Index sets for finite normal predicate logic programs with function symbols

D. Cenzer[1], V.W. Marek[2], and J.B. Remmel[3]

[1] Department of Mathematics, University of Florida, Gainesville, FL 32611
cenzer@ufl.edu
[2] Department of Computer Science, University of Kentucky, Lexington, KY 40506:
marek@cs.uky.edu
[3] Department of Mathematics, University of California at San Diego, La Jolla, CA 92903 jremmel@ucsd.edu

Abstract. We study the *recognition problem* in the metaprogramming of finite normal predicate logic programs. That is, let $\mathcal{L}$ be a computable first order predicate language with infinitely many constant symbols and infinitely many n-ary predicate symbols and n-ary function symbols for all $n \geq 1$. Then we can effectively list all the finite normal predicate logic programs $Q_0, Q_1, \ldots$ over $\mathcal{L}$. Given some property $\mathcal{P}$ of finite normal predicate logic programs over $\mathcal{L}$, we define the index set $I_{\mathcal{P}}$ to be the set of indices e such that Q_e has property $\mathcal{P}$. Then we shall classify the complexity of the index set $I_{\mathcal{P}}$ within the arithmetic hierarchy for various natural properties of finite predicate logic programs.

Keywords: logic programming, index sets, recursive trees

1 Introduction

Past research has demonstrated that logic programming with the stable model semantics and, more generally, answer-set semantics, is an expressive knowledge representation formalism. It can be safely stated that there is a consensus in the Knowledge Representation community that stable models are the correct generalization of the least model of Horn program for the class of normal programs. Although stable model semantics is considered the correct one, past research has shown that the use of arbitrary normal logic programs admitting function symbols is not a reasonable choice for real-life programming. For example, Apt and Blair [2] proved that all arithmetic sets can be defined by using stratified programs. The import is that in general, it is impossible to query the unique stable model of such programs. Marek, Nerode, and Remmel [17] constructed finite predicate logic programs whose stable models could code up the paths through any infinitely branching recursive tree so that the problem of deciding whether a finite predicate logic program has a stable model is Σ^1_1-complete.

For such reasons, researchers have focused on finite predicate logic programs without function symbols. There are a number of highly effective implementations of search engines to find stable models of finite normal predicate logic programs [18, 13, 11].

Nevertheless, researchers have searched for *some* natural classes $\mathcal{K}$ of finite normal predicate logic programs with function symbols where programming is both useful and possible. Actually, it should be clear that finding such a class $\mathcal{K}$ involves two tasks. (1) $\mathcal{K}$ needs to be *processable*. That is, given a program $P \in \mathcal{K}$, we need to have an algorithm that identifies one or more stable models of P which can be effectively queried. That is, one can effectively answer questions such as whether a given atom is in a given stable model of P or whether a given atom is in all stable models of P. (2) $\mathcal{K}$ needs to be *recognizable*. That is, we need to be able to answer the query whether a given program P belongs to $\mathcal{K}$. For instance, the class of stratified programs is recognizable (one of the fundamental results of Apt, Blair and Walker [3]), but not processable. A number of classes $\mathcal{K}$ of such programs which are both processable and recognizable have been found, see [21, 4, 6, 14, 5]. In particular [5] provides an extensive discussion of the reasons why researchers try to find classes of normal predicate logic programs admitting function symbols which are both recognizable and processable.

The goal of this paper is develop a systematic approach the recognition problem for the class of finite normal predicate logic programs over a computable first order predicate language $\mathcal{L}$ with infinitely many constant symbols and infinitely many n-ary predicate symbols and n-ary function symbols for all $n \geq 1$. Let $Q_0, Q_1, \ldots$ be an effective list of all the finite normal predicate logic programs over $\mathcal{L}$. Given some property $\mathcal{P}$ of finite normal predicate logic programs over $\mathcal{L}$, we define the *index set* $I_{\mathcal{P}}$ to be the set of indices e such that Q_e has property $\mathcal{P}$. For example, suppose that $\mathcal{P}$ is the property that a finite normal predicate logic program has a recursive stable model. Then the tools of this paper will allow one to classify the complexity of $I_{\mathcal{P}}$ within the arithmetic hierarchy. We will show in [8] that $I_{\mathcal{P}}$ is Σ^0_3-complete so that one can not effectively recognize the set of finite predicate logic programs which have recursive stable models.

Our approach is to extend the work of Marek, Nerode, and Remmel in [17], who showed that the problem of finding a stable of model of a recursive normal propositional logic program is essentially equivalent to finding an infinite path through an infinite recursive tree. That is, they showed that given any recursive normal propositional logic program P, one could construct a recursive tree such T_P 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_P. Vice versa, given any recursive tree T, they constructed a recursive normal propositional logic program P_T such that there is an effective one-to-one degree preserving correspondence between the set of stable models of P_T and the set of infinite paths through T. Such correspondences also helped to motivate the definition of various natural properties of normal logic programs such as having the finite support property or the recursive finite support property (described below) since these properties correspond to natural properties of recursive trees such as being finitely branching or being highly recursive. The main goal of this paper is to provide similar constructions when we replace recursive normal propositional logic programs by finite normal predicate logic programs. This requires us to significantly modify the original constructions in [17].

To define index sets for primitive recursive trees, we need some notation. Let $\Sigma \subseteq \omega$ where $\omega = \{0, 1, 2, \dots, \}$. Then $\Sigma^{<\omega}$ denotes the set of finite strings of letters from Σ and Σ^{ω} denotes the set of infinite sequences of letters from Σ. If $\sigma = (\sigma_1, \dots, \sigma_n) \in \Sigma^{<\omega}$ and $a \in \Sigma$, then we let $\sigma^\frown a = (\sigma_1, \dots, \sigma_n, a)$. A *tree* T over Σ is a set of finite strings from $\Sigma^{<\omega}$ which contains the empty string $\emptyset$ and is closed under initial segments. We say that $\tau \in T$ is an *immediate successor* of a string $\sigma \in T$ if $\tau = \sigma^\frown a$ for some $a \in \Sigma$. One can easily assign Gödel numbers to the elements of $\omega^{<\omega}$. That is, we can effectively assign a unique code $c(\sigma) \in \omega$ to each $\sigma \in \omega^{<\omega}$ such that we can effectively recover σ from $c(\sigma)$. We will identify T with the set of codes $c(\sigma)$ for $\sigma \in T$. Thus we say that T is primitive recursive, recursive, r.e., etc. if $\{c(\sigma) : \sigma \in T\}$ is primitive recursive, recursive, r.e., etc. If each node of T has finitely many immediate successors, then T is said to be *finitely branching*. We say a tree T is *highly recursive* if it is recursive and there is a recursive function f such that for any $\sigma \in T$, there are $f(\sigma)$ immediate successors of σ. An *infinite path* through a tree T is a sequence $(x(0), x(1), \dots)$ such that $(x(0), \dots x(n)) \in T$ for all n. Let $[T]$ be the set of infinite paths through T and $[T]_r$ denote the set of infinite recursive paths through T. We let $Ext(T)$ denote the set of all $\sigma \in T$ such that σ is an initial segment of x for some $x \in [T]$. We say that T is *decidable* if T is recursive and $Ext(T)$ is recursive. We let $T_0, T_1, \dots$ be an effective list of all primitive recursive trees contained in $\omega^{<\omega}$. It follows that $[T_0], [T_1], \dots$ is an effective list of all Π^0_1 classes, see [9]. Then for any property $\mathcal{P}$ of trees, we let $T_{\mathcal{P}}$ denote the set of all i such that T_i has property $\mathcal{P}$.

Our main result is to show that we can modify the constructions of Marek, Nerode, and Remmel [17] to construct recursive functions f and g such that for all e, (i) there is a one-to-one degree preserving correspondence between the set of stable models of Q_e and the set of infinite paths through $T_{f(e)}$ and (ii) there is a one-to-one degree preserving correspondence between the set of infinite paths through T_e and the set of stable models $Q_{g(e)}$. One can often use these two recursive functions to reduce the complexity of the index set $I_{\mathcal{P}}$ for various properties $\mathcal{P}$ of finite normal predicate logic programs to the complexity of the index set $T_{\mathcal{P}'}$ for an appropriate property of $\mathcal{P}'$ of primitive recursive trees. Actually, in practice, we take the reverse point of view. That is, we shall start with $T_{\mathcal{P}'}$ and try to find an appropriate property $\mathcal{P}$ of finite normal predicate logic programs such that $T_{\mathcal{P}'}$ and $I_{\mathcal{P}}$ are one-to-one equivalent.

We shall consider the following natural properties of trees contained in $\omega^{<\omega}$. Suppose that $g : \omega^{<\omega} \to \omega$. Then we say that
(I) T is **g-bounded** if for all σ and all integers i, $\sigma^\frown i \in T$ implies $i < g(\sigma)$,
(II) T is **almost always g-bounded** if there is a finite set $F \subseteq T$ of strings such that for all strings $\sigma \in T \setminus F$ and all integers i, $\sigma^\frown i \in T$ implies $i < g(\sigma)$,
(III) T is **nearly g-bounded** if there is an $n \geq 0$ such that for all strings $\sigma \in T$ with $|\sigma| \geq n$ and all integers i, $\sigma^\frown i \in T$ implies $i < g(\sigma)$,
(IV) T is **bounded** if it is g-bounded for some $g : \omega^{<\omega} \to \omega$,
(V) T is **almost always bounded** (*a.a.b.*) if it is almost always g-bounded for some $g : \omega^{<\omega} \to \omega$,

(VI) T is **nearly bounded** if it is nearly g-bounded for some $g : \omega^{<\omega} \to \omega$,
(VII) T is **recursively bounded** (*r.b.*) if T is g-bounded for some recursive $g : \omega^{<\omega} \to \omega$,
(VIII) T **almost always recursively bounded** (*a.a.r.b.*) if it is almost always g-bounded for some recursive $g : \omega^{<\omega} \to \omega$, and
(IX) T **nearly recursively bounded** (nearly *r.b.*) if it is nearly g-bounded for some recursive $g : \omega^{<\omega} \to \omega$.

For each of the properties $\mathcal{P}$ above, one can classify the index sets of the set of primitive recursive trees T satisfying property $\mathcal{P}$ and one of the following properties: $[T]$ ($[T]_r$) is empty, $[T]$ ($[T]_r$) is non-empty, $[T]$ ($[T]_r$) has cardinality c ($< c$,$\geq c$) for some natural number c, $[T]$ ($[T]_r$) is finite, or $[T]$ ($[T]_r$) is infinite.

To be able to precisely state our results, we must briefly review the basic concepts of recursion theory, normal logic programs and recursive trees.

We shall assume the reader is familiar with the basics of recursive and recursively enumerable sets, Turing degrees, and the arithmetic hierarchy of Σ^0_n and Π^0_n subsets of ω as well as Σ^1_1 and Π^1_1 sets; see Soare's book [20]. We shall generally use the terminology *recursive* rather than the equivalent term *computable* and likewise use *recursively enumerable* rather than *computably enumerable*. The former terms are standard in the logic programming community, which is an important audience for our paper. A subset A of ω is said to be D^m_n if it is the set-difference of two Σ^m_n sets. A set $A \subseteq \omega$ is said to be an *index set* if for any a, b, $a \in A$ and $\phi_a = \phi_b$ imply that $b \in A$ where $\phi_0, \phi_1, \ldots$ is an effective list of all partial recursive functions. For example, $Fin = \{a : W_a \text{ is finite}\}$ is an index set. We are particularly interested in the complexity of such index sets. Recall that a subset A of ω is said to be Σ^m_n-complete (respectively, Π^m_n-complete, D^m_n-complete) if A is Σ^m_n (respectively, Π^m_n, D^m_n) and any Σ^m_n (respectively, Π^m_n, D^m_n) set B is many-one reducible to A. For example, the set $Fin = \{e : W_e \text{ is finite}\}$ is Σ^0_2-complete.

Then, for example, Cenzer and Remmel [9] proved the following results:
(1) $\{e : T_e$ is r.b. and $[T_e]$ is empty$\}$ is Σ^0_2-complete.
(2) $\{e : T_e$ is r.b. and $[T_e]$ is nonempty$\}$ is Σ^0_3-complete.
(3) $\{e : T_e$ is bounded and $[T_e]$ is empty$\}$ is Σ^0_2-complete.
(4) $\{e : T_e$ is bounded and $[T_e]$ is nonempty$\}$ is Π^0_3-complete.
(5) $\{e : T_e$ is a.a.r.b. and $[T_e]$ is nonempty$\}$ and
$\{e : T_e$ is a.a.r.b. and $[T_e]$ is empty$\}$ are Σ^0_3-complete.
(6) $\{e : T_e$ is a.a.b. and $[T_e]$ is nonempty$\}$ and
$\{e : T_e$ is a.a.b. and $[T_e]$ is empty$\}$ are Σ^0_4-complete.
(8) $\{e : [T_e]$ is nonempty$\}$ is Σ^1_1-complete and
$\{e : [T_e]$ is empty$\}$ is Π^1_1-complete.

For any positive integer c,
(9) $\{e : T_e$ is r.b. and $Card([T_e]) > c\}$, $\{e : T_e$ is r.b. and $Card([T_e]) \leq c\}$, and $\{e : T_e$ is r.b. and $Card([T_e]) = c\}$ are all Σ^0_3-complete.
(10) $\{e : T_e$ is a.a.r.b. and $Card([T_e]) > c\}$,
$\{e : T_e$ is a.a.r.b. and $Card([T_e]) \leq c\}$, and

$\{e : T_e$ is a.a.r.b. and $Card([T_e]) = c\}$ are all Σ^0_3-complete.
(11) $\{e : T_e$ is bounded and $Card([T_e]) \leq c\}$ and
$\{e : T_e$ is bounded and $Card([T_e]) = 1\}$ are both Π^0_3-complete.
(12) $\{e : T_e$ is bounded and $Card([T_e]) > c\}$ and
$\{e : T_e$ is bounded and $Card([T_e]) = c + 1\}$ are both D^0_3-complete.
(13) $\{e : T_e$ is a.a.b. and $Card([T_e]) > c\}$,
$\{e : T_e$ is *a.a.* bounded and $Card([T_e]) \leq c\}$, and
$\{e : T_e$ is *a.a.* bounded and $Card([T_e]) = c\}$ are all Σ^0_4-complete.
(14) $\{e : T_e$ is *r.b*, decidable, and $Card([T_e]) > c\}$,
$\{e : T_e$ is *r.b.*, dec. and $Card([T_e]) \leq c\}$, and
$\{e : T_e$ is *r.b.*, dec. and $Card([T_e]) = c\}$ are all Σ^0_3-complete.
(15) $(\{e : Card([T_e]) > c\})$ is Σ^1_1-complete, $\{e : Card([T_e]) \leq c\}$ is Π^1_1-complete and $\{e : Card([T_e]) = c\}$ is Π^1_1-complete.

This is only a sample of the index set results that have been established for primitive recursive trees. For example, there are similar results when one replaces $[T_e]$ by $[T_e]_r$ in each of these statements. For each of the properties Pr in (**I**)-(**IX**) of trees, our goal is to find a corresponding property Pr' of finite normal predicate logic programs such that the complexity of the set of finite normal predicate logic programs P satisfying property Pr' refined by the cardinality of the stable models (recursive stable models) of P has the corresponding complexity of as the set of primitive recursive trees T satisfying property Pr refined by the cardinality of the set of infinite paths (recursive infinite paths) through T.

The outline of this paper is as follows. In Section 2, we shall define various properties on finite normal predicate logic programs which correspond to the properties (**I**)-(**IX**) described above. Many of the properties such as the finite support property and the recursive finite support property which correspond to bounded trees and recursively bounded trees have appeared in the literature. However, other properties such as a program being decidable, which correspond to decidable trees, are new. In Section 3, we shall state our main results. In Section 4, we state a number of results which classify the complexity of I_P for various properties P of finite normal predicate logic programs.

2 Properties of Finite Normal Logic Programs

In this section, we give the necessary background on normal logic programs.

We shall fix a recursive language $\mathcal{L}$ which has infinitely many constant symbols, infinitely many propositional letters, and infinitely many n-ary relation symbols and n-ary function symbols for each $n \geq 1$. A literal is an atomic formula or its negation. A ground literal is a literal which has no free variables. The Herbrand base of $\mathcal{L}$ is the set $H_{\mathcal{L}}$ of all ground atoms (atomic statements) of the language.

A (normal) logic programming clause C is of the form

$$c \leftarrow a_1, \ldots, a_n, \neg b_1, \ldots, \neg b_m \tag{1}$$

where $c, a_1, \ldots, a_n, b_1, \ldots, b_m$ are atoms of $\mathcal{L}$. Here we allow either n or m to be zero. In such a situation, we call c the *conclusion* of C, $a_1, \ldots, a_n$ the *premises* of C, $b_1, \ldots, b_n$ the *constraints* of C and $a_1, \ldots, a_n, \neg b_1, \ldots, \neg b_m$ the *body* of C and write $concl(C) = c$, $prem(C) = \{a_1, \ldots, a_n\}$, $constr(C) = \{b_1, \ldots, b_m\}$. A ground clause is a clause with no free variables. C is called a Horn clause if $constr(C) = \emptyset$, i.e., if C has no negated atoms in its body.

A finite normal predicate logic program P is a finite set of clauses of the form (1). P is said to be a Horn program if all its clauses are Horn clauses. A ground instance of a clause C is a clause obtained by substituting ground terms (terms without free variables) for all the free variables in C. The set of all ground instances of the program P is called $ground(P)$. The Herbrand base of P, $H(P)$, is the set of all ground atoms that are instances of atoms that appear in P. For any set S, we let 2^S denote the set of all subsets of S.

Given a Horn program P, we let $T_P : 2^{H(P)} \to 2^{H(P)}$ be the one-step provability operator [15] associated with $ground(P)$. That is, for $S \subseteq H(P)$,

$T_P(S) = \{c : \exists_{C \in ground(P)}((C = c \leftarrow a_1, \ldots, a_n) \wedge (a_1, \ldots, a_n \in S))\}$.

Then P has a least model $M = T_P \uparrow_\omega (\emptyset) = \bigcup_{n \geq 0} T_P^n(\emptyset)$ where for any $S \subseteq H(P)$, $T_P^0(S) = S$ and $T_P^{n+1}(S) = T_P(T_P^n(S))$. We denote the least model of a Horn program P by $lm(P)$.

Given a normal predicate logic program P and $M \subseteq H(P)$, we define the *Gelfond-Lifschitz reduct* of P, P_M, via the following two step process. In Step 1, we eliminate all clauses $C = p \leftarrow q_1, \ldots, q_n, \neg r_1, \ldots, \neg r_m$ of $ground(P)$ such that there exists an atom $r_i \in M$. In Step 2, for each remaining clause $C = p \leftarrow q_1, \ldots, q_n, \neg r_1, \ldots, \neg r_m$ of $ground(P)$, we replace C by the Horn clause $C = p \leftarrow q_1, \ldots, q_n$. The resulting program P_M is a Horn propositional program and, hence, has a least model. If that least model of P_M coincides with M, then M is called a *stable model* for P.

Next, we define the notion of P-proof scheme of a normal *propositional* logic program P. Given a normal propositional logic program P, a P-proof scheme is defined by induction on its length n. Specifically, the set of P-proof schemes is defined inductively by declaring that

(I) $\langle\langle C_1, p_1\rangle, U\rangle$ is a P-proof scheme of length 1 if $C_1 \in P$, $p_1 = concl(C_1)$, $prem(C_1) = \emptyset$, and $U = constr(C_1)$ and

(II) for $n > 1$, $\langle\langle C_1, p_1\rangle, \ldots, \langle C_n, p_n\rangle, U\rangle$ is a P-proof scheme of length n if $\langle\langle C_1, p_1\rangle, \ldots, \langle C_{n-1}, p_{n-1}\rangle, \bar{U}\rangle$ is a P-proof scheme of length $n-1$ and C_n is a clause in P such that $concl(C_n) = p_n$, $prem(C_n) \subseteq \{p_1, \ldots, p_{n-1}\}$ and $U = \bar{U} \cup constr(C_n)$

If $\mathbb{S} = \langle\langle C_1, p_1\rangle, \ldots, \langle C_n, p_n\rangle, U\rangle$ is a P-proof scheme of length n, then we let $supp(\mathbb{S}) = U$ and $concl(\mathbb{S}) = p_n$.

Example 1. Let P be the normal propositional logic program consisting of the following four clauses:
$C_1 = p \leftarrow$, $C_2 = q \leftarrow p, \neg r$, $C_3 = r \leftarrow \neg q$, and $C_4 = s \leftarrow \neg t$.
Then we have the following useful examples of P-proof schemes:

(a) $\langle\langle C_1, p\rangle, \emptyset\rangle$ is a P-proof scheme of length 1 with conclusion p and empty support.
(b) $\langle\langle C_1, p\rangle, \langle C_2, q\rangle, \{r\}\rangle$ is a P-proof scheme of length 2 with conclusion q and support $\{r\}$.
(c) $\langle\langle C_1, p\rangle, \langle C_3, r\rangle, \{q\}\rangle$ is a P-proof scheme of length 2 with conclusion r and support $\{q\}$.
(d) $\langle\langle C_1, p\rangle, \langle C_2, q\rangle, \langle C_3, r\rangle, \{q, r\}\rangle$ is a P-proof scheme of length 3 with conclusion r and support $\{q, r\}$.

In this example we see that the proof scheme in (c) had an unnecessary item, the first term, while in (d) the proof scheme was supported by a set containing q, one of atoms that were proved on the way to r. □

A P-proof scheme differs from the usual Hilbert-style proofs in that it carries within itself its own applicability condition. In effect, a P-proof scheme is a *conditional* proof of its conclusion. It becomes applicable when all the constraints collected in the support are satisfied. Formally, for a set M of atoms, we say that a P-proof scheme $\mathbb{S}$ is M-*applicable* or that M *admits* $\mathbb{S}$ if $M \cap supp(\mathbb{S}) = \emptyset$. The fundamental connection between proof schemes and stable models is given by the following proposition which is proved in [17].

Proposition 1. *For every normal propositional logic program P and every set M of atoms, M is a stable model of P if and only if*

(i) for every $p \in M$, there is a P-proof scheme $\mathbb{S}$ with conclusion p such that M admits $\mathbb{S}$ and
(ii) for every $p \notin M$, there is no P-proof scheme $\mathbb{S}$ with conclusion p such that M admits $\mathbb{S}$.

A P-proof scheme may not need all its clauses to prove its conclusion. It may be possible to omit some clauses and still have a proof scheme with the same conclusion. Thus we define a pre-order on P-proof schemes $\mathbb{S}$, $\mathbb{T}$ by declaring that $\mathbb{S} \prec \mathbb{T}$ if (1) $\mathbb{S}, \mathbb{T}$ have the same conclusion and (2) Every clause in $\mathbb{S}$ is also a clause of $\mathbb{T}$. The relation $\prec$ is reflexive, transitive, and well-founded. Minimal elements of $\prec$ are minimal proof schemes. A given atom may be the conclusion of no, one, finitely many, or infinitely many different minimal P-proof schemes. These differences are clearly computationally significant if one is searching for a justification of a conclusion.

If P is a finite normal predicate logic program, then we define a P-proof scheme to be a $ground(P)$-proof scheme. Since we are considering finite normal programs over our fixed recursive language $\mathcal{L}$, we can use standard Gödel numbering techniques to assign code numbers to atomic formulas, clauses, proof schemes, and programs. It is then not difficult to verify that for any given finite normal predicate logic program P, the questions of whether a given n is the code of a ground atom, a ground instance of a clause in P, or a P-proof-scheme are primitive recursive predicates. The key observation to make is that since P is finite and the usual unification algorithm is effective, we can explicitly test

whether a given number m is the code of a ground atom or a ground instance of a clause in P without doing any unbounded searches.

The set of Gödel numbers of well-formed programs is well-known to be primitive recursive (see Lloyd [15]). We let Q_e be the program with Gödel number e when this exists and let Q_e be the empty program otherwise. For any property $\mathcal{P}$ of finite normal predicate logic programs, let $I(\mathcal{P})$ be the set of indices e such that Q_e has property $\mathcal{P}$.

We say that a finite normal predicate logic program P over $\mathcal{L}$ has the *finite support (FS) property* if for every atom $a \in H(P)$, there are only finitely many inclusion-minimal supports of minimal $ground(P)$-proof schemes for a. We say that P has the *almost always finite support (a.a.FSP) property* if for all but finitely many atoms $a \in H(P)$, there are only finitely many inclusion-minimal supports of minimal $ground(P)$-proof schemes for a. We say that P has the *recursive finite support (rec.FSP) property* if it has the finite support property and there is an effective procedure which, given any atom $a \in H(P)$, produces the code of the set of the inclusion-minimal supports of $ground(P)$-proof schemes for a. We say that P has the *almost always recursive finite support (a.a.rec.FSP) property* if it has the a.a.FSP property and there is an effective procedure which, for all but a finite set of atoms $a \in H(P)$, produces the code of the set of the inclusion-minimal supports of $ground(P)$-proof schemes for a.

Next, we define two additional properties of recursive normal propositional logic programs that have not been previously defined in the literature. Suppose that P is a recursive normal propositional logic program consisting of ground clauses in $\mathcal{L}$ and M is a stable model of P. Then for any atom $p \in M$, we say that a minimal P-proof scheme $\mathbb{S}$ is the *smallest minimal P-proof for p relative to M* if $concl(\mathbb{S}) = p$ and $supp(\mathbb{S}) \cap M = \emptyset$ and there is no minimal P-proof scheme $\mathbb{S}'$ such that $concl(\mathbb{S}') = p$ and $supp(\mathbb{S}') \cap M = \emptyset$ and the Gödel number of $\mathbb{S}'$ is less than the Gödel number of $\mathbb{S}$.

We say that P is *decidable* if for all $N > 0$ and any finite (possibly empty) set of ground atoms $\{a_1, \ldots, a_n\} \subseteq H(P)$ such that the code of each a_i is less than or equal to N, and any finite set of minimal P-proof schemes $\{\mathbb{S}_1, \ldots, \mathbb{S}_n\}$ such that $concl(\mathbb{S}_i) = a_i$, we can effectively decide whether there is a stable model of M of P such that
(a) $a_i \in M$ and $\mathbb{S}_i$ is the smallest minimal P-proof scheme for a_i such that $supp(\mathbb{S}_i) \cap M = \emptyset$ and
(b) for any ground atom $b \notin \{a_1, \ldots, a_n\}$ such that the code of b is less than or equal to N, $b \notin M$.
We say that a finite normal predicate logic program is decidable if $ground(P)$ is decidable.

It will turn out that under our coding of trees into finite predicate logic programs, decidable trees induce decidable programs and under our coding of finite predicate logic programs into trees, decidable programs induce decidable trees. Moreover, decidability combined with the property of having the recursive finite support property ensures that there exists processable stable models when there are stable models. That is, we have the following theorem.

Theorem 1. *Suppose that P is a recursive normal logic program which has the recursive finite support property and is decidable. Then if P has a stable model, we can effectively find a recursive stable model of P.*

Proof. Let $a_0, a_1, \ldots$ be a list of all elements of $H(P)$ by increasing code numbers. That is, if c_i is the code of a_i, then $c_0 < c_1 < \ldots$. We will effectively construct a list of pairs of sets (A_i, R_i) for $i \geq 0$ such that for all i, $A_i \cap R_i = \emptyset$, $\{a_0, \ldots, a_i\} \subseteq A_i \cup R_i$, $A_i \subseteq \{a_0, \ldots, a_I\}$, $A_i \subseteq A_{i+1}$, and $R_i \subseteq R_{i+1}$. Then $A = \bigcup_{i \geq 0} A_i$ will be our desired recursive stable model. Thus we shall think of A_i as being the set of atoms that we have accepted to be in the stable model at stage i and R_i as being the set of atoms that have been rejected from being in A at stage i. Our construction will proceed in stages.

Stage 0. Consider a_o. Since P has the recursive finite support property, we can effectively find the supports of all the minimal P-proof schemes with conclusion a_0. If U is the support of a minimal proof scheme with conclusion a_0, then the fact that the set of minimal proofs schemes of P is r.e. means that we can search through the list of minimal proof schemes of P until we find the minimal proof scheme $\mathbb{S}_U$ with the smallest possible code such the conclusion of $\mathbb{S}_U$ is a_0 and the support of $\mathbb{S}_U$ is U. Thus if $U_1, \ldots, U_k$ is the set of all supports of minimal proof schemes with conclusion a_0, then we can effectively find proof schemes $\mathbb{S}_1, \ldots, \mathbb{S}_k$ such that for each i, $\mathbb{S}_i$ is the smallest minimal proof scheme such that the conclusion of $\mathbb{S}_i$ is a_0 and the support of $\mathbb{S}_i$ is U_i. Then, since P is decidable, we use our effective procedure with $N = c_0$ to determine whether there is a stable model M for which $\mathbb{S}_i$ is the smallest minimal proof scheme such that $supp(\mathbb{S}_i) \cap M = \emptyset$. If there is no such i, then a_0 is not in any stable model so we set $A_0 = \emptyset$ and $R_0 = \{a_0\}$. If there is such an i, then we let t_0 be the least such i and we set $A_0 = \{a_0\}$ and $R_0 = supp(\mathbb{S}_{t_0})$.

Stage $s+1$. Assume that at stage s, we have constructed A_s and R_s such that $A_s \cap R_s = \emptyset$, $\{a_0, \ldots, a_s\} \subseteq A_s \cup R_s$, $A_s \subseteq \{a_0, \ldots, a_s\}$, and for each $a \in A_s$, we have constructed a proof scheme $\mathbb{S}_a$ such that if $A_s = \{d_1, \ldots, d_k\}$ and $N_s = c_{s+1}$. Then our decision procedure associated with the decidability of P will answer yes when we give it N_S, the set $\{d_1, \ldots, d_k\}$ and the corresponding proof schemes $\mathbb{S}_{d_1}, \ldots, \mathbb{S}_{d_k}$. Moreover, we assume that

$$R_s = \{a_i : i \leq s \ \& \ a_i \notin A_s\} \cup \bigcup_{i=1}^{k} supp(\mathbb{S}_{d_i}).$$

This means that there is at least one stable model M such that for each i, $\mathbb{S}_{d_i}$ is the least proof scheme that witnesses that d_i is in M and $(\{a_0, \ldots, a_s\} - A_s) \cap M = \emptyset$.

Now consider a_{s+1}. By the fact that P has the recursive support property, we can effectively find the finite set of supports $V_1, \ldots, V_r$ of the minimal P-proof schemes of a_{s+1} and we can find P-proof schemes $\mathbb{T}_1, \ldots, \mathbb{T}_r$ such that for each $1 \leq i \leq r$, $\mathbb{T}_i$ is the smallest possible proof scheme with conclusion

a_{s+1} and support V_i. Then for each $i < r$, we can query the decision procedure associated with the decidability of P on the set $\{d_1, \ldots, d_k, a_{s+1}\}$ and the corresponding proof schemes $\mathbb{S}_{d_1}, \ldots, \mathbb{S}_{d_k}, \mathbb{T}_i$. If we get an answer yes for any i, then we let t_{s+1} be the least such i and we set $A_{s+1} = A_s \cup \{a_{s+1}\}$ and $R_{s+1} = R_s \cup supp(\mathbb{T}_{t_{s+1}})$. Note that since $\mathbb{S}_{d_1}, \ldots, \mathbb{S}_{d_k}, \mathbb{T}_i$ are the smallest minimal P proof schemes that witness that $d_1, \ldots, d_k, a_{s+1}$ are in some fixed stable model M such that $(\{a_0, \ldots, a_{s+1}\} - A_{s+1}) \cap M = \emptyset$, we must have that $A_{s+1} \cap R_{s+1} = \emptyset$. If there is no such i, then there is no stable model M which contains a_{s+1} and is such that for each i, $\mathbb{S}_{d_i}$ is the least proof scheme that witnesses that d_i is in M and $(\{a_0, \ldots, a_s\} - A_s) \cap M = \emptyset$. In that case, we let $A_{s+1} = A_s$ and $R_{s+1} = R_s \cup \{a_{s+1}\}$. It easily follows that our inductive assumption will hold at stage $s + 1$.

This completes the construction. It is easy to see that if $A = \bigcup_{s \geq 0} A_s$ and $R = \bigcup_{s \geq 0} R_s$, then $A \cap R = \emptyset$ and $\{a_0, a_1, \ldots\} \subseteq A \cup R$. Thus A and R partition $H(P)$. It is also easy to see that A is recursive since our construction is effective and at stage s, we have determined whether $a_s \in A$. We claim that A is stable model. That is, if A is not a stable model, then either there exists an a_s such that $a_s \in A$ and a_s has no P-proof scheme admitted by A or there is an $a_t \notin A$ such that a_t has an P-proof scheme which is admitted by A. Our construction ensures that if a_s is in A, then a_s has an P-proof scheme admitted by A. Thus suppose that $a_t \notin A$. Then let $W_1, \ldots, W_k$ be the supports of the minimal proof schemes of a_t. Let a_r be the largest element in $W_1 \cup \ldots \cup W_k$. Then consider what happens at stage r. Suppose $A_r = \{e_1, \ldots, e_k\}$. Then our construction also specifies minimal P-proof schemes $\mathbb{S}_1, \ldots, \mathbb{S}_k$ such that there is a stable model M such that for $1 \leq i \leq k$, $\mathbb{S}_i$ is the smallest proof scheme which witnesses that e_i is in M, $supp(\mathbb{S}_1) \cup \ldots \cup supp(\mathbb{S}_k) \subseteq R_r$, and $\{a_0, \ldots, a_r\} - A_r \cap M = \emptyset$. Thus a_t is not in M. Let $V_1, \ldots, V_b$ be the supports of the minimal P-proof schemes of a_{t+1}. This means that $M \cap supp(V_i) \neq \emptyset$ for each i. But for each i, $supp(V_i) \subseteq \{a_0, \ldots, a_r\}$ and $M \cap \{a_0, \ldots, a_r\} = A_r$. Thus it must be that case that $supp(V_i) \cap A_r \neq \emptyset$ for all i and hence a_t does not have a P-proof scheme admitted by A. Thus A is a stable model of P.

We now introduce and illustrate a technical concept that will be useful for our later considerations. At first glance, there are some obvious differences between stable models of normal propositional logic programs and models of sets of sentences in a propositional logic. For example, if T is a set of sentences in a propositional logic and $S \subseteq T$, then it is certainly the case that every model of T is a model of S. Thus a set of propositional sentences T has the property that if T has a model, then every subset of T has a model. This is not true for normal propositional logic programs. That is, suppose that P_0 is a normal propositional logic program which has a stable model and a is atom which is not in the Herbrand base of P_0, $H(P_0)$. Let P be the normal propositional logic program consisting of P_0 plus the clause $C = a \leftarrow \neg a$. Then P automatically does not have a stable model. That is, consider a potential stable model M of

P. If $a \in M$, then C does not contribute to P_M so that there will be no clause of P_M with a in the head. Hence, a is not in the least model of P_M so that M is not a stable model of P. On the other hand, if $a \notin M$, then C will contribute the clause $a \leftarrow$ to P_M so that a must be in the least model of P_M. It follows that $P_0 \cup \{a \leftarrow \neg a, a \leftarrow\}$ has a stable model but $P_0 \cup \{a \leftarrow \neg a\}$ does not.

One can see from the example above that there may be a finite set of clauses in a normal propositional or predicate logic program P which prevent P from having a stable model. Our next definition captures the key property which ensures that that the T_P which corresponds to a given finite normal predicate logic program is forced to be finite. We say that a finite normal predicate logic program Q_e over $\mathcal{L}$ has an *explicit initial blocking set* if there is an m such that

1. for every $i \leq m$, either i is not the code of an atom of $ground(P)$ or the atom a coded by i has the finite support property relative to P and there is at least one atom a in $H(P)$ whose code is less than or equal to m and
2. for all $S \subseteq \{0, \ldots, m\}$, either
   (a) there exists an $i \in S$ such that i is not the code of an atom in $H(P)$, or
   (b) there is an $i \notin S$ such that there exists a minimal P-proof scheme p such that $concl(p) = a$ where a is the atom of $H(P)$ with code i and $supp(p) \subseteq \{0, \ldots, m\} - S$, or
   (c) there is an $i \in S$ such that every minimal P-proof scheme $\mathbb{S}$ of the atom a of $H(P)$ with code i has $supp(\mathbb{S}) \cap S \neq \emptyset$.

The definition of a finite normal predicate logic program Q_e over $\mathcal{L}$ having an *initial blocking set* is the same as Q_e having an explicit initial blocking set, except that we drop the condition that for every $i \leq m$ which is the code of an atom $a \in H(P)$, a must have the finite support property relative to P.

3 Main Results

Next we state the main results which reduce the problem of computing index sets for finite normal predicate logic programs to the problem of computing index sets for primitive recursive trees. We shall only give a sketch of the proofs of our main results. The full proofs are long and technical and can be found in [8].

Theorem 2. *There is a uniform effective procedure which given any recursive tree $T \subseteq \omega^{<\omega}$ produces a finite normal predicate logic program P_T such that the following hold.*

1. *There is an effective one-to-one degree preserving correspondence between the set of stable models of P_T and the set of infinite paths through T.*
2. *T is bounded if and only if P_T has the FS property.*
3. *T is recursively bounded if and only if P_T has the rec.FS property.*
4. *T is decidable and recursively bounded if and only if P_T is decidable and has the rec.FS property.*

Proof Sketch. Let T be a recursive tree contained in $\omega^{<\omega}$. Note that the empty sequence, whose code is 0, is in T. Below we shall only describe the program P_T. The details that P_T has the desired properties can be found in [8].

A classical result, first explicit in [23] and [1], but known earlier in equational form, is that every r.e. relation can be computed by a suitably chosen predicate over the least model of a finite predicate logic Horn program. An elegant method of proof due to Shepherdson [22] uses the representation of recursive functions by means of finite register machines. When such machines are represented by Horn programs in the natural way, we get programs in which every atom can be proved in only finitely many ways; see also [19]. Thus we have the following.

Proposition 2. *Let $r(\cdot,\cdot)$ be a recursive relation. Then there is a finite predicate logic program P_r computing $r(\cdot,\cdot)$ such that every atom in the least model M_r of P_r has only finitely many minimal proof schemes and there is a recursive procedure such that given an atom a in Herbrand base of P_r produces the code of the set of P_r-proof schemes for a. Moreover, the least model of P_r is recursive.*

It follows that there exists the following three normal finite predicate logic programs such that the set of ground terms in their underlying language are all of the form 0 or $s^n(0)$ for $n \geq 1$ where 0 is a constant symbol and s is a unary function symbol. We shall use n as an abbreviation for the term $s^n(0)$ for $n \geq 1$.

(I) There is a finite predicate logic Horn program P_0 such that for a predicate *tree*$(\cdot)$ of the language of P_0, the atom *tree*(n) belongs to the least Herbrand model of P_0 if and only if n is a code for a finite sequence σ and $\sigma \in T$.

(II) There is a finite predicate logic Horn program P_1 such that for a predicate *seq*$(\cdot)$ of the language of P_1, the atom *seq*(n) belongs to the least Herbrand model of P_1 if and only if n is the code of a finite sequence $\alpha \in \omega^{<\omega}$.

(III) There is a finite predicate logic Horn program P_2 which correctly computes the following recursive predicates on codes of sequences.

- **(a)** *samelength*$(\cdot,\cdot)$. This succeeds if and only if both arguments are the codes of sequences of the same length.
- **(b)** *diff*$(\cdot,\cdot)$. This succeeds if and only if the arguments are codes of sequences which are different.
- **(c)** *shorter*$(\cdot,\cdot)$. This succeeds if and only both arguments are codes of sequences and the first sequence is shorter than the second sequence.
- **(d)** *length*$(\cdot,\cdot)$. This succeeds when the first argument is a code of a sequence and the second argument is the length of that sequence.
- **(e)** *notincluded*$(\cdot,\cdot)$. This succeeds if and only if both arguments are codes of sequences and the first sequence is not an initial segment of the second.
- **(f)** *num*$(\cdot)$. This succeeds if and only if the argument is either 0 or $s^n(0)$ for some $n \geq 1$.

Now let P^- be the finite predicate logic program $P_0 \cup P_1 \cup P_2$. We denote its language by $\mathcal{L}^-$ and we let M^- be the least model of P^-. By Proposition 2, P^- is a Horn program, M^- is recursive, and for each ground atom a in the Herbrand base of P^-, we can explicitly construct the set of all P^--proof schemes of a. In particular, *tree*$(n) \in M^-$ if and only if n is the code of node in T.

Our final program P_T will consist of P^- plus clauses (1)-(7) given below. We assume no predicate that appears in the head of any of these clauses is in the language $\mathcal{L}^-$. However, we do allow predicates from P^- to appear in the body of clauses (1) to (7). It follows that for any stable model of the extended program, its intersection with the set of ground atoms of $\mathcal{L}^-$ will be M^-. In particular, the meaning of the predicates listed above will always be the same. We can now write the additional clauses which, together with P^-, will form the desired program P_T. First of all, we select three new unary predicates:

(i) $path(\cdot)$, whose intended interpretation in any given stable model M of P_T is that it holds only on the set of codes of sequences that lie on infinite path through T. This path will correspond to the path encoded by the stable model of M,
(ii) $notpath(\cdot)$, whose intended interpretation in any stable model M of P_T is the set of all codes of sequences which are in T but do not satisfy $path(\cdot)$, and
(iii) $control(\cdot)$, which will be used to ensure that $path(\cdot)$ always encodes an infinite path through T.

This given, the final 7 clauses of our program are the following.

(1) $path(X) \longleftarrow tree(X),\ \neg notpath(X)$
(2) $notpath(X) \longleftarrow tree(X),\ \neg path(X)$
(3) $path(0) \longleftarrow$
(4) $notpath(X) \longleftarrow tree(X),\ path(Y), tree(Y), samelength(X,Y), diff(X,Y)$
(5) $notpath(X) \longleftarrow tree(X),\ tree(Y),\ path(Y),\ shorter(Y,X), notincluded(Y,X)$
(6) $control(X) \longleftarrow path(Y),\ length(Y,X)$
(7) $control(X) \longleftarrow \neg control(X), num(X)$

Clearly, $P_T = P^- \cup \{(1), \ldots, (7)\}$ is a finite program.

Theorem 3. *There is a uniform recursive procedure which given any finite normal predicate logic program P produces a primitive recursive tree T_P such that the following hold.*

1. *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_P.*
2. *P has the FS property or P has an explicit initial blocking set if and only if T_P is bounded.*
3. *If P has a stable model, then P has the FS property if and only if T_P is bounded.*
4. *P has the rec.FS property or an explicit initial blocking set if and only if T_P is recursively bounded.*
5. *If P has a stable model, then P has the rec.FS property if and only if T_P is recursively bounded.*
6. *P has the a.a.FS property or P has an explicit initial blocking set if and only if T_P is nearly bounded.*
7. *If P has a stable model, then P has the a.a.FS property if and only if T_P is nearly bounded.*

8. *P has the a.a.rec.FS property or an explicit initial blocking set if and only if T_P is nearly recursively bounded.*
9. *If P has a stable model, then P has the a.a.rec.FS property if and only if T_P is nearly recursively bounded.*
10. *If P has a stable model, then P is decidable if and only if T_P is decidable.*

Proof Sketch.

Our basic strategy is to encode a stable model M of $ground(P)$ by a path $f_M = (f_0, f_1, \ldots)$ through the complete ω-branching tree $\omega^{<\omega}$ as follows.

1. First, for all $i \geq 0$, $f_{2i} = \chi_M(i)$. That is, at the stage $2i$, we encode the information about whether or not the atom encoded by i belongs to M. Thus, in particular, if i is not the code of ground atom in $H(P)$, then $f_{2i} = 0$.
2. If $f_{2i} = 0$, then we set $f_{2i+1} = 0$. But if $f_{2i} = 1$ so that $i \in M$ and i is the code of a ground atom in $H(P)$, then we let f_{2i+1} equal $q_M(i)$ where $q_M(i)$ is the least code for a minimal P-proof scheme $\mathbb{S}$ for i such that the support of $\mathbb{S}$ is disjoint from M. That is, we select a minimal P-proof scheme $\mathbb{S}$ for i, or to be precise for the atom encoded by i, such that $\mathbb{S}$ has the smallest possible code of any P-proof scheme $\mathbb{T}$ such that $supp(\mathbb{T}) \cap M = \emptyset$. If M is a stable model, then, by Proposition 1, at least one such P-proof scheme exists for i.

Clearly, $M \leq_T f_M$ since it is enough to look at the values of f_M at even places 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 $f_M \leq_T M$. Thus the correspondence $M \mapsto f_M$ is an effective degree-preserving correspondence.

Then, given a program P, we construct a primitive recursive tree $T_P \subseteq \omega^\omega$ such that $[T_P] = \{f_M : M \in stab(P)\}$. The details of this construction and the verification that it has the desired properties can be found in [8].

4 Corollaries, Conclusions and Further Work

Theorems 2 and 3 allow us to transfer many results about paths through recursive trees to stable models of finite normal predicate logic programs. We give a brief sample of some the results that we have proved in this manner.

(1) $\{e : Q_e$ has the rec.FSP$\}$ is Σ^0_3-complete.

This can be proved as follows. First it is a straightforward exercise to show that the property that Q_e has the rec.FSP is a Σ^0_3 predicate. Then Cenzer and Remmel proved that $\{e : T_e$ is rec. bounded$\}$ is Σ^0_3-complete. Next, it follows from Theorem 3.1 that $\{e : T_e$ is rec. bounded$\}$ is one-to-one reducible to the set $\{e : Q_e$ is rec.FSP$\}$. Hence the set of $\{e : Q_e$ is rec.FSP$\}$ is Σ^0_3-complete.

The following results can be proved by similar type reasoning.

(2) $\{e : Q_e$ has the FSP$\}$ is Π^0_3-complete.
(3) $\{e : Q_e$ has the rec.FSP and is dec.$\}$ is Σ^0_3-complete.
(4) $\{e : Q_e$ has the rec.FSP and $Stab(Q_e) \neq \emptyset\}$ is Σ^0_3-complete.
(5) $\{e : Q_e$ has the FSP and $Stab(Q_e) \neq \emptyset\}$ is Π^0_3-complete.
(6) $\{e : Stab(Q_e) \neq \emptyset\}$ is Σ^1_1-complete.

For any positive integer c,

(7) $\{e : Q_e$ has the rec.FSP and $Card(Stab(Q_e)) > c\}$,
$\{e : Q_e$ has the rec.FSP and $Card(Stab(Q_e)) \leq c\}$,
and $\{e : Q_e$ has the rec.FSP and $Card(Stab(Q_e)) = c\}$ are all Σ^0_3-complete.
(8) $\{e : Q_e$ has the FSP and $Card(Stab(Q_e)) \leq c\}$ and
$\{e : Q_e$ has the FSP and $Card(Stab(Q_e)) = 1\}$ are both Π^0_3-complete.
(9) $\{e : Q_e$ has the FSP and $Card(Stab(Q_e)) > c\}$ and
$\{e : Q_e$ is has the FSP and $Card(Stab(Q_e)) = c+1\}$ are both D^0_3-complete.
(10) $\{e : Q_e$ has the rec.FSP and is dec. and $Card(Stab(Q_e)) > c\}$,
$\{e : Q_e$ has the rec.FSP and is dec. and $Card(Stab(Q_e)) \leq c\}$,
and $\{e : Q_e$ has the rec.FSP and is dec. and $Card(Stab(Q_e)) = c\}$ are all Σ^0_3-complete.
(11) $\{e : Card(Stab(Q_e)) > c\}$ is Σ^1_1-complete and
$\{e : Card(Stab(Q_e)) \leq c\}$ and $\{e : Card(Stab(Q_e)) = c\}$ are Π^1_1-complete.

In [8], we proved many more results of this type. The properties that we considered involve both whether a finite normal predicate logic program possesses a blocking set or has various properties related to finite support property as well has properties about the types and complexity of its stable models such the cardinality of its set of stable models or the cardinality of its set of recursive stable models. This required that we prove some new index type results for trees and to modify the constructions of Theorems 2 and 3.

We believe that the types of relationship established in this paper between the sets of stable models of a finite normal predicate logic programs and the sets of infinite paths through recursive trees is a technology which can be applied to study the complexity of other notions that have appeared in the Answer Set Programming literature. For example, Cenzer and Remmel [10] showed that there is an intimate connection between the well-founded semantics of logic programs and Cantor-Bendixson derivatives. One could also ask whether our correspondences can be extended to handle cases where one adds additional constructs to logic programs such as aggregates, and more generally non-monotone set-constraints.

References

1. Andreka, H., Nemeti, I.: The Generalized Completeness of Horn Predicate Logic as a Programming Language. Acta Cybernetica 4, 3–10 (1978)
2. Apt, K.R., Blair, H.A.: Arithmetic Classification of Perfect Models of Stratified Programs. Fund. Inf 14, 339–343 (1991)

3. Apt, K.R., Blair, H. A., Walker, A.: Towards a Theory of Declarative Knowledge. In: Foundations of Deductive Databases and Logic Programming, pp. 89–148 (1988)
4. Bonatti, P.: Reasoning with infinite stable models. Art. Int. Journal 156, 75–111 (2004)
5. Calautti, M., Greco, S., Spezzano, F., I. Trubitsyna, I.: Checking Termination of Bottom-Up Evaluation of Logic Programs with Function Symbols. Theory and Practice of Logic Programming 15, 854–859 (2015)
6. Calimeri, F., Cozza, S., Ianni, G., Leone, N.: Computable functions in ASP. Theory and Implementation. In: Proceedings of ICLP 2008, pp. 407–424 (2008).
7. Cenzer, D., Marek, V.W., Remmel, J.B.: Index sets for finite predicate logic programs. In: T. Eiter,T., Gottlob, G. (eds.), FLOC '99 Workshop on Complexity-theoretic and Recursion-theoretic methods in Databases, Artificial Intelligence and Finite Model Theory, pp. 72–80 (1999)
8. Cenzer, D., Marek, V.W., Remmel, J.B.: Index sets for finite predicate logic programs, In preparation (2016)
9. Cenzer, D., Remmel, J.B.: Index Sets for Π^0_1 classes. Ann. Pure Appl. Logic 93, 3–61 (1998)
10. Cenzer, D., Remmel, J.B.: A connection between Cantor-Bendixson derivatives and the well-founded semantics of finite logic programs. Ann. of Math. and Art. Int. 65 , 1–24 (2012)
11. Gebser, M., Kaufmann, B., Neumann, A., T. Schaub, T. : Conflict-Driven Answer Set Solving, In: M. Veloso (ed.), Proceedings of Joint International Conference on Artificial Intelligence , p. 386 (2007)
12. Gelfond, M., Lifschitz, V.: The stable semantics for logic programs. In: Logic Programming, Proc. of the 5th International Symposium, pp. 1070–1080. MIT Press (1998)
13. Leone, N., Pfeifer, G., Faber, W., Eiter, T.,Gottlob, G., Perri, S., Scarcello, F.: The dlv system for knowledge representation and reasoning. ACM Trans. Comput. Logic 7, 499–562 (2006)
14. Lierer, Y., Lifschitz, V.: One more decidable class of finitely ground programs. In: Proceedings of ICLP 09, pp. 489–493 (2009)
15. Lloyd, J.: Foundations of Logic Programming. Springer-Verlag (1989)
16. Marek, V.W., Nerode, A., Remmel, J.B.: How complicated is the set of stable models of a recursive logic program. Ann. Pure and App. Logic 56, 119–136 (1992)
17. Marek, V.W., Nerode, A., Remmel, J.B.: The Stable Models of Predicate Logic Programs. J. Logic Program 21, 129–153 (1994)
18. Niemelä, I., Simons, P.: Smodels — an implementation of the stable model and well-founded semantics for normal logic programs. In: Dix, J., U. Furbach, U., Nerode, A. (eds.) Logic Programming and Nonmonotonic Reasoning, Proceedings of the 4th International Conference, volu LNCS, vol. 1265, pp. 420–429. Springer, Heidelberg (1997)
19. Nerode, A., Shore, R.: Logic for Applications. Springer-Verlag (1993)
20. Soare, R.: Recursively Enumerable Sets and Degrees. Springer-Verlag (1987)
21. Syrjänen, T.: Omega-restricted logic programs. In: Proc. LPNMR 2001, pp. 267–279 (2001)
22. Shepherdson, J.C.: Unsolvable Problems for SLDNF-resolution. Journal of Logic Programming 10, 19–22 (1991)
23. Smullyan, R.M.: First-order Logic. Springer-Verlag (1968)