have been extensively studied. For example, our correspondence allows us to transfer all results about the possible degrees of paths through highly recursive trees to results about the degrees of stable models of recursive FC-normal logic programs. That is, all the results of Corollary 2.10 continue to hold if we replace recursive logic program by FC-normal recursive logic program in each part of its statement.

3. Locally Determined Logic Programs

In this section, we shall introduce the key notion of a locally determined logic program P. For the rest of this paper, we shall only consider countable logic programs P. Thus whenever we say that P is a logic program, we shall always assume that P is countable.

The informal notion of a locally determined logic program P is one in which the existence of a proof scheme for an atom a_i (or the lack of existence thereof) can be determined by examining only clauses or proof schemes involving some initial segment of the Herbrand base of P. More formally, fix some countable logic program P and some listing $a_0, a_1, \ldots$ of the atoms of Herbrand base of P without repetitions. (We shall make the convention that if P is a recursive logic program, then there is some recursive function $h : \omega \to \omega$ such that $h(i) = a_i$.) Then given a proof scheme or a clause ψ, we write $max(\psi)$ for the $max(\{i : a_i \text{ occurs in } \psi\})$. We shall write P_n for the set of all clauses $C \in P$ such that $max(C) \leq n$ and let $A_n = \{a_0, \ldots, a_n\}$.

Definition 3.1. We shall say that n is a *level* of P if for all $S \subseteq \{a_0, \ldots, a_n\}$ and all $i \leq n$, whenever there exists a proof scheme ϕ such that $cln(\phi) = a_i$ and $supp(\phi) \cap S = \emptyset$, then there exists a proof scheme ψ such that $cln(\psi) = a_i$, $supp(\psi) \cap S = \emptyset$ and $max(\psi) \leq n$. Note that by definition, the Herbrand base H_{P_n} of P_n is contained in A_n. We let $lev(P) = \{n : n \text{ is a level of } P\}$.

The following result has essentially been proven in [9, 14]

Theorem 3.2. Suppose that n is a level of P and E is a stable model of P. Then $E_n = E \cap \{a_0, \ldots, a_n\}$ is a stable model of P_n.

Proof. If E is a stable model of P, then for any $a_i \in E_n$, there is a proof scheme ψ such that $cln(\psi) = a_i$ and $supp(\psi) \cap E = \emptyset$. Thus, in particular, $supp(\psi) \cap E_n = \emptyset$ so that since n is a level, there exists a proof scheme ψ' such that $max(\psi') \leq n$, $cln(\psi') = a_i$, and $supp(\psi') \cap E_n = \emptyset$. Thus ψ' is a proof scheme of P_n and E_n admits ψ'. Vice versa, if $i \leq n$ and a_i is not in E_n, then there can be no proof scheme ψ of P_n such that $cln(\psi) = a_i$, $max(\psi) \leq n$, and $supp(\psi) \cap E_n = \emptyset$ since this would violate the fact that E is a stable model of of P. Thus E_n is a stable model of P_n.

Definition 3.3. We shall say that a logic program P is **locally determined** if P is countable and there are infinitely many n such that n is a level of P.

Example 3.4. Let P be the logic program with the following set of clauses.

(1)	$a_{2i} \leftarrow \neg a_{2i+1}$	for all $i \in \omega$
(2)	$a_{2i+1} \leftarrow \neg a_{2i}$	for all $i \in \omega$

Thus the Herbrand base of P is $\{a_0, a_1, \ldots\}$. It is easy to see that S is stable model of P if and only if $|S \cap \{a_{2i}, a_{2i+1}\}| = 1$ for all $i \in \omega$. In fact, one can easily prove that $lev(P) = \{2i + 1 : i \in \omega\}$. Moreover, it is clear that P is also locally finite.

Example 3.5. Let Q be the logic program with the following set of clauses for all $i \in \omega$.

$$\begin{array}{lll}
(1) & a_{3i} & \leftarrow \neg a_{3i+1}, \neg a_{3i-2}, \ldots \neg a_1, \neg a_{3i+2}, \neg a_{3i-1}, \ldots \neg a_2, \\
(2) & a_{3i+1} & \leftarrow \neg a_{3i}, \neg a_{3i-3}, \ldots \neg a_0, \neg a_{3i+2}, \neg a_{3i-1}, \ldots \neg a_2, \\
(3) & a_{3i+2} & \leftarrow \neg a_{3i}, \neg a_{3i-3}, \ldots \neg a_0, \neg a_{3i+1}, \neg a_{3i-2}, \ldots \neg a_1, \\
(4) & a_{3i} & \leftarrow a_{3i+3} \\
(5) & a_{3i+1} & \leftarrow a_{3i+4} \\
(6) & a_{3i+2} & \leftarrow a_{3i+5}
\end{array}$$

The Herbrand base of Q is $\{a_0, a_1, \ldots\}$. It is easy to see that P has exactly 3 stable models, namely, $S_0 = \{a_{3i} : i \in \omega\}$, $S_1 = \{a_{3i+1} : i \in \omega\}$ and $S_2 = \{a_{3i+2} : i \in \omega\}$. In this case, Q is not locally finite since for any $i > 0$, the following set of clauses can be used to construct a minimal proof scheme of a_0 with support equal to

$$\{a_{3i+1}, a_{3i-2}, \ldots, a_1, a_{3i+2}, a_{3i-1}, \ldots, a_2\}.$$

$a_{3i} \leftarrow \neg a_{3i+1}, \neg a_{3i-2}, \ldots \neg a_1, \neg a_{3i+2}, \neg a_{3i-1}, \ldots \neg a_2$

$a_{3i-3} \leftarrow a_{3i}$

$a_{3i-6} \leftarrow a_{3i-3}$

$\vdots$

$a_0 \leftarrow a_3.$

However we claim that $lev(P) = \{3i + 2 : i \in \omega\}$. That is, fix $n \geq 0$ and suppose that $T \subseteq \{a_i : i \leq 3n + 2\}$ and suppose that ψ is a proof scheme with conclusion a_r where $supp(\psi) \cap T = \emptyset$ and $r \leq 3n + 2$. We shall consider 3 cases.

Case 1 $T = \emptyset$.

In this case it is easy to see that every element of $\{a_i : i \leq 3n + 2\}$ which is the conclusion of a proof scheme of P_{3n+2} of length one using one of the clauses (1), (2), and (3).

Case 2. There exist a_{3i+s} and a_{3j+t} in T where $s \neq t$ and $s, t \in \{0, 1, 2\}$. In this case it is easy to see that all clauses of the form (1), (2) or (3) where the head of the clause is some a_k with $k > 3n+2$ cannot be part of a proof scheme ψ such that $supp(\psi) \cap T = \emptyset$. Thus the only clauses of the form (1), (2), or (3) that can be part of ψ are clauses from P_{3n+2}. However, in that case, the only clauses of the form (4), (5), and (6) that can be part of ψ must also be in P_{3n+2} because there is no way that we can derive an element a_k with $k \geq 3n+2$ that is in the body of

a clause of the form (4), (5), and (6) if we can only use clauses in ψ of type (1), (2), and (3) in ψ from P_{3n+2}. Here we are using the fact that ψ is a minimal proof scheme. It follows that ψ must be a proof scheme for P_{3n+2}.
Case 3. Conditions of Case 1 or Case 2 do not hold.
In this case, T must be contained in one of the stable models S_0, S_1, or S_2. We shall assume that $T \subseteq S_0$ since the other two cases are similar. Since $T \cap S_0 \neq \emptyset$, there can be no clause of type (2) and (3) in ψ which in not in P_{3n+2}. Now suppose that a clause of type (3) occurs in ψ with head a_{3j} where $j > n$. Then this clause combined with clauses of type (4) in ψ can be used to derive elements of the form a_{3i} with $i \leq n$. But for all $i \leq n$, we can clearly immediately derive a_{3i} form the clause

$$a_{3i} \leftarrow \neg a_{3i+1}, \neg a_{3i-2}, \ldots, \neg a_1, \neg a_{3i+2}, \neg a_{3i-1}, \ldots \neg a_2 \qquad (3)$$

which lies in P_{3n+2} and whose constraints do not intersect T. It follows that we can construct an minimal proof scheme ψ' in P_{3n+2} with the same conclusion as ψ such that $supp(\psi') \cap T = \emptyset$.

It follows that $3n+2$ is level of P for all n. Moreover it is clear from the clauses of type (1) and (2) that $3n$ and $3n+1$ are not levels of P so that $lev(P) = \{3n+2 : n \in \omega\}$ as claimed.

Example 3.6. In this example, we give a program which is very similar to example 3.5, but is not locally determined. Let R be the logic program with the following set of clauses.

(1) a_{3i}	$\leftarrow \neg a_{3i+1}, \neg a_{3i+2}$	for all $i \in \omega$
(2) a_{3i+1}	$\leftarrow \neg a_{3i}, \neg a_{3i+2}$	for all $i \in \omega$
(3) a_{3i+2}	$\leftarrow \neg a_{3i}, \neg a_{3i+1}$	for all $i \in \omega$
(4) a_{3i}	$\leftarrow a_{3i+3}$	for all $i \in \omega$
(5) a_{3i+1}	$\leftarrow a_{3i+4}$	for all $i \in \omega$
(6) a_{3i+2}	$\leftarrow a_{3i+5}$	for all $i \in \omega$

Just as in example 3.5, the Herbrand base of R is $\{a_0, a_1, \ldots\}$ and it easy see that P has exactly 3 stable models, namely, $S_0 = \{a_{3i} : i \in \omega\}$, $S_1 = \{a_{3i+1} : i \in \omega\}$ and $S_2 = \{a_{3i+2} : i \in \omega\}$. In this case, R has no levels. Again it is easy to see that the clauses of the form (1) and (2) ensure that $3n$ and $3n+1$ are not levels of P. However in this case, $3n+2$ is also not a level of P. That is consider $T = \{a_{3n}, a_{3n+1}\}$. It is easy to see that there is no minimal proof scheme ψ of P_{3n+2} such that $supp(\psi) \cap T = \emptyset$ and the conclusion of ψ is a_{3n+2}. However the following is a minimal proof ψ' of P with conclusion a_{3n+2} such that

$supp(\psi') \cap T = \emptyset$.

$$\begin{array}{l}\langle\ a_{3n+5}, a_{3n+5} \leftarrow \neg a_{3n+3}, \neg a_{3n+4}, \{a_{3n+3}, a_{3n+4}\}\rangle \\ \langle\ a_{3n+2}, a_{3n+2} \leftarrow a_{3n+5}\rangle, \{a_{3n+3}, a_{3n+4}\}\rangle\rangle.\end{array}$$

Example 3.7. Suppose that we are given a set $L = \{l_0 < l_1 < \ldots\}$. Then we can construct a program P such that $H_P = \{a_0, a_1, \ldots\}$ and $lev(P) = L$ as follows. Let $l_{-1} = -1$. Then for each $n \geq 0$, we add the clause

$$a_{l_n} \leftarrow$$

to P if $l_n - l_{n-1} = 1$. Otherwise we add the following clauses to P

$$a_{l_{n-1}+k} \leftarrow \neg a_{l_{n-1}+1}, \ldots, \neg a_{l_{n-1}+k-1}, \neg a_{l_{n-1}+k+1}, \ldots, \neg a_{l_{n-1}+(l_n-l_{n-1})}$$

for all $k = 1, \ldots, l_n - l_{n-1}$.

Definition 3.8. Suppose that P is a recursive logic program. Then we say that P is **effectively locally determined** if P is locally determined and there is a recursive function f such that for all i, $f(i) \geq i$ and $f(i)$ is a level of P.

In [18, 17], Marek, Nerode, and Remmel showed that the problem of finding a stable model of a locally finite recursive logic program can be reduced to finding an infinite path through a finitely branching recursive tree and the problem of finding a stable model of a rps logic program can be reduced to finding an infinite path through a highly recursive tree. A locally determined logic program is not always locally finite since it is possible that a given atom has infinitely many proof schemes which involves arbitrarily large atoms as in Example 3.5 above. Vice versa, it is possible to give examples of locally finite logic programs which is not locally determined. Nevertheless, we shall see that we get similar results to those of Marek, Nerode, and Remmel for locally determined and effectively locally determined recursive logic programs.

Theorem 3.9. Let P be a recursive logic program.

1. If P is locally determined, then there is a recursive finitely branching tree T and a one-to-one degree preserving correspondence between the set of stable models $\mathcal{S}(P)$ of P and $[T]$ and

2. If P is effectively locally determined, then there is a highly recursive finitely branching tree T and a one-to-one degree preserving correspondence between the set set of stable models $\mathcal{S}(P)$ of P and $[T]$.

Proof. There is no loss in generality in assuming that $H_p = \omega$ and that $a_0 = 0, a_1 = 1, \ldots$. Next observe that for each n, P_n has only finitely many minimal proof schemes so that we can effectively list all minimal proof schemes $\psi_0 < \psi_1 < \ldots$ in such a way that

1. if $max(\psi_k) = i$ and $max(\psi_l) = j$ and $i < j$, then $k < l$. (This says that if $i < j$, then the proof schemes whose max is i come before those proof schemes whose max is j.)

2. if $max(\psi_k) = max(\psi_l) = i$, $k < l$ if and only if $c(\psi_k) < c(\psi_l)$ where $c(\psi)$ denotes the index assigned to a proof scheme ψ under our effective Gödel numbering of the proof schemes.

We shall encode a stable model M of P by a path $\pi_M = (\pi_0, \pi_1, \ldots)$ through the complete ω-branching tree $\omega^{<\omega}$ as follows. First, for all $i \geq 0$, $\pi_{2i} = \chi_M(i)$. That is, at the stage $2i$ we encode the information if i belongs to M. Next, if $\pi_{2i} = 0$ then $\pi_{2i+1} = 0$. But if $\pi_{2i} = 1$ so that $i \in M$, then $\pi_{2i+1} = q_M(i)$ where $q_M(i)$ is the least q such $cln(\psi_q) = i$ and $supp(\psi_q) \cap M = \emptyset$. Thus $\psi_{q_M(i)}$ is the least minimal proof scheme which shows that $i \in F_{P,M}$.

Clearly $M \leq_T \pi_M$ since it is enough to look at the values of π_M at the even levels to read off M. Now given an M-oracle, it should be clear that for each $i \in M$, we can use an M-oracle to find $q_M(i)$ effectively. This means that $\pi_M \leq_T M$. Thus the correspondence $M \mapsto \pi_M$ is an effective degree-preserving correspondence. It is trivially one-to-one.

Next we construct a recursive tree $T \subseteq \omega^\omega$ such that

$$[T] = \{\pi_E : E \text{ is a stable model of } P\}. \tag{4}$$

Let $L_k = max(\{i : max(\psi_i) \leq k\})$. It is easy to see that since P is a recursive logic program, we can effectively calculate L_k from k. We have to say which finite sequences belong to our tree T. To this end, given a sequence $\sigma = (\sigma(0), \ldots, \sigma(k)) \in \omega^{<\omega}$ set $I_\sigma = \{i : 2i \leq k \wedge \sigma(2i) = 1\}$ and $O_\sigma = \{i : 2i \leq k \wedge \sigma(2i) = 0\}$. Now we define T by putting σ into T if and only if the following four conditions are met:
(a) $\forall i(2i + 1 \leq k \wedge \sigma(2i) = 0 \Rightarrow \sigma(2i + 1) = 0)$.
(b) $\forall i(2i + 1 \leq k \wedge \sigma(2i) = 1 \Rightarrow \sigma(2i + 1) = q$ where ψ_q is a minimal proof scheme such that $cln(\psi_q) = i$ and $supp(\psi_q) \cap I_\sigma = \emptyset)$.
(c) $\forall i(2i + 1 \leq k \wedge \sigma(2i) = 1 \Rightarrow$ there is no $c \in L_{\lfloor k/2 \rfloor}$ such that $cln(\psi_c) = i$, $supp(\psi_c) \subseteq O_\sigma$ and $c < \sigma(2i + 1))$.
(here $\lfloor \cdot \rfloor$ is the so-called number-theoretic "floor" function).
(d) $\forall i(2i \leq k \wedge \sigma(2i) = 0 \Rightarrow$ there is no $c \in L_{\lfloor k/2 \rfloor}$ such that $cln(\psi_c) = i$ and $supp(\psi_c) \subseteq O_\sigma)$.

It is immediate that if $\sigma \in T$ and $\tau \sqsubseteq \sigma$, then $\tau \in T$. Moreover it is clear from the definition that T is a recursive subset of $\omega^{<\omega}$. Thus T is a recursive tree. Also, it is easy to see that our definitions ensure that, for any stable E of P, the sequence π_E is a branch through T, that is, $\pi_E \in [T]$.

We shall show now that every infinite branch through T is of the form π_E for a suitably chosen stable model E. To this end assume that $\beta = (\beta(0), \beta(1), \ldots)$ is an infinite branch through T. There is only one candidate for E, namely $E_\beta = \{i : \beta(2i) = 1\}$. Two items have to be checked, namely, (I) E_β is a stable model of P and (II) $\pi_{(E_\beta)} = \beta$.

To prove (I), first observe that if $i \in E_\beta$, then $\sigma(2i) = 1$ and $\sigma(2i + 1) = q$ where ψ_q is a proof scheme such that $cln(\psi_q) = i$. Moreover condition (b) and the fact that $\sigma_n = (\beta_0, \beta_1, \ldots, \beta_n) \in T$ for all $n \geq 2i + 1$ easily imply that $supp(\psi_q) \cap I_{\sigma_n} = \emptyset$ for all such n and hence $supp(\psi_q) \cap E_\beta = \emptyset$. In addition, condition (c) ensures that ψ_q is the least proof scheme with this property. Similarly if $i \notin E_\beta$, then condition (d) and the fact that $\sigma_n = (\beta_0, \beta_1, \ldots, \beta_n) \in T$ for all $n \geq 2i + 1$ easily imply that there can be no proof scheme ψ_q with $cln(\psi_q) = i$ and $supp(\psi_q) \cap E_\beta = \emptyset$. It then easily follows from Proposition 2.2 that E_β is a stable model of P and that $\pi_{(E_\beta)} = \beta$

The key fact that we need to establish the branching properties of T is that for any sequence $\sigma \in T$ and any i, either $\sigma(2i) = \sigma(2i + 1) = 0$ or $\sigma(2i) = 1$ and $\sigma(2i+1)$ codes a minimal proof scheme for i. To prove this fact simply observe that when a proof scheme $\psi = \sigma(2i + 1)$ does not correspond to a path π_E, then there will be some k such that σ has no extension in T of length k. This will happen once we either find a smaller code for a proof scheme or we find some $u > i$ in the support of ψ such that all possible extensions τ of σ have $\tau(2u) = 1$.

We claim that T is always finitely branching and that if P is effectively locally determined, then T is highly recursive. Clearly the only case of interest is when $2i+1 \leq k$ and $\sigma(2i) = 1$. In this case we will let $\sigma(2i + 1) = c$ where $cln(\psi_c) = i$ and $supp(\psi_c) \cap I_\sigma = \emptyset$ and there is no $a < c$ such that $cln(\psi_a) = i$ and $supp(\psi_a) \cap I_\sigma = \emptyset$. Now suppose that p is a level and $i < p$. Then by definition, there must be a minimal proof scheme ψ such that $max(\psi) \leq p$, $cln(\psi) = i$, and $supp(\psi) \cap I_\sigma = \emptyset$. Thus $\psi = \psi_q$ for some $q \leq L_p$. It follows that $c \leq L_p$ where p is the least level greater than or equal to i. Thus T is always finitely branching. Now if P is effectively locally determined, which is witnessed by the recursive function f, then it will always be the case that $c \leq L_{f(i)}$ so that T will be highly recursive.

Corollary 3.10. Suppose that P is a countable locally determined logic program such that there are infinitely many n such that P_n has a stable model E_n. Then P has a stable model.

Proof. Consider the tree T constructed for P as in Theorem 3.9. Here we again can construct our sequence of minimal proof schemes $\psi_0, \psi_1, \ldots$ recursive in P just as we did in Theorem 3.9. However, we can only conclude that T is recursive in P. Nevertheless, we are guaranteed that T is finitely branching, which is all we need for our argument.

Now fix some level n and consider some $m \geq n$ such that P_m has a stable model E_m. Then by the exact same argument as in Theorem 3.2, $E_n = E_m \cap \{0, \ldots, n\}$ will be a stable model of P_n. Now consider the node $\sigma_{E_n} = (\sigma(0), \ldots, \sigma(2n+1))$ such that

1. $\sigma(2i) = 0$ if $i \notin E_n$,
2. $\sigma(2i) = 1$ if $i \in E_n$,
3. $\sigma(2i+1) = 0$ if $\sigma(2i) = 0$, and
4. $\sigma(2i+1) = c$ where c is least number such that $max(\psi_c) \leq n$, $cln(\phi_c) = i$, and $supp(\psi_c) \cap E_n = \emptyset$. (Note that it follows from our ordering of minimal proof schemes that ψ_c is the least proof scheme ψ such that $cln(\psi) = i$ and $supp(\psi) \cap E_n = \emptyset$.)

It is easy to check that our construction of T ensures that $\sigma \in T$. It follows that T is infinite finitely branching tree and hence T has infinite path π by König's Lemma. Our proof of Theorem 3.9 shows that E_π is a stable model of P.

One can immediately apply a number of known results from the theory of recursively bounded Π^0_1 classes to derive corresponding results about the set of stable models of an effectively locally determined recursive logic program.

Corollary 3.11. Suppose that P is an effectively locally determined recursive logic program which has at least one stable model. Then

1. P has a stable model whose Turing jump is recursive in $\mathbf{0}'$.
2. If P has no recursive stable model, then P has $2^{\aleph_0}$ stable models.
3. If P has only finitely many stable models, then each of these stable models is recursive.
4. There is a stable model E of P in an r.e. degree.
5. There exist stable models E_1 and E_2 of P such that any function, recursive in both E_1 and E_2, is recursive.
6. If P has no recursive stable model, then there is a nonzero r.e. degree $\mathbf{a}$ such that P has no stable model recursive in $\mathbf{a}$.

A similar corollary holds for locally determined recursive logic programs where the statements in Corollary 3.11 are replaced by versions which are relativized to a $\mathbf{0}'$ oracle.

Our next result will be the converse to Theorem 3.9.

Theorem 3.12. Let T be a highly recursive tree contained in $\omega^{<\omega}$. There exists an effectively locally determined recursive logic program P_T such that there is an effective one-to-one degree preserving correspondence between $[T]$ and $\mathcal{S}(P_T)$.

Proof. Intuitively, the Herbrand base of our program will simply be the set of nodes of T plus an infinite set of distinguished elements. However, we have to supply a coding of strings in T to formally make our program into a recursive program. Thus we will let the code of any string σ, $c(\sigma)$, be defined by letting $c(\sigma) = 2n$ if σ is the n-th string in the effective listing of the nodes of T where we order the strings of T first by length and then by lexicographic order. Since T is highly recursive, it follows that we can effectively find $c(\sigma)$. We will always let our set of distinguished elements be the set of odd numbers $\{2n+1 : n \in \omega\}$. Thus formally, the Herbrand base of P will be $H_{P_T} = \{c(\sigma) : \sigma \in T\} \cup \{2n+1 : n \in \omega\}$. Thus if T is infinite, then $H_{P_T} = \omega$ and if T is finite, then H_{P_T} will consist of all odd numbers plus some initial segment of the even numbers.

This given, for any $k \geq 0$, let $2n_k$ be the largest code of a string of T of length k. It is easy to see that our coding ensures the set of codes of strings of length k are precisely $2(n_{k-1}+1), \ldots, 2n_k$ for all $k \geq 0$. Moreover 0 is the code of the empty string so that $n_0 = 0$.

The idea is to have the stable models of P_T to be the set of all $S_\pi = \{c(\sigma) : \sigma \in \pi\}$ where π is an infinite path through T. Then clearly, there will be an effective one-to-one degree preserving correspondence between $[T]$ and $\mathcal{S}(P_T)$. Our desired program P_T consists of four sets of clauses.

(1) $\quad c(\emptyset) \leftarrow$ (This clause ensures that $c(\emptyset) = 0$ is in all stable models.)

(2) For all σ in T which has at least one immediate successor in T, let $\sigma^\frown b_0^\sigma, \ldots, \sigma^\frown b_{m_\sigma}^\sigma$ be set of all immediate successors of σ in T where $b_0^\sigma < \ldots < b_{m_\sigma}^\sigma$. Then for each such σ, we will have the following set of clauses:

$$c(\sigma^\frown b_j^\sigma) \leftarrow c(\sigma), \neg c(\sigma^\frown b_0^\sigma), \ldots, \neg c(\sigma^\frown b_{j-1}^\sigma), \neg c(\sigma^\frown b_{j+1}^\sigma), \ldots, \neg c(\sigma^\frown b_{m_\sigma}^\sigma)$$

for $j = 0, \dots m_\sigma$.
(This set of clauses is designed to ensure that if $c(\sigma)$ is in a stable model M, then one of $c(\sigma^\frown b_j^\sigma)$ is in M.)

(3) For all $\sigma \in T$ such that σ is a terminal node of T, we add the clause

$$2c(\sigma) - 1 \leftarrow c(\sigma), \neg(2c(\sigma) - 1).$$

(This clause will ensure that $c(\sigma)$ cannot be in any stable model of P_T. Recall that we identify the atoms with natural numbers, so there is an atom $2c(\sigma) - 1$.)

(4) For all $k > 0$ such that T has node of length $k - 1$ but no node of length k, then we have the following clause.

$$2n_k + 1 \leftarrow \neg 2n_k + 1.$$

(This clause is designed to ensure that P_T has no stable model if T is finite.)

First observe that since T is a highly recursive tree, then P_T is a recursive program. Moreover the only clauses that have a conclusion corresponding to a node of length $k > 0$ in T are clauses the form (2), which only involve codes of nodes of length k and $k-1$. It easily follows that $2n_k$ is a level of P_T for all $k \geq 0$. Thus if T is infinite, then P_T is effectively locally determined. If T is finite, then P_T is finite so it automatically is effectively locally determined.

Next we consider the possible stable models of P_T. If T is finite, then there is a clause of the form

$$C = 2n + 1 \leftarrow \neg 2n + 1 \tag{5}$$

in P_T. Moreover this is the only clause involving $2n+1$. This means that P_T can have no stable model. That is, if M is a stable model of P_T, then we cannot have $2n + 1$ in M, since the only clause C which has $2n + 1$ as a conclusion cannot be used in a P, M-derivation. Thus we must assume that $2n + 1 \notin M$. But in that case, clause C is M-applicable, so that M would not be closed under all M-applicable clauses. Hence M is not a stable model.

Thus we are reduced to the case where T is infinite. In that case, T has nodes of length k for all $k \geq 0$ since T is finitely branching. This means that there are no clauses of type (4) in P_T. Now suppose that M is a stable model of P_T. We claim the clauses of type (2) and (3) ensure that there is exactly one node σ_k of length k such that the code of σ_k is in M. Clearly, there is only one clause C, namely the clause of type (1), that has $c(\emptyset)$ as its conclusion and since C has no body, $c(\emptyset)$ must be in M. Next suppose by induction that there is exactly one node σ of length k such that $c(\sigma)$ is in M. Then σ cannot be a terminal node due to the clause C_σ of type (3) with $c(\sigma)$ in its body. That is, C_σ is the only clause of with $2c(\sigma)-1$ as its conclusion. Then if $2c(\sigma)-1 \in M$, then C_σ is not M applicable so that there is no P, M-derivation of $2c(\sigma)-1$. If $2c(\sigma)-1 \notin M$, then C_σ is M-applicable and hence M is not closed under all M-applicable clauses which violates our assumption that M is a stable model. Thus it must be the case that σ has at least one successor.

Next it cannot be that none of codes of the successors of σ are in M since otherwise all the clauses of type (2) with $c(\sigma)$ in the body are M-applicable which would imply that M is not closed under all M-applicable clauses. On the other hand, if two or more of the codes of the successors of σ are in M, then all the clauses of type (2) with $c(\sigma)$ in the body are not M-applicable. Since these are the only clauses which have a code of successor of σ as its conclusion, there can be no P, M-derivation of a code of a successor of σ. Hence none of the codes of the successors of σ are in M which we have already shown is impossible. Thus M must contain the code of exactly one of the successors of σ. Moreover since σ is the unique node of length k such that $c(\sigma) \in M$, then any clause which has the code of a node in length $k+1$ as its conclusion, is M-applicable only if $c(\sigma)$ is in its body. Thus there is exactly one node τ of length $k+1$ such that $c(\tau) \in M$ and that node must be a successor of σ. Thus if M is stable model of P_T, then $M = \{c(\sigma) : \sigma \in \pi\}$ for some $\pi \in [T]$. On the other hand, it is easy to see that if $\pi \in [T]$, then $M = \{c(\sigma) : \sigma \in \pi\}$ is stable model of P_T. Thus there is an effective one-to-one degree preserving correspondence between the infinite paths through T and $\mathcal{S}(P_T)$.

We note that just as in the case for rps programs, Theorem 3.12 allows us to transfer all the results of about the degrees of elements of recursively bounded Π^0_1-classes to result about the degrees of the set of stable models of an effectively locally determined program. For example, we have the following.

Corollary 3.13. 1. There is an effectively locally determined recursive program P such that P has $2^{\aleph_0}$ stable models but no recursive stable model.

2. There is an effectively locally determined recursive program P such that P has $2^{\aleph_0}$ stable models and any two stable models $S_1 \neq S_2$ of P are Turing incomparable.

3. If $\mathbf{a}$ is any Turing degree such that $\mathbf{0} <_T \mathbf{a} \leq_T \mathbf{0}'$, then there is an effectively locally determined recursive program P such that P has $2^{\aleph_0}$ stable models but no recursive stable models and P has a stable model of degree $\mathbf{a}$.

4. If $\mathbf{a}$ is any Turing degree such that $\mathbf{0} <_T \mathbf{a} \leq_T \mathbf{0}'$, then there is an effectively locally determined recursive program P such that P has $\aleph_0$ stable models, P has a stable model S of degree $\mathbf{a}$ and if $S' \neq S$ is a stable model of P, then S' is recursive.

5. There is an effectively locally determined recursive program P such that P has $2^{\aleph_0}$ stable models and if $\mathbf{a}$ is the degree of any stable model S of P and $\mathbf{b}$ is any recursively enumerable degree such that $\mathbf{a} <_T \mathbf{b}$, then $\mathbf{b} \equiv_T \mathbf{0}'$.

6. If $\mathbf{a}$ is any recursively enumerable Turing degree, then there is an effectively locally determined recursive program P such that P has $2^{\aleph_0}$ stable models and the set of recursively enumerable degrees $\mathbf{b}$ which contain a stable model of P is precisely the set of all recursively enumerable degrees $\mathbf{b} \geq_T \mathbf{a}$.

4. Ensuring the existence of recursive stable models

In this section, we give two conditions which ensure the existence of recursive stable models.

Definition 4.1. 1. Let P be a locally determined logic program and let $lev(P) = \{l_0 < l_1 < \ldots\}$. Then we say say that P has the **level extension property** if for all k, whenever E_k is a stable model of P_{l_k}, there exists a stable model of E_{k+1} of $P_{l_{k+1}}$ such that $E_{k+1} \cap \{a_0, \ldots, a_{l_k}\} = E_k$.

2. A level n of P is a **strong level** of P if for any levels $m < n$ of P and any stable model E_m of P_m, if there is no stable model E of P with $E \cap \{a_0, \ldots, a_m\} = E_m$, then there is no stable model E_n of P_n with $E_n \cap \{a_0, \ldots, a_m\} = E_m$.

3. P has **effectively strong levels** if P has infinitely many strong levels and there is a computable function f such that for each i, $i \leq f(i)$ and $f(i)$ is a strong level.

Consider the effectively locally determined recursive program P_T of theorem 3.12 where T is an infinite highly recursive tree T contained in $\omega^{<\omega}$. By essentially the same argument that was use in the proof of

Theorem 3.12, one can show that if $k > 0$ and σ is a non-terminal node of length k in T, then $\{c(\tau) : \tau \prec \sigma\}$ is a stable model of $(P_T)_{2n_k}$ where recall that $2n_k$ is largest code of a node of length k in T. Now if all the nodes of length k in T are extendible to an infinite path through T, then $2n_k$ will be a strong level of P_T. If not, $2n_k$ is a level of P_T but not a strong level. Thus if T has the property that there are infinitely many k such that every non-terminal node of length k is extendible to an infinite path through T, then P_T will have effectively strong levels. However, if for every $k > 0$, there is a non-terminal node of length k which is not extendible to an infinite path through T, then P_T will not have strong levels. Since it easy to construct highly recursive trees with this property, there are many examples of programs which are effectively locally determined but do not have strong levels. It is also the case that if T has a non-terminal node of length $k > 0$ which is not extendible to an infinite path through T, then P_T will not have the level extension property.

In general, it is not easy to ensure that a locally determined logic program P has the level extension property. However, there are many natural examples where this condition is satisfied. One way to generate such programs is to consider the following result from recursive combinatorics. Bean [5] showed that there exists highly recursive connected graphs which are 3-colorable but not recursively k-colorable for any k. However, Bean also showed that every infinite connected k-colorable highly recursive graph G is recursively $2k$-colorable. Here a graph $G = (V, E)$ is highly recursive if the vertex set V is a recursive subset of ω, the set of codes of the edges $\{x, y\} \in E$ is recursive, G is locally finite, i.e. the degree of any vertex $v \in V$ is finite, and there is an effective procedure which given any vertex $v \in V$ produces a code of $N(x) = \{y \in V : \{x, y\} \in E\}$. The recursive $2k$-coloring of G can be produced as follows. Given any set $W \subseteq V$, let $N(W) = \{y \in V - W : (\exists x \in W)(\{x, y\} \in E)\}$. Then given any $x \in V$, define an effective increasing sequence of finite sets: $\emptyset = A_0, A_1, A_2, \ldots$ where $A_0 = N(x) \cup \{x\}$ and for all $k > 0$, $A_k = A_{k-1} \cup N(A_{k-1})$. It is easy to see that there can be no edge from an element of A_k to an element of $A_{k+2} - A_{k+1}$. Since G is k-colorable, the induced graph determined by $A_i - A_{i-1}$ is k-colorable for all $i \geq 1$. We then defined a recursive $2k$-coloring of G as follows.
(Step 1) Find a coloring of A_0 using colors $\{1, \ldots k\}$,
(Step 2) Find a coloring $A_1 - A_0$ using colors $\{k + 1, \ldots, 2k\}$,
(Step 3) Find a coloring of $A_2 - A_1$ using colors $\{1, \ldots k\}$,
(Step 4) Find a coloring $A_3 - A_2$ using colors $\{k + 1, \ldots, 2k\}$, etc.
One can easily write a logic program to implement this procedure and

it will naturally be effectively locally finite and have the level extension property. This is the subject of our next example.

Example 4.2. Let $G = \langle \omega, E \rangle$ be a highly recursive k-colorable graph. Suppose that we have renumbered the nodes of our graph G above so that our initial node x in the above algorithm is 0 and the elements of $A_0 - \{0\}$ are labeled with $1, 2, \ldots, |A_0|$. More generally, assume that under our renumbering, the elements of $A_n - A_{n-1}$ are labeled with $|A_{n-1}| + 1, \ldots, |A_n|$ for all $n > 0$. Our idea is to construct an effectively locally determined program P_G such that stable models of P_G correspond to $2k$-colorings of G that can be produced by the algorithm described above. The Herbrand base of P_G will consist of $\{2[n, j] : 1 \leq j \leq 2k\} \cup \{2n+1 : n \in \omega\}$ where $[,] : \omega \times \{1, \ldots, 2k\} \to \omega$ is a recursive pairing function given by $[n, j] = 2kn + j$. Our idea is to interpret the fact that $2[n, j]$ is in a stable model M as the fact that node n is colored with color j in the coloring corresponding to M. Now a coloring that can arise from our algorithm to color G with $2k$ colors would yield a coloring ψ such that $\psi : \omega \to \{1, \ldots, 2k\}$ where $\psi(n) \in \{1, \ldots, k\}$ if $n \in A_{2i} - A_{2i-1}$ for some i and $\psi(n) \in \{k+1, \ldots, 2k\}$ if $n \in A_{2i+1} - A_{2i}$ for some i. Here we set $A_{-1} = \emptyset$ by definition. We can construct an effectively local determined logic program P_G whose stable models are exactly the set $M_\psi = \{2[n, j] : \psi(n) = j\}$ for such a coloring ψ as follows. P_G consists of 4 sets of clauses.

(1) For any n such that there is an i, $n \in A_{2i} - A_{2i-1}$, and $j = 1, \ldots k$, we add the following clause.

$$2[n, j] \leftarrow \neg 2[n, 1], \ldots, \neg 2[n, j-1], \neg 2[n, j+1], \ldots, \neg 2[n, k].$$

(These clauses are designed to ensure that for any stable model M of P_G and for any $n \in A_{2i} - A_{2i-1}$, there is precisely one $j \in \{1, \ldots, k\}$ such that $2[n, j]$ is in M.)

(2) For all $m < n$ such that $\{m, n\} \in E$ and there is an i such that $m, n \in A_{2i} - A_{2i-1}$, we add the following clauses for $j = 1, \ldots, k$.

$$2[m, j] - 1 \leftarrow 2[m, j], 2[n, j], \neg 2[m, j] - 1.$$

(This clause will ensure that we cannot have both $2[m, j]$ and $2[n, j]$ in a stable model M of P_G so that any stable model of P_G will correspond to a proper coloring.)

(3) For any n such that there is an i, $n \in A_{2i+1} - A_{2i}$, and $j = 1, \ldots k$, we add the following clause.

$$2[n, k+j] \leftarrow \neg 2[n, k+1], \ldots, \neg 2[n, k+j-1], \neg 2[n, k+j+1], \ldots, \neg 2[n, 2k].$$

(These clauses are designed to ensure that for any stable model M of P_G and for any $n \in A_{2i+1} - A_{2i}$, there is precisely one $j \in \{1, \dots, k\}$ such that $2[n, k+j]$ is in M.)

(4) For all $m < n$ such that $\{m, n\} \in E$ and there is an i such that $m, n \in A_{2i+1} - A_{2i}$, we add the following clauses for $j = 1, \dots, k$.

$$2[m, k+j] - 1 \leftarrow 2[m, k+j], 2[n, k+j], \neg 2[m, k+j] - 1.$$

(This clause will ensure that we cannot have both $2[m, k+j]$ and $2[n, k+j]$ in a stable model M of P_G so that any stable model of P_G will correspond to a proper coloring.)

By using the same type of arguments that we used in Theorem 3.12, one can show that P_G is a recursive program whose stable models are precisely the set of $M_\psi = \{2[n, j] : \psi(n) = j$ where ψ is a proper $2k$-coloring of G such that for all n, $\psi(n) \in \{1, \dots, k\}$ if $n \in A_{2i} - A_{2i-1}$ for some i and $\psi(n) \in \{k+1, \dots, 2k\}$ if $n \in A_{2i+1} - A_{2i}$ for some i.

What are the possible levels of P_G. First it is easy see that the only possible levels of P_G are of the form $2[m, k]$ if for some $m \in A_{2i} - A_{2i-1}$ by the clauses in (1) and (2) and $2[m, 2k]$ if for some $m \in A_{2i+1} - A_{2i1}$ by the clauses in (3) and (4). Moreover by the clauses in (2), $2[m, k]$ is not a level if $m \in A_{2i} - A_{2i-1}$ and there is a $p \leq m$ such that $p \in A_{2i} - A_{2i-1}$ and there is some $n > m$ such that $n \in A_{2i} - A_{2i-1}$ and $\{p, n\} \in E$. However if there is no such $p, n \in A_{2i} - A_{2i-1}$, then $2[m, k]$ will be a level. In that case, it is not difficult to show that the stable models of $(P_G)_{2[m,k]}$ will correspond to the restriction of a stable model M_ψ for some coloring ψ as described above to $\{0, \dots, 2[m, k]\}$ and hence correspond to a proper coloring on the nodes $0, \dots, m$ of G which can be extended to a proper coloring of G. Similarly by the clauses in (4), $2[m, 2k]$ is not a level if $m \in A_{2i+1} - A_{2i}$ and there there is a $p \leq m$ such that $p \in A_{2i+1} - A_{2i}$ and there is some $n > m$ such that $n \in A_{2i+1} - A_{2i}$ and $\{p, n\} \in E$. Again if there is no such level, $2[m, 2k]$ will be a level of P_G and in that case, one can show that the stable models of $(P_G)_{2[m,2k]}$ will correspond to the restriction of a stable model M_ψ for some coloring ψ described above to $\{0, \dots, 2[m, 2k]\}$ and hence correspond to a proper coloring on the nodes $0, \dots, m$ of G which can be extended to a proper coloring of G. Thus P_G has the level extension property.

Finally it follows that $2[|A_{2n}|, k]$ and $2[|A_{2n+1}|, 2k]$ will be levels of P_G for all n so that P_G is effectively locally determined.

It also turns out that all locally determined FC-normal logic programs have the level extension property. That is, we can prove the following.

Theorem 4.3. Suppose that P is a locally determined FC-normal logic program. Then P has the level extension property.

Proof. Let $a_0, a_1, \dots$ be a listing of H_P such that P is locally determined with respect to this listing. For each $n \in \omega$, let $A_n = \{a_0, \dots, a_n\}$. Let n be a level of P and E_n be a stable model of P_n. Let $c_0, \dots, c_k$ be the set of E_n-applicable clauses of the $P_n - Horn(P_N)$. Next, extend this listing to a listing of all the clauses of $P - Horn(P)$, $c_0, \dots, c_k, c_{k+1}, \dots$. We can use this listing to define an ordering $\prec$ of order type ω on $P - Horn(P)$ by declaring that $c_i \prec c_j$ if and only if $i < j$. Now recall the countable forward chaining construction for P relative to $\prec$.

The countable forward chaining construction of $T^{\prec} = \bigcup_{n\in\omega} T_n^{\prec}$.

Stage 0. Let $T_0^{\prec} = T_{Horn(P)} \Uparrow \omega(\emptyset)$.

Stage $n+1$. Let $\ell(n+1)$ be the least $s \in \omega$ such that $c_s = \varphi \leftarrow \alpha_1, \dots, \alpha_k, \neg\beta_1, \dots, \neg\beta_m$ where $\alpha_1, \dots, \alpha_k \in T_n^{\prec}$ and $\beta_1, \dots, \beta_m, \varphi \notin T_n^{\prec}$. If there is no such $\ell(n+1)$, let $T_{n+1}^{\prec} = T_n^{\prec}$. Otherwise let

$$T_{n+1}^{\prec} = T_{Horn(P)} \Uparrow \omega(T_n^{\prec} \cup \{p_{\ell(n+1)}\}) \tag{6}$$

where $p_{\ell(n+1)}$ is the head of $c_{\ell(n+1)}$.

We claim that $T^{\prec}$ is a stable model of P such that $T^{\prec} \cap A_n = E_n$. Thus for any level q of P with $q > n$, $T^{\prec} \cap A_q = E_q$ is stable model of P_q which extends E_n so that P has the level extension property.

Note that by Theorem 3.2, it is enough to show that $T^{\prec} \cap A_n \subseteq E_n$ since we know that $D_n = T^{\prec} \cap A_n$ is a stable model of P_n. But for stable models of P_n, $D_n \subseteq E_n$ if and only if $D_n = E_n$ by Proposition 2.1. Thus we need only show by induction on s, that for all s, $T_s^{\prec} \cap A_n \subseteq E_n$.

First consider $T_0^{\prec} \cap A_n$. Since $T_0^{\prec} = T_{Horn(P)} \Uparrow \omega(\emptyset)$, if there is an element $x \in A_n$ such that $x \in T_0^{\prec} \cap A_n$, then there is a proof scheme ψ with conclusion x made up of entirely of Horn clauses. Thus $supp(\psi) \cap A_n = \emptyset$. But then since n is a level, there is a proof scheme ψ' of P_n such that $supp(\psi') \cap A_n = \emptyset$. But this means that ψ' is made up entirely of Horn clauses of P_n so that ψ' would witness that x is in every stable model of P_n. Thus $x \in E_n$ and $T_0^{\prec} \cap A_n \subseteq E_n$.

Now suppose that $T_s^{\prec} \cap A_n \subseteq E_n$ and there is an element $x \in A_n$ such that $x \in T_{s+1}^{\prec} - T_s^{\prec}$. We claim that it cannot be that $T_s^{\prec} \cap A_n = E_n$, since otherwise, $x \notin E_n$ and hence $T^{\prec} \cap A_n = D_n$ is a stable model of P which contains $E_n \cup \{x\}$. But then E_n and D_n would be stable models of P_n such that $E_n \subset D_n$ which is impossible by Proposition 2.1. Thus it must be the case that $T_s^{\prec} \cap A_n \subset E_n$, However it is easy to see that $E_n = T_{Horn(P_n)} \Uparrow \omega(\{h(c_0), \ldots, h(c_k)\}) \subseteq T_{Horn(P)} \Uparrow \omega(\{h(c_0), \ldots, h(c_k)\})$ where $h(c_i)$ is the head of clause c_i for $i = 1, \ldots, k$. Hence there must be some $i \leq k$ such that $h(c_i) \notin T_s^{\prec}$. But the c_i has the property that $h(c_i) \notin T_s^{\prec}$ and $cons(c_i) \cap T_s^{\prec} = \emptyset$. It follows that $\ell_{s+1} \leq i$ so that $T_{s+1}^{\prec} = T_{Horn(P)} \Uparrow \omega(T_s^{\prec} \cup \{h(c_{\ell_{s+1}})\})$ where $h(c_{\ell_{s+1}}) \in E_n$ since $c_{\ell_{s+1}}$ is one of the E_n applicable rules for P_n by construction. Thus there is a (P_n, E_n)-derivation of each element of $T_s^{\prec} \cup \{h(c_{\ell_{s+1}})\}$. It then easily follows that for each element of element y in $T_{Horn(P)} \Uparrow \omega(T_s^{\prec} \cup \{h(c_{\ell_{s+1}})\})$, there is a proof scheme ψ with conclusion y made up of entirely of clauses from $c_0, \ldots, c_k$ plus Horn clauses. Thus $supp(\psi) \cap (A_n - E_n) = \emptyset$. But then since n is a level, there must exist a proof scheme ψ' of P_n such that $supp(\psi') \cap (A_n - E_n) = \emptyset$. Thus ψ' is a proof scheme of P_n with conclusion y such that $supp(\psi') \cap (A_n - E_n) = \emptyset$ and hence $y \in E_n$. It follows by that $T_{s+1}^{\prec} \cap A_n \subseteq E_n$ as desired. Hence, by induction, we must have $T_s^{\prec} \cap A_n \subseteq E_n$ for all s so that $T^{\prec} \cap A_n \subseteq E_n$ as claimed.

Theorem 4.4. 1. Suppose that P is an effectively locally determined recursive logic program with the level extension property. Then for every level n and stable model E_n of P_n, there is a recursive stable model of E of P such that $E \cap \{a_0, \ldots, a_n\} = E_n$.

2. Suppose that P is a recursive logic program with effectively strong levels. Then for every level n and stable model E_n of P_n, if there is a stable model E of P with $E \cap \{a_0, \ldots, a_n\} = E_n$, then there is a recursive stable model of E of P such that $E \cap \{a_0, \ldots, a_n\} = E_n$.

Proof. For (1), fix a level n of P and a stable model E_n of P_n. Suppose that f is the function which witnesses the fact that P is effectively locally determined. Then let $b_0, b_1, b_2, \ldots$ be the sequence $n, f(n), f(f(n)), \ldots$. It is easy to see that our level extension property implies that we can effectively construct a sequence of sets

$$E_{b_0}, E_{b_1}, E_{b_2}, \ldots$$

such that (i) $E_{b_0} = E_n$, (ii) for all $j > 0$, E_{b_j} is a stable model of P_{b_j}, and (iii) for all $j \geq 0$, $E_{b_{j+1}} \cap \{a_0, \ldots, a_{b_j}\} = E_{b_j}$. Now consider tree T and the nodes $\sigma_{E_{b_j}}$ as constructed in Corollary 3.10. It is easy to check that for all i, $\sigma_{E_{b_i}} \in T$ and that $\sigma_{E_{b_0}} \sqsubseteq \sigma_{E_{b_1}} \sqsubseteq \sigma_{E_{b_2}} \sqsubseteq \ldots$. It follows that there is a unique path β in $[T]$ which extends all $\sigma_{E_{b_i}}$ and that $E_\beta = \bigcup_{i \geq 0} \sigma_{E_{b_i}}$ is a stable model of P. Moreover E_β is recursive because to decide if $a_j \in E_\beta$, one need only find k such that $b_k \geq j$, in which case, $a_j \in E_\beta$ if and only if $a_j \in \sigma_{E_{b_k}}$.

For (2), assume that f is the recursive function which witnesses the fact that P has strong levels and let $b_0, b_1, b_2, \ldots$ be defined as above. We claim that the property of strong levels once again lets us construct an effective sequence $E_{b_0}, E_{b_1}, E_{b_2}, \ldots$ such that (a) $E_{b_0} = E_n$, (b) for all $j > 0$, E_{b_j} is a stable model of P_{b_j}, and (c) for all $j \geq 0$, $E_{b_{j+1}} \cap \{a_0, \ldots, a_{b_j}\} = E_{b_j}$. That is, suppose that we have constructed E_{b_k} such that there exists a stable model S of P such that $S \cap \{a_0, \ldots, a_{b_k}\} = E_{b_k}$. Now consider the strong level b_{k+2}. Our definition of strong level ensures that there must be some stable model F_{k+2} of $P_{b_{k+2}}$ such that $F_{k+2} \cap \{a_0, \ldots, a_{b_k}\} = E_{b_k}$. Then let $E_{b_{k+1}} = F_{k+2} \cap \{a_0, \ldots, a_{b_{k+1}}\}$. The argument in Theorem 3.2 shows that $E_{b_{k+1}}$ is a stable model of $P_{b_{k+1}}$. Moreover, since b_{k+2} is a strong level, there must be a stable model S' of P such that $E_{b_{k+1}} = S' \cap \{a_0, \ldots, a_{b_{k+1}}\}$ since otherwise, there can be no stable model F of $P_{b_{k+2}}$ such that $E_{b_{k+1}} = F \cap \{a_0, \ldots, a_{b_{k+1}}\}$. This given, we can then construct our desired recursive stable model E_β exactly as in (1).

We end this section with another, more direct approach to producing a recursive stable model.

Definition 4.5. We say that a recursive logic program P has **witnesses with effective delay** if there is a recursive function f such that for all n, $f(n) > n$ and whenever there is a set $S \subseteq \{a_0, \ldots, a_n\}$ such that there is a stable model E of P with $E \cap \{a_0, \ldots, a_n\} = S$ but there is no stable model F of $\mathcal{U}$ such that $F \cap \{a_0, \ldots, a_n, a_{n+1}\} = S$, then either

(i) there is a proof scheme ψ' with $max(\psi') \leq n+1$ such that $cln(\psi') = a_{n+1}$ and $supp(\psi') \subseteq \{a_0, \ldots, a_n\} - S$, or

(ii) for all sets $T \subseteq \{a_{n+2}, \ldots, a_{f(n)}\}$, there is a proof scheme ψ_T with $max(\psi) \leq f(n)$ such that $supp(\psi_T) \subseteq \{a_0, \ldots, a_{f(n)}\} - (T \cup S)$ and $cln(\psi_T) \in \{a_0, \ldots, a_{f(n)}\} - (T \cup S)$.

Note that in case (i), the proof scheme ψ' witnesses that we must have a_{n+1} in any stable model E such that $E \cap \{a_0, \ldots, a_n\} = S$. In case (ii), the proof schemes ψ_T show that we can not have a stable model E of P such that $E \cap \{a_0, \ldots, a_{f(n)}\} = S \cup T$ so that we are again forced to have a_{n+1} in any stable model E such that $E \cap \{a_0, \ldots, a_n\} = S$.

Theorem 4.6. Suppose that P is a recursive logic program which has witnesses with effective delay and has at least one stable model. Then the lexicographically least stable model E of P is recursive.

Proof. We can construct the lexicographically least stable model E of P by induction as follows.

Suppose that for any given n we have constructed $E_n = E \cap \{a_0, \ldots, a_n\}$. Then $E_{n+1} = E_n$ unless either

(i) there is a proof scheme ψ of level n such that $supp(\psi) \subseteq \{a_0, \ldots, a_n\} - E_n$ and $cln(\psi) = a_{n+1}$ or

(ii) for all sets $T \subseteq \{a_{n+2}, \ldots, a_{f(n)}\}$, there is a proof scheme ψ_T with $max(\psi_T) \leq f(n)$ such that $supp(\psi_T) \subseteq \{a_0, \ldots, a_{f(n)}\} - (T \cup E_n)$ and $cln(\psi_T) \in \{a_0, \ldots, a_{f(n)}\} - (T \cup E_n)$.

in which case $E_{n+1} = E_n \cup \{a_{n+1}\}$. Note that since there are only finitely many minimal proof schemes ψ with $max(\psi) \leq k$ for any given k, we can check conditions (i) and (ii) effectively. Since there is a stable model, it is easy to see that our definitions insure that E_n is always contained in the lexicographically least stable model of P. Thus $E = \bigcup_n E_n$ is recursive.

5. Finding Stable Models with Low Complexity

Throughout this section, we shall assume that P is a recursive logic program such that $H_P = \omega = \{0, 1, 2, \ldots\}$. Moreover if P is locally determined, we let $\{l_0 < l_1 < \ldots\}$ denote the set of levels of P and if P has strong levels, then we let $\{s_0 < s_1 < \ldots\}$ denote the set of strong levels of P.

In this section, we shall show how we can use the notions of levels and strong levels to provide conditions which will ensure that P has a stable model which has relatively low complexity. We shall distinguish two representations of P, namely the tally representation of P, $Tal(P)$, and the binary representation of P, $Bin(P)$. In the tally representation of P, we shall identify each $n \in H_P$, with its tally representation, $tal(n)$, where $tal(0) = 0$ and $tal(n) = 1^n$ for $n \neq 0$. In the binary representation of P, we shall identify each natural number n with its binary representation, $bin(n)$. Given a clause $c = \varphi \leftarrow \alpha_1, \ldots, \alpha_n \neg\beta_1, \ldots, \neg\beta_m$, we let the tally and binary representations of r be given by

$$\begin{aligned} tal(c) &= 2tal(\alpha_1)2 \ldots 2tal(\alpha_n)3tal(\beta_1)2 \ldots 2tal(\beta_m)3tal(\varphi) \\ bin(c) &= 2bin(\alpha_1)2 \ldots 2bin(\alpha_n)3bin(\beta_1)2 \ldots 2bin(\beta_m)3bin(\varphi) \end{aligned}$$

We then let $Tal(P) = \{tal(c) : c \in P\}$ and $Bin(P) = \{bin(c) : c \in P\}$.

Given a finite set U $\{u_0 < \ldots < u_k\}$ of H_P, let the tally and binary representation of U be given by

$$\begin{aligned} tal(U) &= 4tal(u_0)4 \ldots 4tal(u_k)4 \\ bin(\psi) &= 4bin(u_0)4 \ldots 4bin(u_k)4 \end{aligned}$$

Similarly given a proof scheme $\psi = \langle\langle p_1, C_1, U_1\rangle, \ldots, \langle p_n, C_n, U_n\rangle\rangle$, we let the tally and binary representations of ψ be given by

$$\begin{aligned} tal(\psi) &= 5tal(p_1)6tal(C_1)7tal(U_1)5\ldots 5tal(p_n)6tal(C_n)7tal(U_n)5 \\ bin(\psi) &= 5bin(p_1)6bin(C_1)7bin(U_1)5\ldots 5bin(p_n)6bin(C_n)7bin(U_n)5 \end{aligned}$$

Finally given a finite set of proof schemes proof $\Gamma = \{\psi_1, \ldots, \psi_s\}$, we let the tally and binary representations of Γ be given by

$$\begin{aligned} tal(\Gamma) &= 8tal(\psi_1)8\ldots 8tal(\psi_s)8 \\ bin(\Gamma) &= 8bin(\psi_1)8\ldots 8bin(\psi_s)8 \end{aligned}$$

Definition 5.1. We say that the logic program P is **polynomial time locally determined in tally** if $Tal(P)$ has the following properties.

1. There is a polynomial time function g such that for any i, $g(tal(i)) = tal(l_{k_i})$ where k_i is the least number k such that $l_k \geq i$.

2. There is a polynomial time function h such that for any i and $S \subseteq \{tal(0), \ldots, tal(l_{k_i})\}$ where $tal(l_{k_i}) = g(tal(i))$, $h(tal(i), tal(S)) = tal(\psi)$ where ψ is a minimal proof scheme ψ of $P_{l_{k_i}}$ such that $cln(\psi) = i$ and $supp(\psi) \cap S = \emptyset$ if such a ψ exists and $h(tal(i), tal(S)) = tal(0)$ otherwise.

Similarly we say that P is polynomial time locally determined in binary, if definition (5.1) holds where we uniformly replace all tally representations by binary representations. We can also define the notions of P being linear time, exponential time, polynomial space, etc. in tally or binary in a similar manner.

This given, we then have the following.

Theorem 5.2. 1. Suppose that P is a polynomial time locally determined logic program in tally which has the level extension property. If l is a level of P and M_l is a stable model of P_l such that there is a unique stable model M of P with $M \cap \{tal(0), \ldots, tal(l)\} = M_l$, then $M \in NP \cap Co\text{-}NP$.

2. Suppose that P is a polynomial space locally determined logic program in tally which has the level extension property. If l is a level of P and M_l is a stable model of P_l such that there is a unique stable model M of P with $M \cap \{0, \ldots, l\} = M_l$, then $E \in PSPACE$

3. Suppose that P is a polynomial time locally determined logic program in binary which has the level extension property. If l is a level of P and M_l is a stable model of P_l such that there is a unique stable model M of P with $M \cap \{bin(0), \ldots, bin(l)\} = M_l$, then $N \in NEXPTIME \cap Co\text{-}NEXPTIME$.

4. Suppose that P a is polynomial space locally determined logic program in binary which has the level extension property. If l is a level of P and M_l is a stable model of P_l such that there is a unique stable model M of P with $E \cap \{bin(0), \ldots, bin(l)\} = M_l$, then $M \in \bigcup_{c \geq 0}(DSPACE(2^{n^c})$.

Proof. For (1), suppose that $l = l_t$ where recall that the set of levels of P is $\{l_0 < l_1 < \ldots\}$. Then for any $i > l_t$, consider the level l_{k_i} where $g(tal(i)) = tal(l_{k_i})$. By the level extension property, it follows that there is a stable model, $M_{l_{k_i}}$, of $P_{l_{k_i}}$ such that $M_{l_{k_i}} \cap \{tal(0), \ldots, tal(l)\} = M_l$. Moreover, it must be the case that $M \cap \{tal(0), \ldots, tal(l_{k_i})\} = M_{l_{k_i}}$ since otherwise we could use the level extension property to show that there is a sequence of stable models $\{M_{l_j} : j \geq k_i\}$ such that for each $j > k_i$, M_{l_j} is a stable model of P_{l_j} where $M_{l_j} \cap \{tal(0), \ldots, tal(l_{j-1})\} = M_{l_{j-1}}$. One can then easily prove that $M' = \cup_{j \geq k_i} M_{l_j}$ is stable model of P such that $M' \cap \{tal(0), \ldots, tal(l)\} = M_l$ contradicting our assumption that M is the unique stable model of P such that $M \cap \{tal(0), \ldots, tal(l)\} = M_l$.

It follows that to decide if $i \in M$, we need only guess $M_{l_{k_i}}$, verify that it is a stable model of $P_{l_{k_i}}$, and check whether $i \in M_{l_{k_i}}$. We claim that this is an NP process. That is, we first guess the sequence $\chi_{M_{l_{k_i}}}(tal(0)) \ldots \chi_{M_{l_{k_i}}}(tal(l_{k_i}))$ where $\chi_{M_{l_{k_i}}}$ is the characteristic function of $M_{l_{k_i}}$. Note that our conditions ensure that there is some polynomial p such that $l_{k_i} \leq p(|tal(i)|)$. It follows that we can compute $l_{k_0} = g(tal(0)), l_{k_1} = g(tal(1)), \ldots, l_{k_{l_{k_i}}} = g(tal(l_{k_i}))$ in polynomial time in $|tal(i)|$. Since for each j, l_{k_j} is the least level greater than or equal to j, it follows that $l_{k_0} \leq l_{k_1} \leq \ldots \leq l_{k_{l_{k_i}}} = l_{k_i}$. Thus if $k_i = s$, we can find $l_0 < l_1 < \ldots < l_s$ is polynomial time in $|tal(i)|$. Note by assumption, $t < s$. Now consider $M_r = M_{l_{k_i}} \cap \{tal(0), \ldots, tal(l_r)\}$ for $r = t, t+1, \ldots, s$. By our definition of levels, it must be the case that each M_r is a stable model of P_{l_r}. We claim that this can be checked in polynomial time. That is, first, we can check that $M_t = M_l$. Now assume by induction that we have checked that for a given $r \in \{t+1, \ldots, s\}$ that M_r is a stable model of P_{l_r}. Then consider an x such that $l_{r-1} < x < l_r$. Now if $x \notin M_r$, there can be no proof scheme ψ of P_{l_r} such that $cln(\psi) = x$ and $supp(\psi) \cap M_r = \emptyset$ since otherwise ψ would witness that $x \in M_s$ and hence $h(tal(x), tal(M_r))$ must be equal to $tal(0)$. Vice versa, if $x \in M_r$, then since l_r is a level, there must be a proof scheme ψ_x such that $max(\psi_x) \leq l_r$, $cln(\psi_x) = x$, and $supp(\psi_x) \cap M_r = \emptyset$. Hence $h(tal(x), tal(M_r))$ must equal $tal(\phi)$ where ϕ is a proof scheme of P_{l_r} such that $cln(\phi) = x$ and $supp(\phi) \cap M_r = \emptyset$. Since we can compute $h(tal(x), tal(M_r))$ for each $l_{r-1} < x \leq l_r$ in polynomial time in $|tal(i)|$, it follows that we can check whether M_r is stable model of P_{l_r} for each $r = t, \ldots, s$ in polynomial time in $|tal(i)|$. Hence it follows that $M \in NP$. However since M is unique, it automatically follows that $M \in Co\text{-}NP$.

The proof of part (2) is similar. However since in this case, the length of the sequence $\chi_{M_{l_{k_i}}}(0) \ldots \chi_{M_{l_{k_i}}}(l_{k_i})$ is bounded by $p(|tal(i)|)$ for some polynomial p, we do not have to guess it. That is, in $p(i)$ space, we check all strings of $\{0,1\}^{l_{k_i}}$ to see if they are the characteristic function of a stable model M^* of $P_{l_{k_i}}$ such that $M^* \cap \{tal(0), \ldots, tal(l)\} = M_l$. Since there is only one such stable model with this property, we can search until we find it. Thus our computations will require only polynomial space.

The proof of parts (3) and (4) uses the same algorithms as in parts (1) and (2). However in this case the string

$$\chi_{M_{l_{k_i}}}(bin(0)) \ldots \chi_{M_{l_{k_i}}}(bin(l_{k_i}))$$

may be as long as $2^{q(|bin(i)|)}$ for some polynomial q. Thus the algorithm could take on the order of $2^{|bin(i)|^c}$ steps in case (3) and require $2^{|bin(i)|^c}$ space in case (4).

We should note that if we replace the hypothesis of polynomial time and polynomial space by linear time and linear space in parts (3) and (4) of Theorem (5.2) respectively, then we get the following.

Theorem 5.3. 1. Suppose that P is linear time locally determined logic program in binary which has the level extension property. If l is a level of P and M_l is a stable model of P_l such that there is a unique stable model M of P with $M \cap \{bin(0), \ldots, bin(l)\} = M_l$, then $M \in NEXP \cap Co\text{-}NEXP$.

2. Suppose that P is linear space locally determined logic program in binary which has the level extension property. If l is a level of P and M_l is a stable model of P_l such that there is a unique stable model M of P with $M \cap \{bin(0), \ldots, bin(l)\} = M_l$, then $M \in EXPSPACE$.

It is easy to show that we can weaken the hypothesis in Theorems 5.2 and 5.3 that there is a unique stable model M of P extending M_l to the assumption that there are only finitely many stable models of P extending M_l and obtain the conclusion that all of the stable models of M_l are in the same corresponding complexity classes. However, if we do not make any assumption about the number of stable models of P which extend M_l, then the only thing we can do is try to construct the lexicographically least stable model of M_l. One can see that in cases (2) and (4) there would be no change in the conclusion. However in case (1), the computations could take 2^{n^c} steps and in case (3), the computations could require $2^{2^{n^c}}$ steps.

Finally we note that similar results can be proven using strong levels instead of the level extension property. We state the appropriate definitions and results without proof. Recall that if P has strong levels, then we let $\{s_0 < s_1 < \ldots\}$ denote the set of all strong levels of P.

Definition 5.4. We say that the logic program P has polynomial time strong levels in tally, if the logic program $Tal(P)$ has strong levels and the following properties.

1. There is a polynomial time function g such that for any i, $g(tal(i)) = tal(s_{k_i})$ where k_i is the least number k such that $s_k \geq i$.

2. There is a polynomial time function h such that for any i and $S \subseteq \{tal(0), \ldots, tal(l_{s_i})\}$ where $tal(l_{s_i}) = g(tal(i))$, $h(tal(i), tal(S)) = tal(\psi)$ where ψ is a minimal proof scheme ψ of $P_{l_{s_i}}$ such that $cln(\psi) = i$ and $supp(\psi) \cap S = \emptyset$ if such a ψ exists and $h(tal(i), tal(S)) = tal(0)$ otherwise.

The definition of a logic program P having polynomial time strong levels in binary is obtained by replacing $Tal(P)$ by $Bin(P)$ and all tally representations by binary representations in the definition above. Similar definitions may be given for space complexity as in Definition 5.1.

This given, we then have the following.

Theorem 5.5. 1. Suppose that P is a logic program which has polynomial time strong levels in tally. If l is a level of P and M_l is a stable model of P_l such that there is a unique stable model M of P with $M \cap \{tal(0), \ldots, tal(l)\} = M_l$, then $M \in NP \cap Co\text{-}NP$.

2. Suppose that P is a logic program which has polynomial space strong levels in tally. If l is a level of P and M_l is a stable model of P_l such that there is a unique stable model M of P with $M \cap \{tal(0), \ldots, tal(l)\} = M_l$, then $M \in PSPACE$.

3. Suppose that P is a logic program which has polynomial time strong levels in binary. If l is a level of P and M_l is a stable model of P_l such that there is a unique stable model M of P with $M \cap \{bin(0), \ldots, bin(l)\} = M_l$, then $M \in NEXPTIME \cap Co\text{-}NEXPTIME$.

4. Suppose that P is a logic program which has polynomial space strong levels in binary. If l is a level of P and M_l is a stable model of P_l such that there is a unique stable model M of P with $M \cap \{bin(0), \ldots, bin(l)\} = M_l$, then $M \in \bigcup_{c \geq 0}(DSPACE(2^{n^c}))$.

Definition 5.6. 1. We say that a logic program P is polynomial time in tally with **witnesses of polynomial time delay in tally** if there is a polynomial time function $f : Tal(\omega) \to Tal(\omega)$ such that for all n, $f(tal(n)) > tal(n)$ and whenever there is a set $S \subseteq \{tal(0), \ldots, tal(n)\}$ such that there is a stable model E of $Tal(P)$ with $E \cap \{tal(0), \ldots, tal(n)\} = S$ but there is no stable model F of $Tal(Q)$ such that $F \cap \{tal(0), \ldots, tal(n), tal(n+1)\} = S$, then either

(i) there is a proof scheme ψ with $max(\psi) \leq n+1$ such that $cln(\psi) = tal(n+1)$ and $supp(\psi) \subseteq \{tal(0), \ldots, tal(n)\} - S$, or

(ii) for all sets $T \subseteq \{tal(n+2), \ldots, f(tal(n))\}$, there is a proof scheme ψ_T with $tal(max(\psi_T)) \leq f(tal(n))$ such that $supp(\psi_T) \subseteq \{tal(0), \ldots, f(tal(n))\} - (T \cup S)$ and $cln(\psi_T) \in \{tal(0), \ldots, f(tal(n))\} - (T \cup S)$.

2. There is a polynomial time function h such that for any i, $h(tal(i)) = tal(\Gamma_i)$ where for all $S \subseteq \{tal(0), \ldots, tal(i)\}$, Γ_i contains the lexicographically least minimal proof scheme ψ of P_i such that $cln(\psi) = tal(i)$ and $supp(\psi) = S$ if such a ψ exists.

Similarly we say a logic program Q is polynomial time in binary with *witnesses of polynomial time delay in binary* if (1) and (2) of the definition of a logic program being polynomial time in tally with *witnesses of polynomial time delay in tally* hold where we replace the tally representation by the binary representation.

Theorem 5.7. 1. Suppose that P is a polynomial time logic program in tally which has witnesses with polynomial time delay in tally and has at least one stable model. Then the lexicographically least stable model E of P is polynomial time.

2. Suppose that P is a polynomial time logic program in binary which has witnesses with polynomial time delay in binary and has at least one stable model. Then the lexicographically least stable model E of P is EXPTIME.

Proof. For part (1), let $f : Tal(\omega) \to Tal(\omega)$ be the function which witnesses that P is a polynomial time logic program in tally which has witnesses with polynomial time delay in tally. We can construct the lexicographically least stable model E of P by induction as follows.

Suppose that for any given n we have constructed

$$E_n = E \cap \{tal(0), \ldots, tal(n)\}.$$

Then $E_{n+1} = E_n$ unless either

(i) there is a proof scheme ψ of with $max(\psi) \leq n+1$ such that $supp(\psi) \subseteq \{tal(0), \ldots, tal(n)\} - E_n$ and $cln(\psi) = tal(n+1)$ or

(ii) for all sets $T \subseteq \{tal(n+2), \ldots, f(tal(n))\}$, there is a proof scheme ψ_T with $tal(max(\psi_T)) \leq f(tal(n))$ such that $supp(\psi_T) \subseteq \{tal(0), \ldots, f(tal(n))\} - (T \cup E_n)$ and $cln(\psi_T) \in \{tal(0), \ldots, f(tal(n))\} - (T \cup E_n)$.

in which case $E_{n+1} = E_n \cup \{tal(n+1)\}$.

Note that we can check conditions (i) and (ii) effectively. That is, we can compute $h((f(tal(n))) = tal(\Gamma_{f(tal(n)})$ in polynomial time in $|tal(n)|$. Thus for all $S \subseteq \{tal(0), \ldots, f(tal(n))\}$, $\Gamma_{f(tal(n))}$ contains the lexicographically least minimal proof scheme ψ of $P_{f(tal(n))}$ such that $cln(\psi) = tal(i)$ and $cons(\psi) = S$ is such a ψ exists. Here $h(tal(i)) = tal(\Gamma_i)$ is the polynomial time function which witnesses that P is polynomial time in tally with witnesses with polynomial time delay in tally. Thus $\Gamma_{f(tal(n))}$ contains at least one proof scheme of every possible support for $P_{f(tal(n))}$ so that we can check conditions (i) and (ii) by simply checking the proof scheme in $\Gamma_{f(tal(n))}$. It follows that we can check whether conditions (i) or (ii) hold in polynomial time in $|tal(n)|$. Thus we can extend E_n to E_{n+1} in polynomial time in $|tal(n)|$ and hence we can compute $E_0, E_1, \ldots, E_{n+1}$ in polynomial time in $|tal(n)|$. Since there is a stable model, it is easy to see that our definitions insure that E_n is always contained in the lexicographically least stable model of P. Thus $E = \bigcup_n E_n$ is polynomial time.

The proof of part (2) is similar.

The problem with Definition 5.6 is that condition 2 is very restrictive. That is, in principle, Γ_i could be required to contain 2^i proof schemes for each i which would mean that the function $h(tal(i)) = tal(\Gamma_i)$ could not possibly be polynomial time. However, in the case where the delay is bounded by a constant, that is, when there is a fixed constant k such that $tal(n+k) \geq f(tal(n)) > tal(n)$ for all n, we can weaken condition 2 and get the same conclusion.

Definition 5.8. 1. We say that a logic program P is polynomial time in tally with **polynomial time witnesses of constant delay in**

tally if there is a fixed constant k and a polynomial time function $f : Tal(\omega) \to Tal(\omega)$ such that for all n, $tal(n+k) \geq f(tal(n)) > tal(n)$ and whenever there is a set $S \subseteq \{tal(0), \dots, tal(n)\}$ such that there is a stable model E of $Tal(P)$ with $E \cap \{tal(0), \dots, tal(n)\} = S$ but there is no stable model F of $Tal(Q)$ such that $F \cap \{tal(0), \dots, tal(n), tal(n+1)\} = S$, then either

(i) there is a proof scheme ψ with $max(\psi) \leq n+1$ such that $cln(\psi) = tal(n+1)$ and $supp(\psi) \subseteq \{tal(0), \dots, tal(n)\} - S$, or

(ii) for all sets $T \subseteq \{tal(n+2), \dots, f(tal(n))\}$, there is a proof scheme ψ_T with $tal(max(\psi_T)) \leq f(tal(n))$ such that $supp(\psi_T) \subseteq \{tal(0), \dots, f(tal(n))\} - (T \cup S)$ and $cln(\psi_T) \in \{tal(0), \dots, f(tal(n))\} - (T \cup S)$.

2. There is a polynomial time function h such that for any $i < n$ and any $S \subseteq \{tal(0), \dots, tal(n)\}$, $h(tal(i), tal(n), tal(S))) = tal(\psi)$ where ψ is a minimal proof scheme of P_n such that $cln(\psi) = tal(i)$ and $supp(\psi) \cap S = \emptyset$ if such a ψ exists and $h(tal(i), tal(n), tal(S))) = tal(0)$ otherwise.

Similarly we say a logic program Q is polynomial time in binary with *polynomial time witnesses of constant delay in binary* if (1) and (2) of the definition of a logic program being polynomial time in tally with *witnesses of polynomial time delay in tally* hold where we replace the tally representation by the binary representation.

Theorem 5.9. 1. Suppose that P is a polynomial time logic program in tally which has polynomial time witnesses with constant delay in tally and has at least one stable model. Then the lexicographically least stable model E of P is polynomial time.

2. Suppose that P is a polynomial time logic program in binary which has polynomial time witnesses with constant delay in binary and has at least one stable model. Then the lexicographically least stable model E of P is EXPTIME.

Proof. For part (1), let $f : Tal(\omega) \to Tal(\omega)$ be the function which witnesses that P is a polynomial time logic program in tally which has witnesses with polynomial time delay in tally. We can construct the lexicographically least stable model E of P by induction as follows.

Suppose that for any given n we have constructed

$$E_n = E \cap \{tal(0), \dots, tal(n)\}.$$

Then $E_{n+1} = E_n$ unless either

(i) there is a proof scheme ψ of with $max(\psi) \leq n+1$ such that $supp(\psi) \subseteq \{tal(0), \ldots, tal(n)\} - E_n$ and $cln(\psi) = tal(n+1)$ or

(ii) for all sets $T \subseteq \{tal(n+2), \ldots, f(tal(n))\}$, there is a proof scheme ψ_T with $tal(max(\psi_T)) \leq f(tal(n))$ such that $supp(\psi_T) \subseteq \{tal(0), \ldots, f(tal(n))\} - (T \cup E_n)$ and $cln(\psi_T) \in \{tal(0), \ldots, f(tal(n))\} - (T \cup E_n)$.

in which case $E_{n+1} = E_n \cup \{tal(n+1)\}$.

Note that we can check conditions (i) and (ii) effectively. That is, to check condition (i), we simply compute $h(tal(n+1), tal(n+1), tal(E_n \cup \{tal(n+1)\}))$. If $h(tal(n+1), tal(n+1), tal(E_n \cup \{tal(n+1)\})) = tal(0)$, then there is no proof scheme ψ of with $max(\psi) \leq n+1$ such that $supp(\psi) \subseteq \{tal(0), \ldots, tal(n)\} - E_n$ and $cln(\psi) = tal(n+1)$. Otherwise, $h(tal(n+1), tal(n+1), tal(E_n \cup \{tal(n+1)\})) = tal(\psi)$ where ψ is a proof scheme $max(\psi) \leq n+1$ such that $supp(\psi) \subseteq \{tal(0), \ldots, tal(n)\} - E_n$ and $cln(\psi) = tal(n+1)$. Similarly, to check condition (ii), we compute $h(tal(i), tal(f(n)), tal(E_n \cup T))$ for all $T \subseteq \{tal(n+2), \ldots, f(tal(n))\}$ and all $i \in \{tal(0), \ldots, f(tal(n))\} - (T \cup E_n)$. Note that since $tal(n) < f(tal(n)) \leq tal(n+k)$, this means that we need only compute at most $2^{k-1} \cdot |f(tal(n))|$ possible values of h where each of the arguments can be computed in polynomial time in $|tal(n)|$. It follows that we can compute all these values of h in polynomial time in $|tal(n)|$. Note that condition (ii) will hold unless there is a fixed $T \subseteq \{tal(n+2), \ldots, f(tal(n))\}$ such that $h(tal(i), tal(f(n)), tal(E_n \cup T)) = tal(0)$ for all $i \in \{tal(0), \ldots, f(tal(n))\} - (T \cup E_n)$. It follows that we can check whether conditions (i) or (ii) hold in polynomial time in $|tal(n)|$. Thus we can extend E_n to E_{n+1} in polynomial time in $|tal(n)|$ and hence we can compute $E_0, E_1, \ldots, E_{n+1}$ in polynomial time in $|tal(n)|$. Since there is a stable model, it is easy to see that our definitions insure that E_n is always contained in the lexicographically least stable model of P. Thus $E = \bigcup_n E_n$ is polynomial time.

The proof of part (2) is similar.

6. Characterizing the set of stable models of a locally determined programs

In this section, we shall provide a characterization of the set of stable models of an effectively locally determined recursive logic program P. For arbitrary propositional logic programs, the possible sets of stable models were characterized by A. Ferry [10] in terms of an inverse-Scott topology.

Let $\mathcal{S}(P)$ be the set of all stable models of P. Our first observation is that if P is locally determined, then $\mathcal{S}(P)$ is closed in the natural (Cantor) topology on subsets of H_P. We note that it was shown in [16] that, in general, $\mathcal{S}(P)$ is a Π^0_2 set.

Proposition 6.1. If P is locally determined, then $\mathcal{S}(P)$ is a closed set.

Proof. Let $H_P = \{u_0, u_1, \ldots\}$ where $u_0 <_P u_1 <_P \ldots$ is some ordering of H_P which witnesses that P is locally determined. Let $E_1, E_2, \ldots$ be a sequence of stable models with limit E in the usual product topology on sets. Suppose that $u_i \in E$. Then there must be some K such that $u_i \in E_k$ for all $k \geq K$. Thus for each $k \geq K$, there is a proof scheme θ_k such that $cln(\theta_k) = u_i$ and $E_k \cap supp(\theta_k) = \emptyset$. Now let l be the least level $\geq i$. Then since P is locally determined, it follows that for each $k \geq K$, there is a minimal proof scheme ψ_k of P_l such that $cln(\psi_k) = u_i$ and $E_k \cap supp(\psi_k) = \emptyset$. But there are only finitely many possible support sets for such minimal proof schemes in P_l so that infinitely many of the ψ_k have the same support S. However since $S \cap E_k = \emptyset$ for infinitely many k, it must be the case that $S \cap E = \emptyset$ and hence $u_i \in E$. Vice versa, suppose that $u_i \notin E$. Thus there must be some K such that for all $k > K$, $u_i \notin E_k$. Suppose, by way of contradiction, that there is a proof scheme ψ with $cln(\psi) = u_i$ and $supp(\psi) = S$ with $S \cap E = \emptyset$. Since S is finite, there must be some M such that $S \cap E_m = \emptyset$ for all $m \geq M$. But this would mean that ψ would witness that $u_i \in E_m$ for all $m \geq M$, contradicting our previous assumption. (This direction applies even if P is not locally determined.)

We should note however that, in general, $\mathcal{S}(P)$ is not a closed set for a logic program P.

Example 6.2. Let P consist of the following set of clauses.

$$u_k \leftarrow \neg u_{2^k(2n+1)} \tag{7}$$

for all n and k. Thus $H_P = \{u_0, u_1, \ldots\}$. This means that for any stable model E of P and any k, $u_k \in E$ if and only if at least one of the set $\{u_{2^k(2n+1)} : n = 0, 1, \ldots\}$ is not in E. It is not hard to see that for any k, there will be a stable model E of P which contains all of $\{u_0, u_1, \ldots, u_k\}$. Thus if $\mathcal{S}(P)$ were closed, then H_P itself would be an stable model, which is clearly false, since none of the clauses of P are H_P-applicable.

We say that a family $\mathcal{E}$ of sets is **noninclusive** if for any two sets $A, B \in \mathcal{E}$, neither $A \subseteq B$ nor $B \subseteq A$. By Proposition 2.1, we know that the set of stable models of logic program is a noninclusive family of sets.

However, not every noninclusive family of sets can be the set of stable models of a logic program as the following example shows.

Example 6.3. Let $\mathcal{E} = \{\{u_i\} : i = 0, 1, \dots\}$, that is, the family of all singleton sets. This is clearly noninclusive. Now suppose that $\mathcal{E}$ were the set of stable models of some logic program P. For the stable model $\{u_0\}$, there must be a proof scheme ψ with finite support S and conclusion u_0. Now just choose some $u_k \notin S$. Then ψ is also E_k-applicable where $E_k = \{u_k\}$. But then E_k is not closed under all E_k-applicable clauses of P so that E_k could not be a stable model of P.

It was shown in [10] that a family of sets is the set of stable models of a propositional logic program if and only if it is noninclusive and it is a G_δ subset of some closed set in a natural inverse Scott topology.

By combining the two ideas of closure and noninclusivity, we can define a condition which guarantees that a family of sets is the set of stable models of a logic program P with strong levels. Given a family $\mathcal{S}$ of subsets of U, let $\mathcal{S}_n = \{E \cap \{u_0, \dots, u_n\} : E \in \mathcal{S}\}$.

Definition 6.4. Let $\mathcal{S}$ be a family of subsets of U.

1. We say that n is a *level* of $\mathcal{S}$ if $\mathcal{S}_n$ is mutually non-inclusive.

2. We say that $\mathcal{S}$ has levels if there are infinitely many n such that n is a level of $\mathcal{S}$.

3. We say that $\mathcal{S}$ has effective levels if there is a recursive function f such that, for all i, $f(i) > i$ and $f(i)$ is a level of $\mathcal{S}$.

There are many examples of families of set of $U = \{u_0, u_1, \dots\}$ with effective levels. For example, consider the family $\mathcal{S}$ of all sets S such that for all n, $|S \cap \{u_0, \dots, u_{2^n}\}| = n$. It is easy to see that for all n, 2^n is a level of $\mathcal{S}$. For a more general example, let U be the set of all finite truth table functions on a countably infinite set $\{a_0, a_1, \dots\}$ of propositional variables. That is, for each sentence ψ of propositional calculus, U contains exactly one sentence logically equivalent to ψ. These are listed in order of the maximum variable a_k on which the sentence depends. Thus, u_0 and u_1 are the (constant) True and False sentences; u_2 and u_3 are the sentences a_0 and $\neg a_0$, $u_4, \dots, u_{15}$ list the sentences depending on a_0 and a_1, and so on. Now let Γ be any consistent set of sentences and let $S(\Gamma)$ be the set of complete consistent models of Γ. The levels of $\mathcal{S} = \{U \cap S : S \in S(\Gamma)\}$ are just the numbers $2^{2^k} - 1$. This is because if two sets in $\mathcal{S}$ disagree on the first $2^{2^k} - 1$ sentences, then there must be some i with $i < k$ such that they disagree on a_i, which means that one of the sets contains a_i but not $\neg a_i$ whereas the other set contains $\neg a_i$ but not a_i. Thus the two sets are mutually noninclusive.

Theorem 6.5. If $\mathcal{E}$ is a closed family of subsets of U with levels, then there exists a logic program P with strong levels such that $H_P = U$ and $\mathcal{E}$ is the set of stable models of P. Furthermore, if $\mathcal{E}$ is a decidable Π^0_1 class and has effective levels, then P may be taken to have effectively strong levels.

Proof. Given the family $\mathcal{E}$, we shall directly construct a logic program P such that $\mathcal{S}(P) = \mathcal{E}$. First, if $\mathcal{E}$ is empty, we let P consist of the single clause $u_0 \leftarrow \neg u_0$. It is easy to see in this case that P has no stable models.

Thus we assume that $\mathcal{E} \neq \emptyset$ and hence that each $\mathcal{E}_n$ is nonempty as well. We then create a set of clauses for every level n of $\mathcal{E}$. For each level n, let $E^n_1, \ldots, E^n_{k_n}$ be the list of all sets of the form $E \cap \{u_0, \ldots, u_n\}$ for $E \in \mathcal{E}$. Then for each such E^n_i and each $r \in E^n_i$, P will contain a clause

$$c_{i,r} = r \leftarrow \neg\beta_1, \ldots, \neg\beta_m \qquad (8)$$

where $\{\beta_1, \ldots, \beta_m\} = \{u_0, \ldots, u_n\} - E^n_i$. It is then easy to see that each E^n_i is a stable model of P_n and that n is a level of P_n. Moreover it easily follows that the set of stable models of P is exactly $\mathcal{E}$.

For the second part of the theorem, we use the same logic program. We note that decidable Π^0-class of sets has the property that there is a highly recursive tree T contained in $\{0, 1\}^*$ such that set of infinite paths through T correspond to the characteristic functions of elements of P and T has no dead ends, i.e. every node $\eta \in T$ can be extended to an infinite path through T. Thus for any level n of $\mathcal{E}$, the sets $E^n_1, \ldots, E^n_{k_n}$ described above will just correspond to the set of nodes of length n in the tree T. Because each of the nodes of length n can be extended to infinite path through T, it follows that each stable model E^n_i of P_n can be extended to a stable model E of P such that $E \cap \{u_0, \ldots, u_n\} = E^n_i$. It then easily follows that n is a strong level of P. Thus P will have effectively strong levels in this case.

References

1. H. Andreka, I. Nemeti, The Generalized Completeness of Horn Predicate Logic as a Programming Language, Acta Cybernetica 4(1978) 3-10.
2. K. R. Apt, Logic programming, in: *Handbook of Theoretical Computer Science*, ed. J. van Leeuven, MIT Press, 1990.

3. K. Apt, H. A. Blair and A. Walker, Towards a theory of declarative knowledge, in: *Foundations of deductive databases and logic programming*, ed. J. Minker, Morgan Kaufmann, 1987, pp. 89-142.
4. K. R. Apt, H. A. Blair, Arithmetical Classification of Perfect Models of Stratified Programs, Fundamenta Informaticae 13 (1990) 1-17.
5. D. R. Bean, Effective Coloration, J. Symbolic Logic 41 (1976), 469-480.
6. D. Cenzer and J. B. Remmel, Index Sets for Π^0_1-classes, Annals of Pure and Applied Logic 93 (1998), 3-61.
7. D. Cenzer and J. B. Remmel, Π^0_1-classes in Mathematics, in: *Handbook of Recursive Mathematics: Volume 2*, eds. Yu. L. Ershov, S.S. Goncharov, A. Nerode, and J.B. Remmel, Studies in Logic and the Foundations of Mathematics, vol. 139, Elsevier, 1998, pp. 623-822.
8. D. Cenzer, J. B. Remmel, and A. K. C. S. Vanderbilt, Locally Determined Logic Programs, in: *Proceedings of the 6th international Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR99)*, Springer-Verlag, 1999, pp. 34-49.
9. P. Cholewiński, Stratified default theories, in *Proceedings of CSL'94*, Lecture Notes in Computer Science, vol. 933, Springer-Verlage, 1995.
10. A. Ferry, A topological characterization of the stable and minimal model classes of propositional logic programs, Ann. Math. Artificial Intelligence 15 (1995), 325-355.
11. M. Gelfond and V. Lifschitz, The stable semantics for logic programs, in: *Proc. 5th Int'l. Symp. Logic Programming*, MIT Press, 1988, pp. 1070-1080.
12. C. G. Jockusch and R.I. Soare, Degrees of members of Π^0_1 classes, Paciific Journal of Mathematics 40(1972), 605–616.
13. C. G. Jockusch and R. I. Soare, Π^0_1 classes and degrees of theories, Transactions of American Mathematical Society 173(1972), 33–56.
14. V. Lifschitz and H. Turner, Splitting a logic program, in: *Proceedings of the Eleventh International Conference on Logic Programming*, ed. P. Van Hentenryck, 1994, pp. 23–37.
15. W. Marek, A. Nerode, and J. B. Remmel: Nonomonotonic rule systems I, Annals of Mathematics and Artificial Intelligence 1 (1990), 241-273.
16. W. Marek, A. Nerode, and J. B. Remmel, Nonomonotonic rule systems II, Annals of Mathematics and Artificial Intelligence 5 (1992), 229-264.
17. W. Marek, A. Nerode, and J. B. Remmel, How complicated is the set of stable models of a recursive logic program? Ann. Pure and Appl. Logic 56 (1992), 119-135.
18. W. Marek, A. Nerode, and J. B. Remmel, The stable models of predicate logic programs, in: *Proceedings of International Joint Conference and Symposium on Logic Programming*, ed. K. R. Apt, editor, MIT Press, 1992, pp. 446–460.
19. W. Marek, A. Nerode, and J. B. Remmel, The stable models of predicate logic programs. Journal of Logic Programming 21 (1994), 129-154.
20. W. Marek, A. Nerode, and J. B. Remmel, Context for belief revision: Forward chaining-normal nonmonotonic rule systems, Annals of Pure and Applied Logic 67 (1994), 269-324.
21. W. Marek and J. B. Remmel, The failure of compactness in nonmonotonic reasoning systems, in preparation.
22. R. Reiter, A logic for default reasoning. Artificial Intelligence 13(1980), 81-132.
23. D. Scott, Domains for denotational semantics, in: *Proceedings of ICALP-82*, Springer-Verlag, 1982, pp. 577-613.

24. R.M. Smullyan, Theory of Formal Systems, Annals of Mathematics Studies, no. 47.

Address for Offprints: Department of Mathematics, University of Florida,
P.O. Box 118105, Gainesville, FL 32611-8105