# Logic Programming with Infinite Sets

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

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

Victor W. Marek (marek@cs.uky.edu)
*University of Kentucky*

December 10, 2004

**Abstract.** Using the ideas from current investigations in Knowledge Representation we study the use of a class of logic programs for reasoning about infinite sets. Our programs reason about the codes for various infinite sets. Depending on the form of atoms allowed in the bodies of clauses we obtain a variety of completeness results for various classes of arithmetic sets of integers.

**Keywords:** nonmonotonic logic, logic programming

**68T27:** 03B70

## 1. Introduction

The motivation for this paper comes out of recent progress in logical foundations of Artificial Intelligence. In particular, it is concerned with recent advances in the area of Knowledge Representation. In the past few years, there has been significant progress in the theory and practice of Logic Programming. In particular, a whole new area called *Answer Set Programming* (ASP) [4] has arisen which can be viewed as a fusion of Logic Programming with Stable Model Semantics (SLP) and satisfiability (SAT). Answer Set Programming has emerged as both a theoretical and practical basis for the development of new generation of systems that are solidly grounded in the theory of Computer Science and capable of handling practical search problems arising in applications. The current generation of ASP systems such as *smodels*, `dlv`, *cmodels* and ASSAT [34, 39, 20, 3, 28], which uses both the native techniques of Logic Programming and the technology developed in SAT [32, 23], carry a lot of promise. Moreover, new types of constraints are introduced that allow for a more compact representation of problems [36, 22]. In such systems, the task of the programmer becomes easier because of the effort spent by the back-end processing engines.

The main purpose of this paper is to develop some extensions of the current ASP formalism that allows one to reason about *infinite*

sets. Since one cannot directly represent infinite sets, the key idea is to use recursion-theoretic techniques to reason about various types of indices (i.e. codes) of finite, recursive and recursively enumerable (r.e.) sets. In particular, we develop a new extension of Logic Programming, called Extended Set Based (ESB) Logic Programming, which allows constraints expressed in terms of such indices.

These new types of constraints are introduced below. However, we first recall the basic definitions of answer set programming and some of its recent extensions such as logic programming with weight and cardinality constraints [34] and set constraint logic programming [30].

A logic programming clause is a construct of the form

$$C = p \leftarrow q_1, \ldots, q_m, not\ r_1, \ldots, not\ r_n$$

where $p, q_1, \ldots, q_m, r_1, \ldots, r_n$ are atoms. A logic program is a set of logic programming clauses. The atoms $q_1, \ldots, q_m, not\ r_1, \ldots, not\ r_n$ form the *body* of $C$ and the atom $p$ is its *head*. We say that a set of atoms $M$ *satisfies* an atom $a$ if $a \in M$ and that $M$ satisfies *not* $a$ if $a \notin M$. We say that a set of atoms $M$ is a model of a clause $C$ if either $M$ does not satisfy the body of $C$ or $M$ satisfies the head of $C$. Finally we say that $M$ is a model of a program $P$ if $M$ is model of all the clauses in $P$.

The clauses $C$ where $n = 0$ are called *Horn* clauses. A program entirely composed of Horn clauses is called a Horn program and a Horn program always has a least model. It is the intended semantics of such a program. For programs with bodies containing the negation operator *not*, we will use the stable model semantics. Following [21], we define a *stable model* of the program as follows. Assume $M$ is a collection of atoms. The *Gelfond-Lifschitz reduct* of $P$ by $M$ is a Horn program arising from $P$ by first eliminating those clauses in $P$ which contain *not* $r$ with $r \in M$. In the remaining clauses, we drop all negative literals from the body. The resulting program $GL_M(P)$ is a Horn program. We call $M$ a stable model of $P$ if $M$ is the least model of $GL_M(P)$. Thus there are four steps are involved in computation of a stable model.

**Step 1.** Guess a candidate set of atoms $M$.

**Step 2.** Compute the reduced program $GL_M(P)$.

**Step 3.** Compute the least model $N$ of $GL_M(P)$.

**Step 4.** Test if $M$ coincides with $N$.

We note that for a Horn program $P$, there is a unique stable model, namely, the least model of $P$.

It is the general consensus of the Knowledge Representation community that stable models are the intended models of logic programs. Once such a consensus emerged, it was natural for both theoreticians and logicians to study various complexity issues associated with stable models of logic programs. There has been extensive effort of the community to investigate both the theoretical issues associated with stable models and the practical algorithms for processing.

The stable models of logic programs that contain function symbols can be quite complex. Recall that the least model of a Horn program $P$ is recursively enumerable in $P$. Thus the least model of a recursive Horn program is recursively enumerable. Starting with [2] and continuing with [5] and [29], a number of results showed that the stable models of logic programs that allow function symbols can be exceedingly complex, even in the case where the program has a unique stable model. For example, Marek, Nerode and Remmel [29] showed that there exist finite predicate logic programs which have stable models but which have no hyperarithmetic stable model. While these results may at first glance appear negative, they had a positive effect in the long run since they forced researchers and designers to limit themselves to cases where programs can be actually processed. The effect was that processing programs (called *solvers*) tended to focus on finite programs that do not admit function symbols.

The designers of the solvers have also focused on the issues of both improving processing of the logic programs (i.e. searching for a stable model) and improving the use of logic programs as a programming language. The latter task consists of extending the constructs available to the programmer to make programming easier and more readable. For example, various researchers discovered that it was possible to introduce meaningful extensions to the logic programming syntax and yet have such extensions be processed in a manner which is entirely analogous to the processing currently employed in case of logic programs proper. We shall briefly describe two such extensions which are particularly relevant to this paper, namely *logic programming with weight and cardinality constraints*(CC-logic programs) [36, 35] and *set constraint programming* (SC-logic programs) [30].

Let $\omega$ denote the set $\{0, 1, 2, \ldots\}$ of natural numbers. Then a *cardinality constraint atom* (CC-atom) is a constraint of the form $kXl$ where $X$ is a finite set of atoms and $k$ and $l$ are elements of $\omega \cup \{\infty\}$ such that $k \leq |X|$ and $k \leq l$. The *meaning* of such an atom is that a putative model $M$ satisfies $kXl$, written $M \models kXl$, if and only if $k \leq |M \cap X| \leq l$. CC-atoms are special cases of more general *weight constraint atoms*. That is, suppose that we have some weight function $wt$ on the set of literals over $X$, then we say that a model $M$ satisfies

a *weight constraint* $l \leq X \leq u$ if and only if

$$l \leq [\sum_{p \in M \cap X} wt(p) + \sum_{p \in X - M} wt(not\ p)] \leq u.$$

Both CC-atoms and weight constraint atoms are special case of *set constraint atoms* introduced in [30]. A set constraint atom (SC-atom) has the form $\langle X, \mathcal{F} \rangle$ where $\mathcal{F} \subseteq 2^X$. Here we say that a set of atoms $M$ satisfies $\langle X, \mathcal{F} \rangle$ if and only if $M \cap X \in \mathcal{F}$. Some specific set constraints were investigated in [22]. Many types of constraints can be expressed in the form $\langle X, \mathcal{F} \rangle$.

**Example 1.1.** (**Cardinality and Weight Constraint Atoms**) A CC-atom $kXl$ can be expressed as the SC-atom $\langle X, \mathcal{F}_{k,l} \rangle$ where $\mathcal{F}_{k,l} = \{Y \subseteq X : k \leq |Y| \leq l\}$. Similarly if we have a weight function $wt$ on literals, the more general weight constraint $kXl$ considered [35] where a model $M$ satisfies $kXl$ if and only if

$$k \leq [\sum_{a \in X \cap M} wt(a)) + \sum_{b \in X - M} wt(not\ \ b)] \leq l$$

can be expressed as the SC-atom $\langle X, \mathcal{F} \rangle$ where

$$\mathcal{F} = \{Y \subseteq X : k \leq [\sum_{a \in Y} wt(a)) + \sum_{b \in X - Y} wt(not\ b)] \leq l\}.$$

□

**Example 1.2.** (**SQL Aggregate Atoms**) Let $X$ be a finite set of atoms and let $\mu : X \to \mathrm{R}$ be a function. Each such function $\mu$ allows us to construct a variety of set constraint atoms. For example, to each $Y \subseteq X$, we can assign the following functions that are used in SQL queries: $|Y|$, $sum(Y) = \sum_{y \in Y} \mu(y)$, $\min(Y) = \min_{y \in Y} \mu(y)$, $\max(Y) = \max_{y \in Y} \mu(y)$, $avg(Y)$, where $avg$ assigns to $Y$ the real number 0 if $Y = \emptyset$ and assigns the real number $\frac{sum(Y)}{|Y|}$, otherwise. For every two real numbers $a, b$ such that $a \leq b$, we define the following families of sets:

1. $C_X^{a,b} = \{Y : a \leq |Y| \leq b\}$
2. $S_X^{a,b} = \{Y : a \leq sum(Y) \leq b\}$
3. $Max_X^{a,b} = \{Y : a \leq \max(Y) \leq b\}$
4. $Min_X^{a,b} = \{Y : a \leq \min(Y) \leq b\}$
5. $avg_X^{a,b} = \{Y : a \leq avg(Y) \leq b\}$

For each family $\mathcal{F}$ described in (1)-(5), we obtain a set constraint $\langle X, \mathcal{F}\rangle$. □

**Example 1.3.** (**Programs with External Modules**) In [19], Eiter, Gottlob and Veith studied logic programs whose clauses contain *modules* in their bodies. Modules are programs $\pi$ (written in some fixed programming language) that return subsets of some finite set of atoms $X$. Let us define $R_\pi$ as the set of those subsets of $X$ that can be returned by $\pi$. Eiter, Gottlob and Veith show how a stable semantics can be assigned to programs that contain atoms of the form $\langle X, R_\pi\rangle$ in the body of clauses. The construction of SC-stable models due to Marek and Remmel [30] described below extends the work of [19] in that SC-logic programming allows modules to occur both in the heads and in the bodies of clauses. □

There are other families of subsets of a set that could be of interest.

**Example 1.4.** Given a finite set of atoms, let $\mathcal{F}_{even} = \{Y \subseteq X : |Y| \textit{ is even}\}$ and $\mathcal{F}_{odd} = \{Y \subseteq X : |Y| \textit{ is odd}\}$. Then $\langle X, \mathcal{F}_{even}\rangle$ and $\langle X, \mathcal{F}_{odd}\rangle$ are set constraint atoms. □

Other examples of constraints of interest can be found in [22].

Formally a set-constraint clause (or SC-clause) is an expression of the form

$$\langle X, \mathcal{F}\rangle \leftarrow \langle X_1, \mathcal{F}_1\rangle, \ldots, \langle X_m, \mathcal{F}_m\rangle$$

A set-constraint (SC) program is a collection of SC-clauses. It is easy to see that ordinary logic programming can be reduced to set-constraint programming. That is, the meaning of atom $a$ is the same as that of set constraint $\langle\{a\}, \{\{a\}\}\rangle$ and the meaning of *not* $a$ is the same as $\langle\{a\}, \{\emptyset\}\rangle$. Our definition of stable model is an extension of the version of the Gelfond-Lifschitz transform introduced by Niemelä, Simons and Soininen [34] for cardinality constraint programs and we call it the *NSS transform*. Again the process of constructing a stable model is based on some form of "Horn" programs, reduction, and least fixed points of the one-step provability operators for Horn programs. First, we say that a family $\mathcal{F}$ of subsets of $X$ is *upper closed* if $Y \subseteq Z \subseteq X$ and $Y \in \mathcal{F}$ implies $Z \in \mathcal{F}$. We will call a SC-clause *Horn* if

1. the head of that clause is a single atom (recall that atoms are represented as set constraints) and
2. whenever $\langle X_i, \mathcal{F}_i\rangle$ appears in the body, then $\mathcal{F}_i$ is an upper closed family of subsets of $X_i$.

A set-constraint Horn program $P$ is a SC-program which consist entirely of Horn clauses. There is a natural one-step provability operator

associated to a SC-Horn program $P$, $T_P : 2^X \to 2^X$ where $X$ is the underlying set of atoms of the program, defined by $T_P(S)$ equals the set of all $p$ such that there is clause

$$C = p \leftarrow \langle X_1, \mathcal{F}_1\rangle, \ldots, \langle X_m, \mathcal{F}_m\rangle \in P$$

such that $S$ satisfies the body of $C$. Our definitions ensure that $T_P$ is a monotone operator and hence each SC-Horn program $P$ has a least model $M^P$. $M^P$ can be computed in a manner analogous to the computation of the least model of a definite Horn program as $T_P \uparrow^\omega (\emptyset)$. The NSS transform $\mathbf{NSS}_M(P)$ of the set-constraint program $P$ for a given set of atoms $M$ is defined as follows. First eliminate all clauses with bodies not satisfied by $M$. Next, for each remaining clause $\langle X, \mathcal{F}\rangle \leftarrow \langle X_1, \mathcal{F}_1\rangle, \ldots, \langle X_m, \mathcal{F}_m\rangle$ and each $p \in M \cap X$, put the clause

$$p \leftarrow \langle X_1, \overline{\mathcal{F}}_1\rangle, \ldots, \langle X_m, \overline{\mathcal{F}}_m\rangle$$

into $\mathbf{NSS}_M(P)$. Here $\overline{\mathcal{F}_i}$ is the least family $\mathcal{G}$ containing $\mathcal{F}_i$ and closed upwards. Clearly the resulting program $\mathbf{NSS}_M(P)$ is a SC-Horn program and hence has a least model $M^{\mathbf{NSS}_M(P)}$. $M$ is a stable model of $P$ if $M$ is a model of $P$ and $M = M^{\mathbf{NSS}_M(P)}$. It can be shown [34] that this construction corresponds to the same notion of Gelfond-Lifschitz stable models when we restrict ourselves to ordinary logic programs.

In this paper we would like to use the mechanism of set-constraints to reason about *infinite* sets. The motivation for reasoning about infinite sets comes from several areas. For example, one may want to reason about regions in 2-dimensional space, i.e. polygons, circles, or in 3-dimensional space, i.e. cubes, spheres, etc. In such a situations, one can specify the region by certain sets of equations. However one might be interested in the spread of certain diseases or the spread of pests such as Africanized bees. In that case, one might want to talk about regions whose boundaries are specified in a more dynamic manner, i.e. regions whose boundaries are specified as the solutions to ordinary or partial differential equations or which evolve according some dynamical system. In the most general case, one can specify some sort of *algorithm* which determines which points are in the region or which enumerates the points in the region. Another example is where one would like a logic program to reason about the stable models of other logic programs. In this case, even if the other logic programs were Horn predicate logic programs, the resulting least models might be infinite r.e. sets. In each of the cases above, one has no intention of writing out the entire infinite set. Instead, one use equations, differential equations, programs and/or algorithms to *encode* the infinite sets by *finite* means. That is, one uses various types of *indices* or codes to specify the finite sets. As we shall see

later, if one requires that such codes or indices have a certain number of effective properties one will still be able to process programs in effective manner, see Theorem 2.5. Examples of such effective properties include algorithms which, given the indices $e$ and $f$ of two infinite sets $A$ and $B$, would allow one to find the index of $A \cup B$ or to decide if $A \subseteq B$ or $A \cap B = \emptyset$.

Our basic idea is as follows. Assume that we have a particular coding scheme for some family of subsets of a set $X$. Let $\mathcal{F}$ be a finite family of such codes. We will write $C_e$ for the set with the code $e$. Then we can write two types of constraints. One constraint $\langle X, \mathcal{F}\rangle^{\subseteq}$ has the meaning that the putative set of integers $M$ satisfies $\langle X, \mathcal{F}\rangle^{\subseteq}$ if and only if $M \cap X \supseteq C_e$ for some $e \in \mathcal{F}$. Similarly, we shall also consider constraints of the form $\langle X, \mathcal{F}\rangle^{=}$ where we say that $M$ satisfies $\langle X, \mathcal{F}\rangle^{=}$ if and only if $M \cap X = C_e$ for some $e \in \mathcal{F}$. Observe that constraints of the form $\langle X, \mathcal{F}\rangle^{\subseteq}$ behave like atoms $p$ in that they are preserved when the set grows while constraints of the form $\langle X, \mathcal{F}\rangle^{=}$ behave like constraints *not* $p$ in that they are not always preserved as the set grows.

Now, it is clear that once we introduce these type of constraint schemes, we can consider various coding schemes for the set of indices. For example, in this paper, we will consider three such schemes: explicit indices of finite sets, recursive indices of recursive sets and r.e. indices of r.e. sets. We shall then define an extended set-based clause $C$ to be a clause of the form

$$\langle X, \mathcal{A}\rangle^{*} \leftarrow \langle Y_1, \mathcal{B}_1\rangle^{\subseteq}, \ldots, \langle Y_k, \mathcal{B}_k\rangle^{\subseteq}, \langle Z_1, \mathcal{C}_1\rangle^{=}, \ldots, \langle Z_l, \mathcal{C}_l\rangle^{=},$$

where $*$ is either $=$ or $\subseteq$ and define an *extended set based program* (ESB) $P$ to be a set of extended set based clauses.

We should note that our work here is closely related to a number of other approaches to extend logic programming. Constraint Logic Programming [25] allows one to use other modules to reason about various specialized types of sets and relations. The work of Dovier, Omodeo, Pontelli and Rossi [9, 16, 17, 18] describe various mechanisms which extend logic programming and allow one to reason about finite sets. The work of Eiter, Gottlob and Veith [19] who studied logic programs whose clauses contain *modules* in their bodies is another example of such an extension. In all of these extensions, the goal is to reason about finite sets so that the various types of additional construction involve only finite sets and relations. Since the use of function symbols is convenient in actual programming, various attempts have been made to re-introduce function symbols, at least partly, into ASP. Let us point out two such attempts. First, Bonatti [6, 8] introduced a class of normal programs called *finitary* programs. These programs do admit function symbols but put two restrictions on the properties of the dependency

graph of the program. First, Bonatti requires that every ground atom transitively depends on only finitely many ground atoms. Second, the number of negative literals that are on cycles of odd length is finite. With these restrictions, given a finitary program $P$ and a ground atom $p$, one can find a finite set of atoms $U(P,p)$ and a finite 'relevant' subprogram $R(P,p)$ so that atoms occurring in $R(P,p)$ are all in $U(P,p)$ and the basic reasoning questions about $P$ and $p$ can be answered by asking the same questions about $R(P,p)$ and $p$. Thus, membership of $p$ in all stable models of $P$ is equivalent to the membership of $p$ in all stable models of $R(P,p)$. A similar result holds for the 'membership in some' question. Since $R(P,p)$ is finite, these questions can be answered, in fact they are in class NP, once the program $R(P,p)$ is an input. The key property exhibited by finitary programs is that the set $U(P,p)$ is a *splitting set* (see [15, 27]). The class of finitary programs has several other desirable properties. For example, it satisfies a natural type of compactness condition in that if all finite ground subprograms of a finitary program $P$ have a stable model, then $P$ has a stable model. The compactness property does not hold, in general, for normal logic programs [11]. It is also not the case that one can effectively test whether a program is finitary. That is, Bonatti noticed that the set of finitary programs is not recursive.

A radically different proposal has been, actually implemented in the ASP solver *smodels* [34, 36]. In this implementation the functions are stored. This implies that the Herbrand base is, in effect, finite, for each time we deal with a term $f(\vec{a})$, it is evaluated against the stored definition of $f$. Thus the Herbrand base is finite and all stable models are finite, too.

The outline of the paper is as follows. In Section 2, we shall formally define extended set based constraints, clauses and programs. We shall also define the analogue of Horn programs and stable models for extended set based programs. There are some major differences between extended set based Horn programs and standard logic programs. For example, the least model of a Horn program is always computed by iterating the one step provability operator for at most $\omega$ steps while this is not the case for ESB Horn programs. Also the least model of a recursive Horn program is always recursively enumerable (r.e.) while this is not necessarily the case for ESB Horn programs. In Section 3, we shall study the complexity of the least model $M$ of a recursive ESB Horn program $P$ with recursive constraints, i.e. all the constraints $P$ involve either explicit or recursive indices. In general, $M$ can be very complex. For example, for any arithmetic set $A$, we shall construct a recursive ESB Horn program $P_A$ with recursive constraints such that the degree of the least model of $P_A$ is $A$. We also show that the least

model of a recursive ESB Horn program $P$ with recursive constraints is always $\Pi^1_1$ and that there is a recursive ESB Horn program $P$ with recursive constraints whose least model is $\Pi^1_1$-complete. Moreover we show the question of whether $\omega$ is the least model of a recursive ESB Horn program $P$ with recursive constraints is $\Pi^1_1$-complete, as is the question of whether a recursive ESB Horn program $P$ with recursive constraints has a recursive least model.

There is a natural class of ESB programs which we call *weakly finite*, whose stable models are much better behaved. Roughly, an ESB program $P$ is weakly finite if there are only finitely many heads of clauses of $P$ that involve recursive or r.e. indices. In Section 2, we show that a recursive weakly finite ESB Horn program always has an r.e. least model. Moreover, we shall describe a class of recursive weakly finite ESB programs called effectively decidable where one can effectively compute an r.e. index of the least model from a recursive index of the program. In Section 4, we study the complexity of the question of when a recursive weakly finite ESB program has a recursive stable model. For example, we show that the predicate that a recursive weakly finite ESB with recursive constraints has a recursive stable model is $\Sigma^0_3$-complete. In Section 5, we state our conclusions and directions for further research.

## 2. ESB Constraints, Clauses and Programs

In this section, we shall give the formal definitions of ESB constraints, clauses, programs and define the analogue of Horn programs and stable models for ESB programs. To describe our constraints, we first need to describe three types of indices (i.e. codes) for subsets of the natural numbers.

(1) **Explicit indices of finite sets**. For each finite set $F \subseteq \omega$, we shall define the explicit index of $F$ as follows. The explicit index of the empty set is 0 and the explicit index of $\{x_1 < \ldots < x_n\}$ is $2^{x_1} + \cdots + 2^{x_n}$. We shall let $F_n$ denote the finite set whose index is $n$.

(2) **Recursive indices of recursive sets**. Let $\phi_0, \phi_1, \ldots,$ be an effective list of all partial recursive functions. By a recursive index of a recursive set $R$, we mean an $e$ such that $\phi_e$ is the characteristic function of $R$. If $\phi_e$ is a total $\{0,1\}$-valued function, then $R_e$ will denote the set $\{x \in \omega : \phi_e(x) = 1\}$.

(3) **R.e. indices of r.e. sets**. By a r.e. index of a r.e. set $W$, we mean an $e$ such that $W$ equals the domain of $\phi_e$, that is, $W_e = \{x \in \omega : \phi_e(x) \text{ converges}\}$.

No matter what type of indices we use, we shall always consider two types of constraints based on $X$ and a finite set of indices $\mathcal{F}$, namely,

$\langle X, \mathcal{F}\rangle^{=}$ and $\langle X, \mathcal{F}\rangle^{\subseteq}$. For any subset $M \subseteq \omega$, we shall say that $M$ *is a model of* $\langle X, \mathcal{F}\rangle^{=}$, written $M \models \langle X, \mathcal{F}\rangle^{=}$, if there exists an $e \in \mathcal{F}$ such that $M \cap X$ equals that set with index $e$. Similarly, we shall say that $M$ *is a model of* $\langle X, \mathcal{F}\rangle^{\subseteq}$, written $M \models \langle X, \mathcal{F}\rangle^{\subseteq}$, if there exists an $e \in \mathcal{F}$ such that $M \cap X$ contains the set with index $e$. One could consider more general constraints where we allow $\mathcal{F}$ to be an infinite set as well. We restrict ourselves to consider only constraints where $\mathcal{F}$ is a finite set of indices since these are the only types of constraints which have the possibility of being effectively processed.

Let us fix a recursive *pairing* function, say $[x, y] = \frac{1}{2}((x + y)^2 + 3x + y)$ from $\omega \times \omega$ to $\omega$. For any sequence $a_1, \ldots, a_n$, with $n \geq 2$, we define the code $c(a_1, \ldots, a_n)$ by the usual inductive procedure of defining $c(a_1, a_2) = [a_1, a_2]$ and $c(a_1, \ldots, a_n) = [a_1, c(a_2, \ldots, a_n)]$ if $n \geq 3$. The explicit index $ind(a_1, \ldots, a_n)$ of the sequence $\vec{s} = (a_1, \ldots, a_n)$ is defined by induction. If $n = 2$, then $ind(a_1, a_2) = [2, [a_1, a_2]]$ and if $n \geq 3$, then $ind(a_1, \ldots, a_n) = [n, c(a_1, \ldots, a_n)]$.

In this paper, we shall consider three different types of constraints.

(A) **Finite constraints**. Here we assume that we are given an explicit index $x$ of a finite set $X$ and a finite family $\mathcal{F}$ of explicit indices of finite subsets of $X$. Throughout this paper we shall identify the finite constraints $\langle X, \mathcal{F}\rangle^{=}$ and $\langle X, \mathcal{F}\rangle^{\subseteq}$ with their codes, $ind(0, 0, x, n)$ and $ind(0, 1, x, n)$ respectively where $\mathcal{F} = F_n$, that is the $n^{th}$ finite set (see above). Here the first coordinate 0 tells us that the constraint is finite, the second coordinate is 0 or 1 depending on whether the constraint is $\langle X, \mathcal{F}\rangle^{=}$ or $\langle X, \mathcal{F}\rangle^{\subseteq}$, and the third and fourth coordinates are the codes of $X$ and $\mathcal{F}$ respectively.

(B) **Recursive constraints**. Here we assume that we are given a recursive index $x$ of a recursive set $X$ and a finite family $\mathcal{R}$ of recursive indices of recursive subsets of $X$. Again we shall identify the recursive constraints $\langle X, \mathcal{R}\rangle^{=}$ and $\langle X, \mathcal{R}\rangle^{\subseteq}$ with their codes, $ind(1, 0, x, n)$ and $ind(1, 1, x, n)$ respectively, where $\mathcal{R} = F_n$. Here the first coordinate 1 tells us that the constraint is recursive, the second coordinate is 0 or 1 depending on whether the constraint is $\langle X, \mathcal{R}\rangle^{=}$ or $\langle X, \mathcal{R}\rangle^{\subseteq}$, and the third and fourth coordinates are the codes of $X$ and $\mathcal{R}$ respectively.

(C) **R.e. constraints**. Here we are given a r.e. index $x$ of a r.e. set $X$ and a *finite* family $\mathcal{W}$ of r.e. indices of r.e. subsets of $X$. Again we identify the finite constraints $\langle X, \mathcal{W}\rangle^{=}$ and $\langle X, \mathcal{W}\rangle^{\subseteq}$ with their codes, $ind(2, 0, x, n)$ and $ind(2, 1, x, n)$ respectively, where $\mathcal{W} = F_n$. The first coordinate 2 tells us that the constraint is r.e., the second coordinate is 0 or 1 depending on whether the constraint is $\langle X, \mathcal{W}\rangle^{=}$ or $\langle X, \mathcal{W}\rangle^{\subseteq}$, and the third and fourth coordinates are the codes of $X$ and $\mathcal{W}$.

An *extended set-based clause* is defined to be a clause of the form

$$\langle X, \mathcal{A}\rangle^* \leftarrow \langle Y_1, \mathcal{B}_1\rangle^{\subseteq}, \ldots, \langle Y_k, \mathcal{B}_k\rangle^{\subseteq}, \langle Z_1, \mathcal{C}_1\rangle^{=}, \ldots, \langle Z_l, \mathcal{C}_l\rangle^{=}, \qquad (1)$$

where $*$ is either $=$ or $\subseteq$. We shall refer to $\langle X, \mathcal{A}\rangle^*$ as the head of $C$, written $head(C)$, and $\langle Y_1, \mathcal{B}_1\rangle^{\subseteq}, \ldots, \langle Y_k, \mathcal{B}_k\rangle^{\subseteq}, \langle Z_1, \mathcal{C}_1\rangle^{=}, \ldots, \langle Z_l, \mathcal{C}_l\rangle^{=}$ as the body of $C$, written $body(C)$. Here either $k$ or $l$ may be 0. $M$ is said to be a model of $C$ if either $M$ does not model every constraint in $body(C)$ or $M \models head(C)$.

Again we shall talk about three different types of clauses.

(a) **Finite clauses**. These are clauses in which all of the constraints are finite constraints.

(b) **Recursive clauses**. These are clauses where all the constraints appearing in the clause are finite or recursive constraints and at least one constraint is a recursive constraint.

(c) **R.e. clauses**: These are clauses where all the constraints appearing in the clause are finite, recursive or r.e. constraints and there is at least one r.e. constraint.

*An extended set-based* (ESB) program $P$ is a set of clauses of the form of (1). We say that an ESB program $P$ is recursive, if the set of codes of the clauses of $P$ is a recursive set. Here the code of a clause $C$ of the form of (1) is $ind(c, e_1, \ldots, e_k, f_1, \ldots, f_l)$ where $c$ is the code of $\langle X, \mathcal{A}\rangle^*$, $e_i$ are the codes of $\langle Y_i, \mathcal{B}_i\rangle^{\subseteq}$ for $i = 1, \ldots, k$ and $f_j$ are the codes of $\langle Z_j, \mathcal{C}_j\rangle^{=}$ for $j = 1, \ldots, l$.

Given a program $P$, we let $Fin(P)$ ($Rec(P)$, $RE(P)$) denote the set of all finite (recursive, r.e.) clauses in $P$. It is easy to see from our coding of clauses that if $P$ is a recursive ESB program, then $Fin(P)$, $Rec(P)$ and $RE(P)$ are also recursive ESB programs.

**Definition 2.1.** *Let $P$ be a recursive ESB program.*

1. *We say that $P$ is* recursive with finite constraints *if $P = Fin(P)$. (Note if $P = Fin(P)$, then all the constraints only involve finite sets.)*

2. *We say that $P$ is* recursive with recursive constraints *if $P = Fin(P) \cup Rec(P)$ and $Rec(P) \neq \emptyset$.*

3. *We say that $P$ is* recursive with r.e. constraints *if $RE(P) \neq \emptyset$.*

4. *We say that $P$ is* weakly finite with recursive constraints *if $P$ is recursive with recursive constraints and the set of heads of clauses in $Rec(P)$ is finite.*

5. *We say that $P$ is* weakly finite with r.e. constraints *if $P$ is recursive with r.e. constraints and the set of heads of clauses in $Rec(P) \cup RE(P)$ is finite.*

Next we define the analogue of Horn programs for ESB programs. A (ESB) *Horn program* $P$ is a set of clauses of the form

$$\langle X, \mathcal{A}\rangle^{\subseteq} \leftarrow \langle Y_1, \mathcal{B}_1\rangle^{\subseteq}, \ldots, \langle Y_k, \mathcal{B}_k\rangle^{\subseteq}. \tag{2}$$

where $\mathcal{A}$ is a singleton, that is $\mathcal{A}$ consists of a single index. We define the *one-step provability operator*, $T_P : 2^\omega \to 2^\omega$, so that for any $S \subseteq \omega$, $T_P(S)$ is the union of the set of all $D_e$ such that there exists a clause $C \in P$ such that $S \models body(C)$, $head(C) = \langle X, \mathcal{A}\rangle^{\subseteq}$ and $A = \{e\}$ where $D_e = F_e$ if $head(C)$ is a finite constraint, $D_e = R_e$ if $head(C)$ is a recursive constraint, and $D_e$ is $W_e$ if $head(C)$ is an r.e. constraint. It is easy to see that $T_P$ is a monotone operator and hence there is a least fixed point of $T_P$ which we denote by $N^P$. Moreover it is easy to check that $N^P$ is a model of $P$.

If $P$ is an ESB Horn program in which the body of every clause consists of *finite* constraints, then one can easily prove that the least fixed point of $T_P$ is reached in $\omega$-steps, that is, $N^P = T_P \uparrow^\omega (\emptyset)$. However, if we allow clauses whose bodies contain either recursive or r.e. constraints, then we can no longer guarantee that we reach the least fixed point of $T_P$ in $\omega$ steps. Here is an example.

**Example 2.2.** Let $e_n$ be the explicit index of the set $\{n\}$ for all $n \geq 0$, let $w$ be a recursive index of $\omega$ and $f$ be a recursive index of the set of even numbers $E$. Consider the following program.

$$\begin{aligned} \langle\{0\}, \{e_0\}\rangle^{\subseteq} &\leftarrow \\ \langle\{2x+2\}, \{e_{2x+2}\}\rangle^{\subseteq} &\leftarrow \langle\{2x\}, \{e_{2x}\}\rangle^{\subseteq} \quad \text{(for every number } x\text{)} \\ \langle\omega, \{w\}\rangle^{\subseteq} &\leftarrow \langle E, \{f\}\rangle^{\subseteq} \end{aligned}$$

Clearly $\omega$ is the least model of $P$ but it takes $\omega + 1$ steps to reach the fixed point. That is, it is easy to check that $T_P \uparrow^\omega = E$ and that $T_P \uparrow^{\omega+1} = \omega$.

**Lemma 2.3.** *If $P$ is a recursive ESB Horn Program with finite constraints, then the least fixed point of the one step provability operator $T_P$ is r.e..*

*Proof.* Suppose that $W$ is an r.e. set and $W$ is the domain of the total recursive function $\phi$. For all $s \geq 0$, we let $W_s$ equal the set of all $n \leq s$ such that $\phi(n)$ converges in $s$ or fewer steps. It is then easy to see that since $P$ is recursive and all the constraints involved in clauses of $P$ are

finite, that we can uniformly compute the r.e. index of the set $A_s$ of all $x$ such that there exists a clause

$$C = \langle X, \mathcal{A}\rangle^{\subseteq} \leftarrow \langle Y_1, \mathcal{B}_1\rangle^{\subseteq}, \ldots, \langle Y_k, \mathcal{B}_k\rangle^{\subseteq},$$

such that $W_s$ satisfies the body of $C$ and $x \in F_e$ where $\mathcal{A} = \{e\}$. Thus $T_P(W_s) = A_s$ and $A_0, A_1, \ldots$ is an effective sequence of increasing r.e. sets. It follows that we can effectively compute $A = \bigcup_{s\geq 0} A_s = T_P(W)$ from any r.e. index of $W$. It then easily follows by induction that we can uniformly compute an r.e. index $U_n = T_P^n(\emptyset)$ for each $n \geq 0$. Thus the least model of $P$, $U = \bigcup_{n\geq 0} U_n = T_P \uparrow_\omega (\emptyset)$, is r.e.. □

As evidenced by the work in [9, 16, 17, 18], one can extend the syntax of logic programming to reason about finite sets directly. ESB programs with finite constraints is clearly another way to extend the syntax of logic programming to reason about finite sets directly. However, there is a difference between our approach to reasoning about finite sets and the approach in [9, 16, 17, 18], namely, we reason about finite sets by dealing with their codes (indices) and not their explicit representations. This does not allow us to use the set constructors to develop data structures as is possible in the language *log* [17], but because we reason about indices of sets, it is easy to extend our formalism to allow other types of indices such as recursive and r.e. indices. Theorem 2.5 below provides at least one set of conditions on the set of indices of sets that we employ which ensures that we can effectively compute with weakly finite ESB Horn programs as opposed to just computing with ESB Horn programs with finite constraints. However, before stating such conditions, we first prove a basic result on the complexity of the least models of recursive weakly finite ESB Horn programs with recursive or r.e. constraints.

**Theorem 2.4. (a)** *If $P$ is a weakly finite ESB Horn program with recursive constraints such that $Fin(P)$ is recursive, then the least fixed point of the one step provability operator $T_P$ is r.e.*

**(b)** *If $P$ is a weakly finite ESB Horn program with r.e. constraints such that $Fin(P)$ is recursive, then the least fixed point of the one step provability operator $T_P$ is r.e.*

*Proof.* To prove part (a), we will present an informal program which constructs the least fixed point in a finite number of steps.
**Step (1).** First take $Fin(P)$ and construct the least fixed point which we will call $U_0$. Since $Fin(P)$ is recursive, $U_0$ is r.e., by Lemma 2.3. Next consider the set $T_1 = U_0 \cup S_0$ where $S_0$ is the union of the set of all $C_e$ such that there exists a clause $C \in Rec(P)$ such that $U_0 \models body(C)$ and $head(C) = \langle X, \mathcal{F}\rangle^{\subseteq}$ where $\mathcal{F} = \{e\}$ and $C_e = F_e$ if $head(C)$ is a

finite constraint and $C_e = R_e$ if $head(C)$ is a recursive constraint. Even though we cannot find $S_0$ recursively, our hypothesis ensures that $S_0$ is a finite union of finite or recursive sets and hence is a recursive set. Thus $T_1$ is an r.e. set. Now if $S_0 = \emptyset$, then we halt and return $T_1$; $T_1$ is the least fixed point of $P$. Otherwise go onto step (2).

We now present the description of Step $n+1$, for $n \geq 1$. Assume that $T_n$ is the result of step $n$.
**Step** $(n+1)$**.** Consider the set $U_n = T_{Fin(P)} \uparrow^\omega (T_n)$. It is easy to see that that since $T_n$ is r.e., $U_n$ is r.e. Next consider the set $T_{n+1} = U_n \cup S_n$ where $S_n$ is the union of the set of all $c_e$ such that there exists a clause $C \in Rec(P)$ such that $U_n \models body(C)$ and $head(C) = \langle X, \mathcal{F} \rangle^{\subseteq}$ where $\mathcal{F} = \{e\}$ and $C_e = F_e$ if $head(C)$ is a finite constraint and $C_e = R_e$ if $head(C)$ is a recursive constraint. Again, even though we cannot find $S_n$ recursively, our hypothesis ensures that $S_n$ is a finite union of finite or recursive sets and hence is a recursive set. Thus $T_{n+1}$ is an r.e. set. Now if $S_n = \emptyset$, then we halt and return $T_{n+1}$; $T_{n+1}$ is the least fixed point of $P$. Otherwise go onto step $(n+2)$.

Since the set of all $head(C)$ such $C \in Rec(P)$ is finite, it easily follows that this process must stop after a finite number of steps and hence the least model of $P$ is r.e..

The proof of part (b) is similar to the proof of part (a). □

We note that there is an alternative way to obtain the least model of weakly finite ESB program with recursive or r.e. constraints. Namely, if $P$ is a recursive weakly finite program with either recursive or r.e. constraints, let $\mathcal{H}(P)$ denote the set of all $head(C)$ where $C \in Rec(P) \cup RE(P)$. By definition, $\mathcal{H}(P)$ is a finite set consisting of constraints of the form $C_{X,e} = \langle X, \mathcal{A} \rangle^{\subseteq}$ where $A = \{e\}$. In such a situation, we let $S_{C_{X,e}} = F_e$ if $e$ is an explicit index of a finite set, $S_{C_{X,e}} = R_e$ if $e$ is a recursive index and $S_{C_{X,e}} = W_e$ if $e$ is an r.e. index. If $S \subseteq \mathcal{H}(P)$, then let $U_S = \bigcup_{C_{X,e} \in S} S_{C_{X,e}}$. Then it is clear from our proof of Theorem 2.4 that the least model $M$ of $P$ is of the form

$$M = T_{Fin(P)} \uparrow^\omega (U_S)$$

for some $S \subseteq \mathcal{H}(P)$. This point of view allows us to specify some simple conditions which ensure that we can effectively compute the least fixed point of the operator $T_P$ of a weakly finite program $P$ with recursive constraints or r.e. constraints. That is, suppose that $P$ is a weakly finite recursive program $P$ with recursive or r.e constraints. Then for each subset $S$ of $\mathcal{H}(P)$, we let $M_S = T_{Fin(P)} \uparrow^\omega (U_S)$. It is easy to see that $M_S$ is r.e. for all $S \subseteq \mathcal{H}(P)$. Then we say that $P$ is *effectively decidable* if there is an algorithm which given any constraint $\langle X, \mathcal{F} \rangle^{\subseteq}$

that appears in the body of some clause in $Rec(P)$ or $RE(P)$ and any subset $S \subseteq \mathcal{H}(P)$, we can effectively decide if $M_S \models \langle X, \mathcal{F} \rangle^{\subseteq}$. Note that if we consider the informal algorithm to compute the least fixed point in parts (a) and (b) of Theorem 2.4, then it is easy to see that if $P$ is effectively decidable, then we have an effective procedure to compute the index of the sets $S_0, S_1, \ldots$ that appear in the computation which, in turn, means that we have an effective procedure to compute the indices of $T_0, T_1, \ldots$. Effective decidability does not allows us to decide whether a given $S_i = \emptyset$, but this does not matter since even if $S_i = \emptyset$, we can still effectively compute an r.e. index for $T_i$. Now, if $|\mathcal{H}(P)| = n$, then it must be the case that $T_n$ is the least model of $P$. Thus we have the following.

**Theorem 2.5.** **(a)** *If $P$ is a recursive ESB Horn Program with finite constraints, then we can effectively compute an r.e. index of the least fixed point of the one step provability operator $T_P$ from a recursive index of $P$.*

**(b)** *If $P$ is a weakly finite recursive ESB Horn program with recursive constraints and $P$ is effectively decidable, then we can effectively compute an r.e. index of the least fixed point of the one step provability operator $T_P$ from recursive index of $P$.*

**(c)** *If $P$ is a weakly finite recursive ESB Horn program with r.e. constraints and $P$ is effectively decidable, then we can effectively compute an r.e. index of the least fixed point of the one step provability operator $T_P$ from a recursive index of $P$.*

The hypothesis that the $P$ is a weakly finite Horn program with recursive or r.e. constraints is absolutely necessary for the proof of Theorem 2.4 as our next example will show.

**Example 2.6.** Suppose that we are given a sequence of pairwise disjoint infinite recursive sets $Y, X_0, A_0, X_1, A_1, \ldots$. Let

$$\begin{aligned} Y &= \{y_0 < y_1 < \ldots\}, \\ X_e &= \{x_{0,e} < x_{1,e} < \ldots\} \text{ for each } e \in \omega \text{ and} \\ A_e &= \{a_{0,e} < a_{1,e} < \ldots\} \text{ for each } e \in \omega. \end{aligned}$$

For all $k \geq 0$, we shall let $X_{e,\geq k} = \{x_{k,e} < x_{k+1,e} < \cdots\}$.

Given an atom $a$, the finite set constraint $\langle \{a\}, \{n\} \rangle^{\subseteq}$ where $F_n = \{a\}$ is satisfied by a model $M$ iff $a \in M$ so that the set constraint $\langle \{a\}, \{n\} \rangle^{\subseteq}$ acts like a atom in a normal logic program. Thus we shall abbreviate that finite set constraint $\langle \{a\}, \{n\} \rangle^{\subseteq}$ by the atom $a$.

Let $W_e^s$ denote the finite set of elements $z$ such that $\phi_e(z)$ converges in $s$ or fewer steps. Now consider the following program.

(1) $x_{n,e} \leftarrow a_{[n,s],e}$ for all $n$ such that $n \in W_e^s - W_e^{s-1}$.

(2) $a_{[n,s],e} \leftarrow$ for all $n$ such that $n \in W_e^s - W_e^{s-1}$.

(Note the set of clauses in (1) and (2) are recursive since the condition that $n \in W_e^s - W_e^{s-1}$ is a recursive condition. Moreover the effect of these clauses is to ensure that $\{x_{n,e} : n \in W_e\}$ is contained in the least model of $P$ for every $e$. We will ensure that these are the only clauses in $P$ with an $x_{n,e}$ contained in the head of the clause so that the least model of $P$ restricted to $X_e$ will be precisely $\{x_{n,e} : n \in W_e\}$.)

(3) $y_e \leftarrow \langle X_e, \{n_k\}\rangle^{\subseteq}$ for all $k \geq 0$ where $n_k$ is a recursive index of $X_{e,\geq k}$. (Note that the net effect of these clauses is to ensure that $y_e$ is in the least model of $P$ if and only if $X_{e,\geq k} \subseteq M \cap X_e$ for some $k$, which happens if and only if $W_e$ is cofinite.)

It is now easy to see that $P$ is a recursive ESB Horn program such that the least model of $P$ equals $\{y_e : W_e \text{ is cofinite}\} \cup \{x_{n,e} : n \in W_e\}$. However it is known [37] that the set $\{e : W_e \text{ is cofinite}\}$ is a complete $\Sigma_3^0$ set so that $M$ is a complete $\Sigma_3^0$ set and hence is certainly not r.e. □

We are in a position to define the analogue of a stable model for ESB programs.

**Definition 2.7.** *Suppose that $M$ is a model of an ESB program $P$.*

1. *We define the analogue of the NSS-transform by defining the clauses in $\mathbf{NSS}_M(P)$ corresponding to each clause $C \in P$ of the form (1), as follows. $\mathbf{NSS}_M(C)$ is nil if $M$ does not satisfy the body of $C$. If $M$ does satisfy the body of $C$, then since $M$ is a model of $P$, it must also be a model of the head of $C$, $\langle X, \mathcal{A}\rangle^*$ where $*$ is either $=$ or $\subseteq$. If $* = \subseteq$, there must be an explicit (recursive, r.e.) index $e$ in $\mathcal{A}$ such that $M \cap X$ contains the set with index $e$ and for each such $e$, $\mathbf{NSS}_M(C, e)$ is the clause*

$$\langle X, \{e\}\rangle^{\subseteq} \leftarrow \langle Y_1, \mathcal{B}_1\rangle^{\subseteq}, \ldots, \langle Y_k, \mathcal{B}_k\rangle^{\subseteq}, \langle Z_1, \mathcal{C}_1\rangle^{\subseteq}, \ldots, \langle Z_l, \mathcal{C}_l\rangle^{\subseteq}. \quad (3)$$

*Similarly, if $*$ is $=$, there must be an index $e$ such that $M \cap X$ is the set coded by $e$ and again for each such $e$, $\mathbf{NSS}_M(C, e)$ is the clause*

$$\langle X, \{e\}\rangle^{\subseteq} \leftarrow \langle Y_1, \mathcal{B}_1\rangle^{\subseteq}, \ldots, \langle Y_k, \mathcal{B}_k\rangle^{\subseteq}, \langle Z_1, \mathcal{C}_1\rangle^{\subseteq}, \ldots, \langle Z_l, \mathcal{C}_l\rangle^{\subseteq}. \quad (4)$$

*Then $\mathbf{NSS}_M(P) = \{\mathbf{NSS}_M(C, e) : C \in P\}$ will be an ESB Horn program.*

2. *We then say that $M$ is a* stable model of $P$ *if $M$ is a model of $P$ and $M$ equals the least model of* $\mathbf{NSS}_M(P)$.

We will now give an example that illustrates the use of our ESB programs. In this example, we show how an ESB program can be used to reason about certain subspaces of an $n$-dimensional vector space $V_n$, over rational numbers.

**Example 2.8.** Suppose that $V_\infty$ is an infinite dimensional vector space over the rational numbers $Q$. We can think of $V_\infty$ as the set of all finite sequences $\langle a_1, \ldots, a_n \rangle$ where $a_i \in Q$ and $a_n \neq 0$ and, under a suitable coding, we can consider $V_\infty$ to be a recursive subset of the natural numbers $N$. Now suppose that we are given a finite or recursive independent set $I \subseteq V_\infty$ with at least 6 elements and we wish to write a ESB Horn program whose stable models are precisely the set of all three dimensional subspaces of $V_\infty$ that are generated by three elements subsets of $I$. Given any finite set of elements $a_1, \ldots, a_k$ of $V_\infty$, let $\{a_1, \ldots, a_k\}^*$ denote the subspace generated by $a_1, \ldots, a_n$. Note that we can find a recursive index $R_{a_1,\ldots,a_k}$ for $\{a_1, \ldots, a_k\}^*$ from $a_1, \ldots, a_n$. Moreover for each finite set $S \subseteq V_\infty$, we let $e_S$ denote the explicit index of the set $S$. If $S$ is a singleton, say $S = \{v\}$, we shall simply write $v$ for the set constraint $\langle S, \{e_S\}\rangle^{\subseteq}$. We shall let $\alpha$ denote the explicit index of the empty set and $\beta$ denote the recursive index of $V_\infty$. Then we consider the ESB program $P$ which consists of the following sets of clauses.

$A_{b_1,b_2,b_3} : \langle V_\infty, \{R_{b_1,b_2,b_3}\}\rangle^{\subseteq} \leftarrow b_1, b_2, b_3$
for all three element subsets $\{b_1, b_2, b_3\}$ of $I$.

$B_{b_1,b_2,b_3,b_4} : \langle V_\infty, \{\beta\}\rangle^{\subseteq} \leftarrow b_1, b_2, b_3, b_4$
for all four element subsets $\{b_1, b_2, b_3, b_4\}$ of $I$.

$C_{b_1,b_2,b_3} : \langle \{b_1, b_2, b_3\}, \{e_{\{b_1,b_2,b_3\}}\}\rangle^{\subseteq} \leftarrow \langle I/\{b_1, b_2, b_3\}, \{\alpha\}\rangle^{=}$
for all three element subsets $\{b_1, b_2, b_3\}$ of $I$.

Note that if $I$ is finite, then $P$ will be a finite ESB program.

Now consider a set $M \subseteq V_\infty$ which is a model of $P$. If $M \cap I = \{b_1, b_2, b_3\}$ for some three element subset of $I$, then $M$ satisfies the body of $A_{b_1,b_2,b_3}$. It follows that $M$ must also satisfy the head of $A_{b_1,b_2,b_3}$ so that $M$ must contain $\{b_1, b_2, b_3\}^*$. Then in this case $\mathbf{NSS}_M(P)$ consists of the following two clauses,

$\langle V_\infty, \{R_{b_1,b_2,b_3}\}\rangle^{\subseteq} \leftarrow b_1, b_2, b_3$

$\langle \{b_1, b_2, b_3\}, \{e_{\{b_1,b_2,b_3\}}\}\rangle^{\subseteq} \leftarrow \langle I/\{b_1, b_2, b_3\}, \{\alpha\}\rangle^{\subseteq}$

and it is easy to see that the least model of these two clauses is $\{b_1, b_2, b_3\}^*$ so that $M$ is a stable model of $P$ if and only if $M = \{b_1, b_2, b_3\}^*$.

If $|M \cap I| \geq 4$, then $M$ satisfies the body of one of clauses $B_{b_1,b_2,b_3,b_4}$ for some 4 element subset $\{b_1, b_2, b_3, b_4\}$ of $I$. But then since $M$ is a model of $P$, $M$ must satisfy the body of the $B_{b_1,b_2,b_3,b_4}$ which implies that $M = V_\infty$. Thus in this case, $M$ does not satisfy any of the clauses of the form $C_{b_1,b_2,b_3}$ and hence $\mathbf{NSS}_M(P)$ consists of all clauses of the form $A_{b_1,b_2,b_3}$ and $B_{b_1,\ldots,b_4}$. It is then easy to see that the least model of $\mathbf{NSS}(P)$ is empty so that $M$ is not stable.

If $|M \cap I| \leq 2$, then $M$ satisfies the body of one of clauses $C_{b_1,b_2,b_3}$ for some 3 element subset $\{b_1, b_2, b_3\}$ of $I$. But then since $M$ is a model of $P$, $M$ must satisfy the body of the $C_{b_1,b_2,b_3}$ which implies that $\{b_1, b_2, b_3\} \subseteq M$ which is a contradiction.

Thus the only stable models of $P$ are of the form $\{b_1, b_2, b_3\}^*$ for some three element subset of $I$. □

## 3. The complexity of the least model of a recursive ESB Horn program

There are many complexity issues that need to be explored with respect of ESB programs. In this section, we shall study the complexity of the least model of a recursive ESB Horn program. The arithmetic hierarchy is defined as usual so that the recursive sets are both $\Sigma^0_0$ and $\Pi^0_0$, a $\Sigma^0_{n+1}$ set is obtained by existential number quantification over a $\Pi^0_n$ set, and a $\Pi^0_{n+1}$ set is the complement of a $\Sigma^0_{n+1}$ set. In particular, a set of natural numbers is $\Sigma^0_1$ if and only if it is an r.e. set. We say that a subset $A$ of $\omega$ is $\Delta^0_n$ is $A$ is both $\Sigma^0_n$ and $\Pi^0_n$.

Our next two results will show that if we remove the assumption of *weakly finite* in the premises of Theorem 2.4, then the least model of an ESB Horn program is no longer restricted to appear in a certain level of the arithmetic hierarchy.

**Theorem 3.1.** *For any arithmetic set $A$, there is a recursive ESB Horn program $P_A$ with recursive constraints such that the Turing degree of the least model $M$ of $P_A$ is equal to the degree of $A$.*

*Proof.* Let $A$ be an arithmetic set. Then there is a recursive predicate $R(i, x_1, \ldots, x_n)$ such that

$$i \in A \iff (Q_1 x_1)(Q_2 x_2) \ldots (Q_n x_n)(R(i, x_1, \ldots, x_n)) \qquad (5)$$

where each $Q_i$ is either $\forall$ or $\exists$. For any $1 \leq k < n$, let $X_{i,x_1,\ldots,x_k} = \{[k+2, i, x_1, \ldots, x_k, x] : x \in \omega\}$. Clearly, $X_{i,x_1,\ldots,x_k}$ is a recursive set. We then let $e_{i,x_1,\ldots,x_k}$ be a recursive index for $X_{i,x_1,\ldots,x_k}$. In addition for any $n \in \omega$, we let $f_n$ be an explicit index for the set $\{n\}$ and we shall abbreviate the set constraint $\langle\{n\}, \{f_n\}\rangle^{\subseteq}$ by the atom $n$.

Now consider the ESB Horn program $P = P_A$, consisting of the following $n+2$ groups of clauses, created in reverse order. Let $M = M^P$ denote the least model of $P_A$.

$\mathbf{C}_{n+1}$: For all $i, x_1, \ldots, x_n$ such that $R(i, x_1, \ldots, x_n)$, put in the clause

$$[n+1, i, x_1, \ldots, x_n] \leftarrow .$$

(This will be the only group of clauses whose head will involve elements of the form $[n+1, z]$, so that $M$ will contain $[n+1, z]$ if and only if $z = [i, x_1, \ldots, x_n]$ and $R(i, x_1, \ldots, x_n)$ holds.)

Now assume by induction that we have defined the set of clauses $\mathbf{C}_{n+1}, \mathbf{C}_n, \ldots, \mathbf{C}_{k+1}$ so that $[k+1, i, x_1, \ldots, x_k] \in M$ if and only if

$$(Q_{k+1}x_{k+1}) \ldots (Q_n x_n)(R(i, x_1, \ldots, x_k, x_{k+1}, \ldots, x_n))$$

holds. (Note that for $k = n$, the sequence of quantifiers will be empty.) Then we define the clauses in $\mathbf{C}_k$ as follows.

$\mathbf{C}_k$: There are two cases.

**Case** $1_k$. $Q_k = \forall$. In this case, we add for all $i, x_1, \ldots, x_k$ the clause

$$[k, i, x_1, \ldots, x_{k-1}] \leftarrow \langle X_{i,x_1,\ldots,x_{k-1}}, \{e_{i,x_1,\ldots,x_{k-1}}\}\rangle^{\subseteq}$$

(Again we shall ensure that these are the only clauses that involve elements of the form $[k, z]$ in the head, so that $M$ will satisfy

$$\langle X_{i,x_1,\ldots,x_{k-1}}, \{e_{i,x_1,\ldots,x_{k-1}}\}\rangle^{\subseteq}$$

only if for all $x \in \omega$, $[k+1, i, x_1, \ldots, x_{k-1}, x] \in M$. By our induction hypothesis, this happens if and only if

$$(\forall x_k)(Q_{k+1}x_{k+1}) \ldots (Q_n x_n)(R(i, x_1, \ldots, x_{n-1}, x_n)).$$

Thus in this case, $M$ will contain $[k, z]$ if and only if $z = [i, x_1, \ldots, x_{n-1}]$ and $(\forall x_k)(Q_{k+1}x_{k+1}) \ldots (Q_n x_n)(R(i, x_1, \ldots, x_{n-1}, x_n))$ holds.)

**Case** $2_k$. $Q_k = \exists$. In this case, we add, for all $x \in \omega$, the clause

$$[k, i, x_1, \ldots, x_{k-1}] \leftarrow [k+1, i, x_1, \ldots, x_{k_1}, x]$$

(Again we shall ensure that these are the only clauses that involve elements of the form $[k, z]$ in the head. Now by our induction hypothesis, $M$ contains $[k+1, i, x_1, \ldots, x_{k_1}, x]$ if and only if

$$(Q_{k+1}x_{k+1})\ldots(Q_nx_n)(R(i, x_1, \ldots, x_{k-1}, x, x_{k+1}, \ldots, x_n)).$$

Thus in this case, $M$ will contain $[k, z]$ if and only if $z = [i, x_1, \ldots, x_{k-1}]$ and

$$(\exists x_k)(Q_{k+1}x_{k+1})\ldots(Q_nx_n)(R(i, x_1, \ldots, x_{k-1}, x_k, \ldots, x_n)).$$

Clearly $P_A$ is an ESB Horn program with recursive constraints. It follows by induction that the least model $M$ of $P_A$ will equal

$\{[i, 1] : (Q_1x_1)\ldots(Q_nx_n)(R(i, x_1, \ldots, x_n))\} \cup$
$\bigcup_{k=1}^{n}\{[k+1, x_1, \ldots, x_k] : (Q_{k+1}x_{k+1})\ldots(Q_nx_n)(R(i, x_1, \ldots, x_n))\}$.

Clearly $\{[i, 1] : (Q_1x_1)\ldots(Q_nx_n)(R(i, x_1, \ldots, x_n))\}$ is Turing equivalent to $A$. Moreover, one can show that for any degree $\delta$ that contains an arithmetic set, there is a set $A$ of degree $\delta$ and a recursive predicate $R$ such that
(i) $i \in A \iff (Q_1x_1)(Q_2x_2)\ldots(Q_nx_n)(R(i, x_1, \ldots, x_n))$ and
(ii) $\{(x_1, \ldots, x_{k-1}) : (Q_kx_k)\ldots(Q_nx_n)(R(i, x_1, \ldots, x_n))\}$ is Turing reducible to $A$ for all $2 \leq k \leq n$. Thus for such $A$, the least model $M^P$ of $P_A$ will be Turing equivalent to $A$.

The only remaining thing to check is that $P_A$ is a recursive program. We note that for all $i, x_1, \ldots, x_k$, we can use the $S$-$m$-$n$ Theorem to uniformly compute recursive indices $e_{i,x_1,\ldots,x_k}$ for the set $X_{i,x_1,\ldots,x_k}$. Thus it is easy to see that $P_A$ is a recursive program. □

More generally, our next result will show that various questions about the least model of an arbitrary recursive ESB Horn program are $\Pi^1_1$ hard. Note that Theorem 2.4 allowed us to prove that the least model of a ESB Horn Program is r.e. if we impose significant limitations on ESB Horn program $P$, i.e. that $FIN(P)$ is recursive and $P$ is weakly finite. Our next result we show that in general, the least model of an ESB recursive Horn program can be very complex.

**Theorem 3.2.** *Let $R$ be a recursive set and let $P$ be a recursive ESB Horn program.*

*(a) The least model $M$ of $P$ is a $\Pi^1_1$ set and there is a recursive ESB Horn program $P^*$ whose least model $M^*$ is $\Pi^1_1$-complete.*

*(b) The predicate "$R$ is the least fixed point of $P$" is $\Pi^1_1$-complete.*

*(c) The predicate "$P$ has a recursive stable model" is $\Pi^1_1$-complete.*

*Proof.* Before we can give our proof, we must first establish some notation for trees. Let $\omega^{<\omega}$ denote the set of finite strings of natural numbers, let $\emptyset$ denote the empty string and let $\sigma \prec \tau$ denote that $\sigma$ is an initial segment of $\tau$. A subset $T$ of $\omega^{<\omega}$ is a *tree* if $\emptyset \in T$ and whenever $\tau \in T$ and $\sigma \prec \tau$, then $\sigma \in T$. An infinite sequence $X \in \omega^{\omega}$ is said to be a *path* through the tree $T$ if $(X(0), X(1), \ldots, X(n-1)) \in T$ for all $n$. Let $T_e$ denote the $e$-th primitive recursive tree as defined in [13]. It is well-known that $\{e : \text{there is no infinite path through } T_e\}$ is a complete $\Pi^1_1$ set, see [24]. For any finite string $\sigma = (\sigma_1, \ldots, \sigma_n)$, we will write $[\sigma]$ for the code $[n, \sigma_1, \ldots, \sigma_n]$. We let $0 = [\emptyset]$ be the code of the empty sequence $\emptyset$. For any finite string $\sigma = (\sigma_1, \ldots, \sigma_n)$, we let $\sigma^\frown i$ denote the string $(\sigma_1, \ldots, \sigma_n, i)$. In what follows, we shall abbreviate the finite constraint $\langle \{n\}, \{f_n\} \rangle^{\subseteq}$ where $f_n$ is the explicit index of $\{n\}$ by simply writing the atom $n$.

*Proof of part (a).* Given a recursive ESB Horn program $P$, there is a $\Pi^0_1$ monotone operator $\Gamma$ such that the least model $M$ of $P$ is the closure of $\Gamma$ started on the empty set, which is automatically $\Pi^1_1$ by the classical result of Spector [38].

For the completeness, we will define a recursive ESB Horn program $P$ with recursive restraints and least fixed point $M$ such that

$$[e, [\emptyset]] \in M \iff T_e \text{ has no infinite path.}$$

Let $R_{f(e,\sigma)}$ denote the recursive set $\{[e, [\sigma^\frown i]] : i \in \omega\}$. The program $P$ has the following two sets of clauses.

(1) $[e, [\sigma]] \leftarrow$ for all $\sigma \notin T_e$ and

(2) $[e, [\sigma]] \leftarrow \langle \omega, \{f(e, \sigma)\} \rangle^{\subseteq}$ for all $e \in \omega$ and $\sigma \in \omega^{<\omega}$.

Now consider the one-step provability operator $T_P$ for $P$. We define $T^\alpha_P$ for all ordinal $\alpha > 0$ by defining $T^1_P(S) = T_P(S)$, $T^{\alpha+1}_P(S) = T_P(T^\alpha_P(S))$ when $\alpha + 1$ is a successor ordinal and $T^\lambda_P(S) = \bigcup_{\alpha<\lambda} T^\alpha_P(S)$ for $\lambda$ a limit ordinal. Then the least model $M$ of $P$ equal $T^\beta_P(\emptyset)$ for some ordinal $\beta \geq 1$. One can easily prove by ordinal induction that if $\sigma$ is a node on an infinite path through $T_e$, then $[e, [\sigma]]$ is not in $T^\alpha_P(\emptyset)$ for all $\alpha \leq \beta$. Hence if $T_e$ has an infinite path, then $[e, [\emptyset]] \notin M$. On the other hand, suppose that $[e, [\emptyset]] \notin M$. It can not be the case that for all $i$, $[e, [(i)]] \in M$ since otherwise there exists some ordinal $\alpha_i$ such that $[e, [(i)]] \in T^{\alpha_i}_P(\emptyset)$ and hence if $\gamma = sup\{\alpha_i : i \in \omega\}$, then it would be the case that $[e, [\emptyset]] \in T^{\gamma+1}_P$. Thus it follows that there exists an $i_0$ such that $[e, [(i_0)] \notin M$ and hence $(i_0) \in T_e$. Then one can use the same argument to show that there must be some $i_1$ such that $[e, [(i_0, i_1)]] \notin M$ and hence $(i_0, i_1) \in T_e$. Continuing by recursion in

this way, we can construct an infinite path $(i_0, i_1, \ldots)$ through $T_e$.

*Proofs of parts (b) and (c).* We will define a recursive function $h$ and recursive ESB Horn programs $P_{h(e)}$ with recursive constraints such that $\omega$ is the least model $M_e$ of $P_{h(e)}$ if $T_e$ has no infinite path and such that $M_e$ is not recursive if $T_e$ has an infinite path. Let $W$ be a non-recursive r.e. set and let $g$ be a one-to-one recursive function such that $W = \{g(0), g(1), \ldots\}$ is the range of $g$. Let $R_{f_2(e,\sigma)}$ denote the recursive set $\{5^{[\sigma^\frown i]} : i < \omega\}$. There are five sets of clauses in $P_{h(e)}$.

1. $2^{[i,g(i)]} \leftarrow$ for all $i \in \omega$.
2. $3^x \leftarrow 2^{[y,x]}$ for all $x, y \in \omega$.
3. $5^{[\sigma]} \leftarrow$ for all $\sigma \notin T_e$.
4. $5^{[\sigma]} \leftarrow \langle \omega, \{f_2(e,\sigma)\}\rangle^{\subseteq}$ for all $\sigma \in \omega^{<\omega}$.
5. $m \leftarrow 5^{[\emptyset]}$ for all $m \in \omega$.

Note the set of clauses of type of (1) are in $P_{h(e)}$ are recursive since $g$ is a total recursive function. The set of clauses of type (3) are recursive since $T_e$ is a primitive recursive tree. Finally the set of clauses of type (2), (4) and (5) are recursive. Thus $P_{h(e)}$ is a recursive ESB Horn program with recursive restraints for all $e$.

If $T_e$ has an infinite path $X$, then we can argue as in part (a), that $5^{[\emptyset]} \notin M$, and hence $3^a \in M_e \iff a \in W$. Thus in such a case, $M_e$ is not recursive. If $T_e$ has no infinite path, then $5^{[\emptyset]} \in M_e$ and then the clauses of type (5) ensure that $M_e = \omega$. Thus $\omega$ is the least model of $P_{h(e)}$ if and only if $T_e$ has no infinite path which proves (b). Furthermore, $M_e$ is recursive iff $T_e$ has no infinite path, which proves (c). □

## 4. The complexity of recursive weakly finite ESB programs

As we saw in Section 2, a recursive weakly finite ESB Horn program $P$ with recursive or r.e. constraints always has least model which is r.e.. Hence the least models of recursive weakly finite ESB Horn programs are much better behaved than the least models of arbitrary recursive ESB Horn programs. In this section, we shall study the complexity of the following question: "When does a recursive weakly finite ESB program have a *recursive* stable model". For example, we shall show that the predicate that a recursive weakly finite ESB Horn program

with recursive constraints has a recursive least model is $\Sigma_3^0$-complete. That is, the set $S$ of all $e$ such that $e$ is a recursive index of a recursive weakly finite ESB Horn program $P$ with recursive constraints where the least model $N^P$ of $P$ is recursive is a complete $\Sigma_3^0$ set. Similarly, we shall show the set $T$ of all $e$ such $e$ is a recursive index of a weakly finite ESB program $P$ with recursive constraints which has a recursive stable model is $\Sigma_3^0$-complete.

The first step in analyzing the complexity of the question of when a recursive weakly finite ESB program $P$ has a recursive stable model is to analyze the complexity of predicates of the form $M \models D$ where $D$ is a constraint, a clause or a program. Our next three lemmas are proved by carefully writing out the definitions of the predicates involved. We will just give the proofs in a few cases. Note that a predicate involving finite, recursive, or r.e. sets is really a predicate on the *indices* for those sets. In particular, suppose a recursive set $M$ has some recursive index $m$ where $\phi_m$ is a total recursive function. For a clause $C$, when we say that the predicate "$M \models C$" is recursive, $\Sigma_1^0$, etc., we mean that there is a recursive ($\Sigma_1^0$, etc.) relation $R$ such that if $m$ is a recursive index of a recursive set $M$ and $c$ is the index of a clause $C$, then $M \models C \iff R(m, c)$. Similarly for a recursive ESB program $P$, when we say that "$M \models P$" is $\Sigma_k^0$, we mean that there is a $\Sigma_k^0$ relation $R$ such that, if $m$ is a recursive index of the recursive set $M$ and $i$ is a recursive index of the program $P$, then $M \models P \iff R(m, i)$. In particular, if either $m$ or $i$ is not the index of a total recursive function, then $R(m, i)$ may be either true or false. It is important to note that we assume that for all clauses $\langle X, \mathcal{R} \rangle^*$, that for each $e \in \mathcal{R}$, $R_e \subset X$ (and similarly for finite or r.e. constraints); otherwise, $P$ is not a program and the question of whether $M \models P$ does not apply.

We will say that $e$ is a recursive index of a recursive ESB program if $R_e$ is an ESB program. Note that predicate "$e$ is a recursive index of an ESB program" is $\Pi_2^0$ since we need only check that $\phi_e$ is a total $\{0, 1\}$-valued function which is $\Pi_2^0$ and that for all $x \in R_e$, $x$ is the code of clause of the proper form. It follows that the $\{e :$ $e$ is a recursive index of an ESB program$\}$ is a $\Pi_2^0$ set. We will say that $e$ is a recursive index of a weakly finite program if $e = [s, t]$ where $s$ is an explicit index of a finite set of constraints and $t$ is a recursive index of a recursive ESB program such that for all clauses $C \in P_t$ such that $C$ involves either recursive or r.e. constraints, $head(C) \in F_s$. Again it is easy to see that the set of indices of recursive weakly finite ESB programs is $\Pi_2^0$. By the same type of reasoning, it is easy to see that for each of the following types of programs, the set of $e$ such that $e$ is a recursive index of an ESB program of this type is also $\Pi_2^0$: (1) programs with finite constraints, (2) Horn programs with finite constraints, (3)

weakly finite programs with recursive constraints, (4) weakly finite Horn programs with recursive constraints, (5) weakly finite programs with r.e. constraints, and (6) weakly finite Horn programs with r.e. constraints.

**Lemma 4.1.** *Suppose that $M$ is a recursive set, $V$ is an r.e. set, $\langle X, \mathcal{F}\rangle^{=}$ and $\langle X, \mathcal{F}\rangle^{\subseteq}$ are finite constraints, $\langle Y, \mathcal{R}\rangle^{=}$ and $\langle Y, \mathcal{R}\rangle^{\subseteq}$ are recursive constraints, and $\langle Z, \mathcal{W}\rangle^{=}$ and $\langle Z, \mathcal{W}\rangle^{\subseteq}$ are r.e. constraints.*

*(a) The predicate $M \models \langle X, \mathcal{F}\rangle^{\subseteq}$ and the predicate $M \models \langle X, \mathcal{F}\rangle^{=}$ are both recursive. The predicate $V \models \langle X, \mathcal{F}\rangle^{\subseteq}$ is $\Sigma^0_1$. The predicate $V \models \langle X, \mathcal{F}\rangle^{=}$ is $\Delta^0_2$, in fact, it is the difference of two $\Sigma^0_1$ predicates.*

*(b) The predicates $M \models \langle Y, \mathcal{R}\rangle^{=}$ and $M \models \langle Y, \mathcal{R}\rangle^{\subseteq}$ are both $\Pi^0_1$. The predicates $V \models \langle Y, \mathcal{R}\rangle^{=}$ and $V \models \langle Y, \mathcal{R}\rangle^{\subseteq}$ are both $\Pi^0_2$.*

*(c) The predicate $M \models \langle Z, \mathcal{W}\rangle^{\subseteq}$ is $\Pi^0_1$. The predicates $M \models \langle Z, \mathcal{W}\rangle^{=}$, $V \models \langle Z, \mathcal{W}\rangle^{=}$, and $V \models \langle Z, \mathcal{W}\rangle^{\subseteq}$ are all $\Pi^0_2$.*

*Proof.* Let $M$ be a recursive set with index $m$, i.e. the characteristic function of $M$ is $\phi_m$. Let $V$ be an r.e. set such that $V = W_a = dom(\phi_a)$.

(a) Let $X$ be a finite set with explicit index $x$ and let $\mathcal{F} = \{e_0, \ldots, e_{n-1}\}$ be a finite family of explicit indices of finite subsets of $X$. Then

$$M \models \langle X, \mathcal{F}\rangle^{\subseteq} \iff (\exists i < n)(\forall u)[u \in F_{e_i} \to u \in R_m \ \&\ u \in F_x]$$

and

$$M \models \langle X, \mathcal{F}\rangle^{=} \iff (\exists i < n)(\forall u)[u \in F_{e_i} \iff u \in R_m \ \&\ u \in F_x].$$

From the index $e_i$, we can extract a list of the elements of $F_{e_i}$, so that the universal quantifier $(\forall u)$ is bounded, which makes both predicates recursive.

For $V$, note that

$$V \models \langle X, \mathcal{F}\rangle^{\subseteq} \iff (\exists i < n)(\forall u)[u \in F_{e_i} \to u \in W_a \ \&\ u \in F_x]$$

and

$$V \models \langle X, \mathcal{F}\rangle^{=} \iff (\exists i < n)(\forall u)[u \in F_{e_i} \iff u \in W_a \ \&\ u \in F_x].$$

Here the quantifier $(\forall u)$ is still bounded, but the predicate "$u \in W_a$ is $\Sigma^0_1$ so the predicate $[u \in F_{e_i} \to u \in W_a \ \&\ u \in F_x]$ is $\Sigma^0_1$ and the predicate $[u \in F_{e_i} \iff u \in W_a \ \&\ u \in F_x]$ is the conjunction of $\Sigma^0_1$ and $\Pi^0_1$ predicate.

For (b), let $Y$ be a recursive set with recursive index $y$ and let $\mathcal{F} = \{e_0, \ldots, e_{n-1}\}$ be a finite family of recursive indices of subsets of $Y$. The analysis in this case is similar to part (a) except that the quantifier $(\forall u)$ is now unbounded.
For example, in the predicate

$$M \models \langle Y, \mathcal{R}\rangle^{\subseteq} \iff (\exists i < n)(\forall u)[u \in R_{e_i} \rightarrow u \in R_m \ \&\ u \in R_y].$$

the $(\forall u)$ quantifier is unbounded so this is $\Pi^0_1$. Similarly, the predicate

$$V \models \langle Y, \mathcal{R}\rangle^{\subseteq} \iff (\exists i < n)(\forall u)[u \in R_{e_i} \rightarrow u \in W_a \ \&\ u \in R_y]$$

is $\Pi^0_2$.

(c) Let $Z$ be an r.e. set with r.e. index $z$ and let $\mathcal{W} = \{e_0, \ldots, e_{n-1}\}$ be a finite family of r.e. indices of subsets of $Z$. Note that by assumption, $W_{e_i} \subseteq W_z$ for all $i < n$, so that the predicate

$$M \models \langle Z, \mathcal{W}\rangle^{\subseteq} \iff (\exists i < n)(\forall u)[u \in W_{e_i} \rightarrow u \in R_m \ \&\ u \in W_z]$$

is equivalent to $M \models \langle Z, \mathcal{W}\rangle^{\subseteq} \iff (\exists i < n)(\forall u)[u \in W_{e_i} \rightarrow u \in R_m]$, which is $\Pi^0_1$. However the predicate

$$M \models \langle Z, \mathcal{F}\rangle^{=} \iff (\exists i < n)(\forall u)[u \in W_{e_i} \iff u \in R_m \ \&\ u \in W_z].$$

will be $\Pi^0_2$. It is also easy to see that the predicates $V \models \langle Z, \mathcal{W}\rangle^{\subseteq}$ and $V \models \langle Z, \mathcal{W}\rangle^{\subseteq}$ are $\Pi^0_2$. □

The following lemmas are now immediate.

**Lemma 4.2.** *Let $M$ be a recursive set, let $V$ be an r.e. set, and let $C$ be a clause of the form (1).*

*(a) If $C$ is a finite clause, then the predicate $M \models C$ is recursive and the predicate $V \models C$ is $\Delta^0_2$.*

*(b) If $C$ is a recursive clause, then the predicate $M \models C$ is $\Delta^0_2$ and the predicate $V \models C$ is $\Delta^0_3$.*

*(c) If $C$ is an r.e. clause, then the predicate $M \models C$ is $\Delta^0_3$ and $V \models C$ is $\Delta^0_3$.*

*(d) Let $C$ be an r.e. clause. Then*

*(i) If $C$ has r.e. equality constraints in the head but not the body, then the predicate $M \models C$ is $\Pi^0_2$,*

*(ii) If $C$ has r.e. equality constraints in the body but not the head, then the predicate $M \models C$ is $\Sigma^0_2$, and*

*(iii) If $C$ does not have any r.e. equality constraints, then the predicate $M \models C$ is a $\Delta^0_2$. In particular, if $C$ is a Horn clause, then $M \models C$ is a $\Delta^0_2$ predicate.* □

Recall that $m$ is a recursive index of a recursive ESB program $P$ is $\phi_m$ is a total recursive function and the set of codes of clauses in $P$ equals $R_m$. Similarly, we say that $e$ is an r.e. index of a r.e. ESB program $P$ if the set of codes of clauses in $P$ equals $W_e$.

**Lemma 4.3.** *Let $M$ be a recursive set and let $V$ be a r.e. set.*

*(a) If $P$ is a recursive or r.e. ESB program with finite constraints, then $M \models P$ is a $\Pi^0_1$ predicate and $V \models P$ is a $\Pi^0_2$ predicate.*

*(b) If $P$ is a recursive ESB program with recursive constraints, then $M \models P$ is a $\Pi^0_2$ predicate and $V \models P$ is a $\Pi^0_3$ predicate.*

*(c) If $P$ is a recursive ESB program with r.e. constraints, then the predicate $M \models P$ is $\Pi^0_3$ and the predicate $V \models P$ is $\Pi^0_3$.*

*(d) If $P$ is a recursive ESB program with r.e. constraints and there is no clause $C$ in $P$ which has an r.e. equality constraint in the body, then the predicate $M \models P$ is $\Pi^0_2$. In particular, if $P$ is a recursive Horn ESB program with r.e. constraints, then $M \models P$ is a $\Pi^0_2$ predicate.* □

*Proof.* The proof relies on the results of Lemma 4.2. We shall shall only give the proof of the first part of (a) as the proof of the rest of the parts is similar. Suppose that $m$ is the recursive index of a recursive ESB program with finite constraints. For any $x \in P$, let $C(x)$ equal the clause coded by $x$. Then

$$M \models P \iff (\forall x)(\phi_m(x) = 1 \to M \models C(x)).$$

Since $M \models C(x)$ is a recursive predicate, it follows that $M \models P$ is $\Pi^0_1$. Similarly, if $e$ is an r.e. index for a r.e. ESB program with finite constraints, then

$$M \models P \iff (\forall x)(x \notin W_e \vee M \models C(x)).$$

Since $x \notin W_e$ is $\Pi^0_1$, it is still the case that $M \models P$ is $\Pi^0_1$. □

To give a more refined analysis for recursive weakly finite programs, let us say that a Horn program $P$ with recursive or r.e. constraints is **n-weakly finite** if there are $n$ different heads of clauses in $P - Fin(P)$.

**Lemma 4.4.** *Let $P$ be an $n$-weakly finite Horn program with recursive or r.e. constraints and with least model $M$. Let $a_1, \ldots, a_n$ be the finite list of indices $a$ such that there is a clause $C$ in $P$ with recursive or r.e. constraints and $head(C) = \langle X, \{a\}\rangle^{\subseteq}$. Let $T_{Fin(P)}$ be the one-step provability operator for $Fin(P)$. For any subset $F$ of $\{a_1, \ldots, a_n\}$, let $W_F = \cup\{X_a : a \in F\}$, where $X_a = F_a$ if $\langle X, \{a\}\rangle$ is a finite constraint, $X_a = R_a$ if $\langle X, \{a\}\rangle$ is a recursive constraint and $X_a = W_a$ if $\langle X, \{a\}\rangle$ is an r.e. constraint. Let $M_F = T_{Fin(P)} \uparrow^\omega (W_F)$. Then*

*(i) $M = T_{Fin(P)} \uparrow^\omega (W_F)$, where $F = \{a_i : X_{a_i} \subset M\}$ and,*

*(ii) if $F$ has minimal cardinality among those subsets of $\{a_1, \ldots, a_n\}$ such that $M = M_F$, then for any $G \subset F$, $\neg(M_G \models P)$.*

*Proof.* (i) Since $W_F \subset M$ and $M$ is closed under $T_{Fin(P)}$, it follows that $M_F = T_{Fin(P)} \uparrow^\omega (W_F) \subseteq M$. On the other hand, we claim that $M_F = T_{Fin(P)} \uparrow^\omega (W_F) \models P$. To see this, let $C = H \leftarrow B$ be any clause of $P$ such that $M_F \models B$. If $C \in Fin(P)$, then $M_F \models H$ since $M_F$ is closed under $T_{Fin(P)}$. If $C \notin Fin(P)$, then $M \models B$ since $M_F \subseteq M$ and $M_F \models B$. But then $H$ must equal $\langle X, \{a\}\rangle^{\subseteq}$ for some $a \in F$ and hence $M_F \models H$. Thus $M \subseteq M_F$ since $M$ is the least model of $P$. Hence $M = M_F$ as claimed.

(ii) This is immediate from our definitions. □

One can then use Lemmas 4.2 and 4.3 to prove the following.

**Theorem 4.5.** *Let $M$ be a recursive set and let $V$ be a r.e. set.*

*(a) If $P$ is a recursive or r.e. ESB Horn program with finite constraints, then the predicates "$M$ is the least model of $P$" and "$V$ is the least model of $P$" are both $\Pi^0_2$.*

*(b) If $P$ is a recursive or r.e. weakly finite ESB Horn program with recursive constraints or with r.e. constraints, then the predicate "$M$ is the least model of $P$" is $\Sigma^0_3$ and the predicate "$V$ is the least model of $P$" is a difference of $\Sigma^0_3$ sets.*

*Proof.* Let $M^P$ denote the unique least model of $P$.

(a) It follows from our remarks above that $M^P = T_P \uparrow^\omega (\emptyset)$ and hence we can find an r.e. index for $M^P$ uniformly from a recursive or r.e. index for $P$. That is, there is a recursive function $\psi$ such that, given a primitive recursive index $e$ for $P$, produces an r.e. index $\psi(e)$ for $M^P$. Thus to check that the recursive set $M$ is the least model of $P$, we need only check whether $M = W_{\psi(e)}$ which is a $\Pi^0_2$ predicate.

(b) Let $\{a_1, \ldots, a_n\}$ be the finite list from Lemma 4.4. Then, by Lemma 4.4, $M^P$ is the least model of $P$ if and only if there exists a finite

$F \subseteq \{a_1, \ldots, a_n\}$ such that $M = M_F$, $M_F \models P$ and, for all $G \subset F$, $\neg(M_G \models P)$. Since $Fin(P)$ is a recursive or r.e. ESB Horn program, we can uniformly find an r.e. index $e_F$ of $M_F$ from $F$ for all $F \subseteq \{a_1, \ldots, a_n\}$. Now the predicates $M = M_F$ and $V = M_F$ are $\Pi^0_2$. Since $M$ is a recursive set, $V$ is an r.e. set and each $M_G$ is a r.e. set, it follows from Lemma 4.3 that $M \models P$ is $\Pi^0_2$, $V \models P$ is $\Pi^0_3$, and $\neg(M_G \models P)$ is $\Sigma^0_3$. Thus the predicate $M$ is the least model of $P$ is $\Sigma^0_3$ and the the $V$ is the least model of $P$ is the conjunction of a $\Sigma^0_3$ and a $\Pi^0_3$ predicate. □

**Theorem 4.6.** *(a) If $P$ is a recursive or r.e. ESB Horn program with finite constraints, then the predicate "the least model of $P$ is recursive" is $\Sigma^0_3$.*

*(b) If $P$ is a recursive or r.e. weakly finite ESB Horn program with recursive constraints or with r.e. constraints, then the predicate "the least model of $P$ is recursive" is $\Sigma^0_3$.*

*Proof.* (a) Let us suppose that $P$ is a recursive or r.e. ESB Horn program with finite constraints. Note that $P$ has a recursive stable model if and only there exists an $m$ such that $\phi_m$ is a total $\{0,1\}$-valued function and $R_m$ is the least model of $P$. The result immediately follows since the predicates "$\phi_m$ is a total {0,1}-valued function" is $\Pi^0_2$ and the predicate "$R_m$ is the least model of $P$" is $\Pi^0_2$ by part (a) of Theorem 4.5.

(b) The proof of part (b) is similar except that we use the results of part (b) of Theorem 4.5. □

**Theorem 4.7.** *(a) The set $T_0$ of all $e$ such that $e$ is a recursive index of a recursive ESB Horn program $P$ with finite constraints such that the least model of $P$ is recursive is a complete $\Sigma^0_3$ set.*

*(b) The set $T_1$ of all $e$ such that $e$ is a recursive index of a recursive weakly finite ESB Horn program with recursive (r.e) constraints such that the least model of $P$ is recursive is a complete $\Sigma^0_3$ set.*

*(c) The set $T_2$ of all $e$ such that $e$ is a r.e. index of a recursive ESB Horn program $P$ with finite constraints such that the least model of $P$ is recursive is a complete $\Sigma^0_3$ set.*

*(d) The set $T_3$ of all $e$ such that $e$ is a r.e. index of a recursive weakly finite ESB Horn program with recursive (r.e) constraints such that the least model of $P$ is recursive is a complete $\Sigma^0_3$ set.*

*Proof.* By Theorem 4.6, $T_i$ is $\Sigma^0_3$ for $i = 0, 1, 2, 3$. For the completeness, will give a reduction of the well-known $\Sigma^0_3$-complete set $Rec = \{e : W_e \text{ is recursive}\}$ to each $T_i$. That is, for $T_0$ and $T_1$, we will define a

recursive function $f$ such that for all $e$, $f(e)$ is the recursive index of a recursive weakly finite ESB Horn program $P_{f(e)}$ with finite constraints such that the least model of $P_{f(e)}$ is Turing equivalent to $\{2a : a \in W_e\}$. Then clearly $W_e$ is recursive if and only if $P_{f(e)}$ has a recursive least model so that *Rec* is one-to-one reducible to both $T_0$ and $T_1$ and hence both $T_0$ and $T_1$ are $\Sigma^0_3$-complete.

Let $W_{e,s}$ be the uniformly computable finite set of elements which are enumerated into $W_e$ by stage $s$. The program $P_{f(e)}$ has two classes of clauses.

(1) $2[2, a, s] + 1 \leftarrow$ for all $a$ and $s$ such that $a \in W_{e,s}$.

(2) $2a \leftarrow\ 2[2, a, s] + 1$ for all $a$ and $s$.

It is easy to see that $P_{f(e)}$ is a recursive ESB Horn program and that the least model $M_e$ of $P_{f(e)}$ equals $\{2a : a \in W_e\} \cup \{2[2, a, s]+1 : a \in W_{e,s}\}$. Hence $M_e$ is recursive if and only if $W_e$ is recursive. Thus $f$ shows that $T_0$ is $\Sigma^0_3$ complete. Since for all $e$, $P_{f(e)}$ is also a recursive weakly finite ESB Horn program with finite (and hence recursive and r.e.) constraints, it also follows that $T_1$ is $\Sigma^0_3$-complete.

It is also clear that there is a recursive function $g$ such that $g(e)$ is an r.e. index of $P_{f(e)}$. Thus $g$ shows that $T_2$ and $T_3$ are $\Sigma^0_3$-complete. □

For general ESB programs, we have the following.

**Theorem 4.8.** *Let $M$ be a recursive set.*

*(a) If $P$ is a recursive ESB program with finite constraints, then the predicate "$M$ is a stable model of $P$" is $\Pi^0_2$.*

*(b) If $P$ is a recursive weakly finite ESB program with recursive constraints or $P$ is a recursive weakly finite ESB program with r.e. constraints which has no clause $C$ with an r.e. equality constraint in the body, then the predicate "$M$ is a stable model of $P$" is $\Sigma^0_3$.*

*(c) If $P$ is a recursive weakly finite ESB program with r.e. constraints, then the predicate "$M$ is a stable model of $P$" is a difference of $\Sigma^0_3$ predicates.*

*Proof.* By definition, $M$ is a stable model of $P$ if and only if $M \models P$ and $M$ is the least model of $\mathbf{NSS}_M(P)$. First observe by Lemma 4.3 that the predicate $M \models P$ is $\Pi^0_1$ if $P$ is a recursive ESB program with finite constraints, is $\Pi^0_2$ if $P$ is a recursive weakly finite program with recursive constraints or $P$ is a recursive weakly finite ESB program with r.e. constraints which has no clause $C \in P$ with an r.e. equality constraint in the body, and is $\Pi^0_3$ if $P$ is a recursive weakly finite program with r.e. constraints. Thus to complete our analysis of the complexity of

$M$ being a stable model, we need only analyze the complexity of the program $\mathbf{NSS}_M(P)$.

Recall that for each clause $C$ in $P$, we eliminate $C$ if $M$ does not satisfy the body of $C$. If $M \models body(C)$ and $head(C) = \langle X, \mathcal{F}\rangle^{\subseteq}$, then for each $e \in F$ such that set coded by index $e$ is contained in $M \cap X$, we add a clause

$$\langle X, \{e\}\rangle^{\subseteq} \leftarrow \langle Y_1, \mathcal{B}_1\rangle^{\subseteq}, \ldots, \langle Y_k, \mathcal{B}_k\rangle^{\subseteq}, \langle Z_1, \mathcal{C}_1\rangle^{\subseteq}, \ldots, \langle Z_l, \mathcal{C}_l\rangle^{\subseteq}. \quad (6)$$

Similarly, if $M \models body(C)$ and $head(C) = \langle X, \mathcal{F}\rangle^{=}$, then for each $e \in F$ such that set coded by index $e$ equals $M \cap X$, we add a clause

$$\langle X, \{e\}\rangle^{\subseteq} \leftarrow \langle Y_1, \mathcal{B}_1\rangle^{\subseteq}, \ldots, \langle Y_k, \mathcal{B}_k\rangle^{\subseteq}, \langle Z_1, \mathcal{C}_1\rangle^{\subseteq}, \ldots, \langle Z_l, \mathcal{C}_l\rangle^{\subseteq}. \quad (7)$$

It easily follows that if $M$ is recursive and $P$ has only finite constraints, then $\mathbf{NSS}_M(P)$ is r.e.. Similarly, if $P$ is a recursive weakly finite program with recursive or r.e. constraints, then $\mathbf{NSS}_M(Fin(P)$ is r.e.

If $C$ is clause with recursive constraints, then the questions of whether $R_e = M \cap X$ or $R_e \subseteq M \cap X$ are $\Pi^0_1$. If $C$ is a clause with r.e. constraints, then the questions of whether $W_e = M \cap X$ or $W_e \subseteq M \cap X$ is $\Pi^0_2$. Thus if $P$ is a recursive weakly finite recursive program, let $\langle X_1, \mathcal{F}_1\rangle^{*_1}, \ldots, \langle X_n, \mathcal{F}_n\rangle^{*_n}$ be the finite set of heads of clauses $\langle X, \mathcal{F}\rangle$ such that there is a clause $C$ in $P$ which contain recursive or r.e. constraints and $head(C) = \langle X, \mathcal{F}\rangle^*$ where $*$ is either $=$ or $\subseteq$. By our conventions, we can effectively find $\langle X_1, \mathcal{F}_1\rangle^{*_1}, \ldots, \langle X_n, \mathcal{F}_n\rangle^{*_n}$ from the recursive index of $P$. Now if $\langle X_i, \mathcal{F}_1\rangle^{*_i}$ is a finite constraint, then we can effectively find all constraints $\langle X_i, \{e\}\rangle^{\subseteq}$ that could be generated by the head $\langle X_i, \mathcal{F}_1\rangle^{*_i}$ in the construction of $\mathbf{NSS}_M(P)$. Now if $\langle X_i, \mathcal{F}_1\rangle^{*_i}$ is a recursive or r.e. constraint, then set of all constraints $\langle X_i, \{e\}\rangle^{\subseteq}$ that could be generated by the head $\langle X_i, \mathcal{F}_1\rangle^{*_i}$ in the construction of $\mathbf{NSS}_M(P)$ is defined by a $\Pi^0_2$ predicate. Now if $P$ is a recursive weakly finite program with recursive constraints or $P$ is a recursive weakly finite ESB program with r.e. constraints which has no clause $C \in P$ with an r.e. equality constraint in the body, then the predicate $M \models body(C)$ is $\Pi^0_1$ by Lemma 4.1. It then follows that the set $D$ consisting of all constraints of the form $\langle X_i, \{e\}\rangle^{\subseteq}$ that are generated by the head of some clause $CinP - Fin(P)$ such that $M \models body(C)$ is a $\Sigma^0_3$ set. Finally, if $P$ is a weakly finite program with r.e. constraints, then the predicate $M \models body(C)$ is $\Pi^0_2$ by Lemma 4.1. so that it still follows that the set $D$ consisting of all constraints of the form $\langle X_i, \{e\}\rangle^{\subseteq}$ that are generated by the head of some clause $C \in P - Fin(P)$ such that $M \models body(C)$, is a $\Sigma^0_3$ set.

(a) Since $\mathbf{NSS}_M(P)$ is a r.e. ESB Horn program, the question of whether $M$ is the least model of $\mathbf{NSS}_M(P)$ is $\Pi^0_2$ by Theorem 4.5.

(b) Suppose that $P$ is a recursive weakly finite program with recursive constraints or that $P$ is a recursive weakly finite ESB program with r.e. constraints which has no clause $C \in P$ with an r.e. equality constraint in the body. Let $\langle X_1, \mathcal{F}_1\rangle^{*_1}, \ldots, \langle X_n, \mathcal{F}_n\rangle^{*_n}$ be the finite set of heads of clauses $\langle X, \mathcal{F}\rangle$ such that there is a clause $C$ in $P$ which contain recursive or r.e. constraints and $head(C) = \langle X, \mathcal{F}\rangle^*$ where $*$ is either $=$ or $\subseteq$. By our conventions, we can effectively find $\langle X_1, \mathcal{F}_1\rangle^{*_1}, \ldots, \langle X_n, \mathcal{F}_n\rangle^{*_n}$ from the recursive index of $P$. By our comments above, $\mathbf{NSS}_M(Fin(P))$ is r.e.. Let $T_{\mathbf{NSS}_M(Fin(P))}$ be the one-step provability operator corresponding to the program $\mathbf{NSS}_M(Fin(P))$. Let $D$ consisting of all constraints of the form $\langle X_i, \{e\}\rangle^{\subseteq}$ that are generated by the head of some clause $C$ with recursive or r.e. constraints. Let $F = \{a_1, \ldots, a_n\}$ denote the set of all $e$ such that there there exists an $i$ such that $\langle X_i, \{e\}\rangle^{\subseteq} \in D$ and let $D_e$ denote the set whose index is $e$. $D$ and $F$ are finite sets but they are only $\Sigma^0_3$ definable. Moreover, for any given $i$, it may be the case that there are infinitely many clauses $C$ in $P$ with $head(C) = \langle X_i, \mathcal{F}_i\rangle^{*_i}$, so that $\mathbf{NSS}_M(P)$ is not necessarily a recursive program. Nevertheless, it still follows as in the proof of Theorem 4.5 that the least fixed point of $\mathbf{NSS}_M(P)$ is of the form

$$M_F = T_{NNS_M(Fin(P)}(\bigcup_{a_i \in F} D_{a_i})$$

for some $F \subseteq \{a_1, \ldots, a_n\}$ where $M_F \models \mathbf{NSS}_M(P)$ and, for all $G \subset F$, $\neg(M_G \models \mathbf{NSS}_M(P))$. Note than since $\mathbf{NSS}_M(Fin(P))$ is r.e., we can find an r.e. index $e_F$ for $M_F$ uniformly in $F$. Now the predicate $M_F = M$ is $\Pi^0_2$. It follows that $M$ is stable model of $P$ if and only if

1. $M \models P$,
2. there is an $x$ such that the finite set $F_x$ with explicit index $x$ satisfies
   a) $F_x \subseteq D$,
   b) $M_{F_x} = M$,
   c) $M \models \mathbf{NSS}_M(P)$, and
   d) For all $G \subset F_x$, $\neg(M_G \models \mathbf{NSS}(P))$.

In this case $M \models P$ is a $\Pi^0_2$ predicate. Now $F_x \subseteq D$ is a $\Sigma^0_3$ predicate and $M_{F_x} = M$ is a $\Pi^0_2$ predicates. Thus we only have consider the complexity of the predicates $M \models \mathbf{NSS}_M(P)$ and $M_G \models \mathbf{NSS}_M(P)$ for $G \subseteq \{a_1, \ldots, a_n\}$. For any clause $C$ such that $M \models body(C)$, it is automatic that $M$ is a model of every clause of $\mathbf{NSS}_M(P)$ generated by $C$. Thus $M$ automatically models $\mathbf{NSS}_M(P)$. Finally, $neg(M_G \models$

$NNS_M(P)$) holds if and only if there exists a clause $C \in P$ such that $M \models body(C)$, $M_G \models body(C)$, and either
(i) $head(C) = \langle X, \mathcal{F}\rangle^{\subseteq}$, there is an $e \in \mathcal{F}$ such that $(D_e \subseteq M \cap X)$, and $D_e \not\subseteq M_G \cap X)$ or
(ii) $head(C) = \langle X, \mathcal{F}\rangle^{=}$, there is an $e \in \mathcal{F}$ such that $(D_e = M \cap X)$ and $D_e \not\subseteq M_G \cap X)$.
The predicates $M \models body(C)$ and $M_G \models body(C)$ are at worst $\Pi^0_2$. Similarly the predicates $(D_e \subseteq M \cap X)$ and $(D_e = M \cap X)$ are $\Pi^0_2$. Finally the predicate $D_e \not\subseteq M_G \cap X)$ are all $\Sigma^0_2$. Thus the predicate $neg(M_G \models NNS_M(P))$ is $\Sigma^0_3$ and, hence, the predicate "$M$ is a recursive stable model of $P$" is $\Sigma^0_3$.

(c) In this case we can use the same analysis as in part (b). The only difference is that $M \models P$ is a $\Pi^0_3$ predicate so that "$M$ is a stable model of $P$" is the conjunction of a $\Pi^0_3$ predicate and $\Sigma^0_3$ predicate. □

The following result immediately follows from Theorem 4.7 and Theorem 4.8.

**Theorem 4.9.** *1. The set of all e such that e is a recursive index of a recursive weakly finite ESB program P with recursive constraints and P has a recursive stable model is $\Sigma^0_3$-complete.*

*2. The set of all e such that e is a recursive index of a recursive weakly finite ESB program P with r.e. constraints such that there is no clause C in P such that C has an r.e. equality constraint in the body and P has a recursive stable model is $\Sigma^0_3$-complete.*

## 5. Conclusions and Further Research

In this paper we defined a natural extension, which we called Extended Set Based (ESB) Logic Programming, of the Set Constraints Logic Programming paradigm introduced by Marek and Remmel [30] in which we could reason about infinite sets. In particular, we considered two basic types of constraints, $\langle X, \mathcal{F}\rangle^{\subseteq}$ and $\langle X, \mathcal{F}\rangle^{=}$ where $\mathcal{F}$ is a finite set of indices of subsets of $X$. We then considered three types of constraints depending on what type of indices were used, (i) finite constraints where $X$ is a finite set and $\mathcal{F}$ is a finite set of explicit indices of finite subsets of $X$, (ii) recursive constraints where $X$ is recursive set and $\mathcal{F}$ is a finite set of recursive indices of recursive subsets of $X$, and (iii) r.e. constraints where $X$ is an r.e. set and $\mathcal{F}$ is finite set of r.e. indices of r.e. subsets of $X$. An ESB program is a set of clauses of the form

$$\langle X, \mathcal{A}\rangle^* \leftarrow \langle Y_1, \mathcal{B}_1\rangle^{\subseteq}, \ldots, \langle Y_k, \mathcal{B}_k\rangle^{\subseteq}, \langle Z_1, \mathcal{C}_1\rangle^{=}, \ldots, \langle Z_l, \mathcal{C}_l\rangle^{=},$$

where $*$ is either $=$ or $\subseteq$.

We then defined natural analogues of Horn programs and stable models for ESB programs. We analyzed the question of when the least model of a recursive ESB Horn program with either finite, recursive or r.e. constraints is recursive. We showed that the question of whether a recursive ESB Horn program has a recursive least model is $\Sigma_3^0$-complete. We also introduced the notion of weakly finite ESB programs which are ESB programs in which the set of heads of clauses that involve either recursive or r.e. constraints is finite and studied the question of when such programs have recursive stable models. We showed, for example, that the question of whether a recursive weakly finite ESB program with recursive constraints has a stable model is also $\Sigma_3^0$-complete.

Our paper represents one way that one can use Logic Programming to reason about infinite sets. The current developments in Computational Knowledge Representation and in particular the research of [36, 30] raises our hopes that with an appropriately chosen extension of Logic Programming, we will be able to effectively reason about stable models of programs involving constraints of the form $\langle X, \mathcal{A}\rangle^{\subseteq}$ and $\langle X, \mathcal{A}\rangle^{=}$ for at least some classes of indices. It should be observed that we are not the only ones interested in the issue of reasoning about infinite sets. For instance, in database community, using entirely different means (quantifier elimination), researchers have shown that one can reduce some queries about infinite sets to queries about finite databases [26]. We believe that more attempts to reason about infinite sets will be made, and and that it will become an important area of Computer Science investigations.

Our formalism allows for reasoning about infinite sets within the broader ASP approach [31, 33]. To make this approach applicable to specific problems requires the ability to find effective indexing of infinite sets. For example, if one wants to analyze voice signals, one would consider sets of points in Euclidean space and graphs of functions in such a space. For various problems in graphics, one may need to develop techniques to approximate the sets occurring in practical applications by means of finite sets or finite sets of equations which are easily manipulated. If one would like to cascade programs, i.e. use the output of programs as inputs to other programs, one might want to "approximate" the infinite stable models of a program. In such situation, we might try to have the stable models of a given program $P$ be approximated by finite models of a (simpler) program $P'$. In general, one needs a way of representing objects or sets by indices which are easily manipulated and for which there are efficient algorithms on indices to make decisions about the various relations on the underlying objects or sets. We have defined one class of programs which involve indices

of infinite sets, effectively decidable weakly finite recursive ESB Horn programs with recursive or r.e. constraints, where one can effectively find an r.e. index of the least model from the index of the program. One can define other abstract sets conditions which allow one to effectively compute with programs involving infinite sets. However, more research needs to be done on effective indexing problems in real application areas if our formalism is to be applicable.

## Acknowledgments

The authors acknowledge partial support of National Science Foundation. The second author has been partly supported by NSF grants IIS 0097278 and 0325063. The third author has been partly supported by NSF grant DMS 0400507. The authors acknowledge helpful remarks of anonymous reviewers. This paper constitutes an extended version of a paper presented at the International Symposium on Mathematics and Artificial Intelligence, Ft. Lauderdale, January 2004, [10]

## References

1. C. Anger, K. Konczak, and T. Linke. `NoMoRe`: A System for Non-Monotonic Reasoning under Answer Set Semantics. *Proceedings of* $6^{th}$ *International Conference on Logic Programming and Nonmonotonic Reasoning*, *Lecture Notes in Computer Science* 2173, pages 406–410. Springer-Verlag, 2001.
2. K. Apt and H.A. Blair. Arithmetical classification of perfect models of stratified programs. *Fundamenta Informaticae*, 12:1 – 17, 1990.
3. Y. Babovich and V. Lifschitz. *Cmodels*, 2002. `http://www.cs.utexas.edu/users/tag/cmodels.html`.
4. C. Baral. *Knowledge Representation, Reasoning and Declarative Problem Solving*, Cambridge University Press, 2003.
5. H.A. Blair, V.W. Marek, and J. Schlipf. The expressiveness of locally stratified programs. *Annals of Mathematics and Artificial Intelligence* 15(2):209–229, 1995.
6. P.A. Bonatti. Reasoning with infinite stable models. *Proceedings of the* $17^{th}$ *International Joint Conference on Artificial Intelligence*, pages 603-610, Morgan Kaufmann, 2001.
7. P.A. Bonatti. Prototypes for Reasoning with Infinite Stable Models and Function Symbols. *Proceedings of the* $6^{th}$ *International Conference on Logic Programming and Nonmonotonic Reasoning Lecture Notes in Computer Science* 2173, pages 416-419, Springer-Verlag, 2001.
8. P.A. Bonatti. Reasoning with infinite stable models. *Artificial Intelligence* 156(1); 75-111, 2004.
9. P. Bruscoli, A. Dovier, E. Pontelli, G. Rossi. Compiling intensional sets in CLP. *Proceedings of* $11^{th}$ *Conference on Logic Programming*, pages 647-661, MIT Press, 1994.

10. D. Cenzer, V.W. Marek and J.B. Remmel. Using logic programs to reason about infinite sets. *Proceedings of the Symposium on Mathematics and Artificial Intelligence* `http:rutcor.rutgers.edu/$\sim$amai/aimath04`.
11. D. Cenzer, V.W. Marek and J.B. Remmel. Compactness property for logic programs. *Forthcoming*.
12. D. Cenzer and J.B. Remmel. Index sets for $\Pi^0_1$ classes. *Annals of Pure and Applied Logic* 93:3-61, 1998.
13. D. Cenzer and J.B. Remmel. $\Pi^0_1$ Classes in Mathematics. *Handbook of Recursive Mathematics*, 623–821. Elsevier, 1999.
14. D. Cenzer, J.B. Remmel and A. Vanderbilt. Locally determined logic programs and recursive stable models. *Annals of Mathematics and Artificial Intelligence* 40:225-262, 2004.
15. P. Cholewiński. Stratified Default Theories. *Proceedings of Computer Science Logic Conference*, CSL'94. *Lecture Notes in Computer Science* 933, pages 456-470, Springer-Verlag, 1995.
16. A. Dovier, E.G. Omodeo, E. Pontelli, G. Rossi. A Logic Programming Language with Finite Sets. *Proceedings of* $8^{th}$ *International Conference on Logic Programming* pages 111-124, MIT Press, 1991.
17. A. Dovier, E.G. Omodeo, E. Pontelli, G. Rossi. flogg: a language for programming in logic with finite sets. *Journal of Logic Programming* 28(1): 1-44, 1996.
18. A. Dovier, E. Pontelli, G. Rossi. Checked Intensional sets in CLP. *Proceedings of International Conference on Logic Programming*, *Lecture Notes on Computer Science* 2916, pages 284-299, Springer-Verlag, 2003.
19. T. Eiter, G. Gottlob and H. Veith. Modular Logic Programs and General Quantifiers. *Proceedings of the* $4^{th}$ *International Conference on Logic Programming and Nonmonotonic Reasoning*, *Lecture Notes in Computer Science* 1265, pages 290–309, Springer-Verlag, 1997.
20. T. Eiter, N. Leone, C. Mateis, G. Pfeifer, and F. Scarcello. The KR System dlv: Progress Report, Comparisons, and Benchmarks. In *Proceedings of Sixth International Conference on Principles of Knowledge Representation and Reasoning (KR-98)*, pages 406–417, Morgan Kaufmann, 1998.
21. M. Gelfond and V. Lifschitz. The stable semantics for logic programs. *Proceedings of* $5^{th}$ *International Conference on Logic Programming* pages 1070–1080, MIT Press, 1988.
22. M. Gelfond and N. Leone. Logic Programming and Knowledge Representation – A-Prolog perspective. *Artificial Intelligence Journal* 138:3–38, 2002.
23. E. Goldberg, Y. Novikov. BerkMin: a Fast and Robust SAT-Solver. *Proceedings of Conference on Design, Automation and Test in Europe*, pages 142–149, 2002.
24. P. G. Hinman. *Recursion-Theoretic Hierarchies*. Springer-Verlag, 1978.
25. J. Jaffar and M. Maher. Constraint logic programming: A survey. *Journal of Logic Programming*, 19-20:503–581, 1994.
26. G.M. Kuper, L. Libkin and J. Paradaens. *Constraint Databases*, Springer-Verlag, 2000.
27. V. Lifschitz and H. Turner. Splitting a Logic Program. *Proceedings of International Conference on Logic Programming, ICLP'94*, pages 23–37, 1994.

28. F. Lin and Y. Zhao. ASSAT: Computing answer sets of a logic program by SAT solvers. *Proceedings of* $18^{th}$ *National Conference on Artificial Intelligence*, pages 112–117, Morgan Kaufmann, 2002.
29. W. Marek, A. Nerode, and J. B. Remmel. The stable models of predicate logic programs. *Journal of Logic Programming*, 21(3):129–154, 1994.
30. V.W. Marek and J.B. Remmel. Set Constraints in Logic Programming. *Proceedings of* $7^{th}$ *International Conference on Logic Programming and Nonmonotonic Reasoning*, *Lecture Notes in Computer Science* 2923, pages 167–179. Springer-Verlag, 2004.
31. V.W. Marek and M. Truszczynski. Stable models and an alternative logic programming paradigm. In: *The Logic Programming paradigm*, pages 375–398, Springer-Verlag, 1999.
32. M.W. Moskewicz, C.F. Magidan, Y. Zhao, L. Zhang, and S. Malik. Chaff: engineering an efficient SAT solver. *Proceedings of Design Automation Conference* pages 530–535. 2001.
33. I. Niemelä. Logic programs with stable model semantics as a Constraint Programming paradigm. *Annals of Mathematics and Artificial Intelligence* 25:241–273, 1999.
34. I. Niemelä, P. Simons, and T. Soininen. Stable model semantics of weight constraint rules. In *Proceedings of* $5^{th}$ *International Conference on Logic Programming and Nonmonotonic Reasoning*, *Lecture Notes in Computer Science* 1730, pages 317–331. Springer-Verlag, 1999.
35. I. Niemelä and P. Simons. Extending the smodels system with cardinality and weight constraints. In *Logic-Based Artificial Intelligence*, pages 491–521. Kluwer Academic Publishers, 2000.
36. P. Simons, I. Niemelä, and T. Soininen. Extending and implementing the stable model semantics. *Artificial Intelligence*, 138:181–234, 2002.
37. R.I. Soare. *Recursively Enumerable Sets and Degrees*. Springer Verlag, 1987.
38. C. Spector. Inductively defined sets of natural numbers. In *Infinitistic Methods*, pages 97–102. Pergamon Press, 1961.
39. T. Syrjänen. Manual of Lparse version 1.0, `http://saturn.tcs.hut.fi/Software/smodels`
40. T. Syrjänen. Omega-restricted logic programs. *Proceedings of* $6^{th}$ *International Conference on Logic Programming and Nonmonotonic Reasoning*, *Lecture Notes in Computer Science* 2173, pages 267–279, Springer-Verlag, 2001.

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