Set Based Logic Programming

H.A. Blair[1], V.W. Marek[2], and J.B. Remmel[3]

[1] Department of Electrical Engineering and Computer Science, Syracuse University, Syracuse, NY 13244 *
[2] Department of Computer Science, University of Kentucky, Lexington, KY 40506 **
[3] Departments of Mathematics and Computer Science, University of California at San Diego, La Jolla, CA 92903 ***

Abstract. In a previous paper [BMR01], the authors showed that the mechanism underlying Logic Programming can be extended to handle the situation where the atoms are interpreted as subsets of a given space X. The view of a logic program as a one-step consequence operator along with the concepts of supported and stable model can be transferred to such situations. In this paper, we show that we can further extend this paradigm by creating a new one-step consequence operator by composing the old one step consequence operator with a monotonic idempotent operator (miop) in the space of all subsets of X, 2^X. We call this extension *set based logic programming*. We show that such a set based formalism for logic programming naturally supports a variety of options. For example, if the underlying space has a topology, one can insist that the new one-step consequence operator always produces a closed set or always produces an open set. The flexibility inherent in the semantics of set based logic programs is due to both the range of natural choices available for specifying the semantics of negation, as well as the role of monotonic idempotent operators (miops) as parameters in the semantics. This leads to a natural type of polymorphism for logic programming, i.e. the same logic program can produce a variety of outcomes depending on the miop associated with the semantics. We develop a general framework for set based programming involving miops. Among the applications, we obtain integer-based representations of real continuous functions as stable models of a set based logic program.

1 Introduction

In [BMR01], the authors developed an extension of the logic programming paradigm which can directly reason about regions in space and time as might be required, for example, for applications in graphics, image compression, or job scheduling. Thus instead of having the intended underlying universe be the Herbrand base of the program, one replaces the underlying Herbrand universe

* email: blair@ecs.syr.edu
** email: marek@cs.uky.edu
*** email: jremmel@ucsd.edu

by some fixed space X and has each atom of the language of the program specify a subset of X, i.e. an element of the set 2^X.

If we reflect for a moment on the basic aspects of logic programming with an Herbrand model interpretation, a slight change in our point of view shows that it is natural to interpret atoms as subsets of the Herbrand base. In normal logic programming, we determine the truth value of an atom p in an Herbrand interpretation I by declaring $I \models p$ if and only if $p \in I$. However, this is equivalent to defining the *sense*, $[\![p]\!]$, of a ground atom p to be the set $\{p\}$ and declaring that $I \models p$ if and only if $[\![p]\!] \subseteq I$. By this simple move, we have permitted ourselves to interpret the sense of an atom as a subset of a set X rather than the literal atom itself in the case where X is the Herbrand base of the language of the program.

More generally, in [BMR01] we showed that it is a natural step to take the *sense* $[\![p]\!]$ of ground atom p to be a subset of some fixed nonempty set X and to define a $I \subseteq X$ to be a model of p, written $I \models p$, if and only if $[\![p]\!] \subseteq I$. This approach to setting up a semantics makes available multiple truth values, intentional constructs, and interpreted relationships among the elements and subsets of X. Observe that the assignment of a *sense* to ground atoms is intrinsically intentional. Interpreted relationships among the elements and subsets of X allow the programs that use this approach, which we called *spatial logic programing* in [BMR01], to serve as front-ends for existing systems and still have a seamless model theoretic semantics for the system as a whole.

It turns out that if the underlying space X has structure such as a topology or an algebraic structure such as a group, ring, field, or vector space, then a number of other natural options present themselves. For example, if we are dealing with a topological space, one can construct a new one-step consequence operator T by composing the one-step consequence operator for spatial logic programming with an operator that produces, for example, the topological closure of a set, or the interior of a set. In such a situation, we can ensure that the new one-step consequence operator T always produces a closed set or always produces an open set. Similarly, if the underlying space is a vector space, one might construct a new one-step consequence operator T by composing the one-step consequence operator for spatial logic programming with an operator that produces the smallest subspace containing a set, the *span* operator, or with an operator that produces the smallest convex closed set containing a set, the *convex closure* operator, so that one can ensure that the new one-step consequence operator T always produces a subspace or always produces a convex closed set. More generally, we say that an operator $O : 2^X \to 2^X$ is *monotonic* if for all $Y \subseteq Z \subseteq X$, we have $O(Y) \subseteq O(Z)$ and we say that O is *idempotent* for all $Y \subseteq X$, $O(O(Y)) = O(Y)$. Notice that each of the operators described above, *closure*, *interior*, *span* and *convex-closure*, are monotonic idempotent operators. We call a monotonic idempotent operator a *miop* (pronounced "my op"). We say that a set Y is *closed* with respect to miop O iff $Y = O(Y)$. This terminology is synonymous with being a fixed point of O, but in many of the situations that we have in mind, such as having the operator always produce a subgroup of a group, a subfield of field, an ideal in Boolean algebra, etc., closure seems

a more natural way to think. By composing the one-step consequence operator for spatial logic programs with the operator O, we can ensure that the resulting one-step consequence operator always produces a fixed point of O. We can then think of the operator O as a parameter. This naturally leads us to a situation where we have a natural polymorphism for set based logic programming. That is, one can use the same logic program to produce stable models with different properties depending on how the operator O is chosen.

Moreover, in such a setting, one also has a variety of options for how to interpret negation. In normal logic programming, a model M satisfies $\neg p$ if $p \notin M$. From the spatial logic programming point of view, when p is interpreted as a singleton $\{p\}$, this would be equivalent to saying that M satisfies $\neg p$ if (i) $\{p\} \cap M = \emptyset$, or (equivalently) (ii) $\{p\} \not\subseteq M$. When the sense of p is a set with more than one element, it is easy to see that saying that M satisfies $\neg p$ if $[\![p]\!] \cap M = \emptyset$ (strong negation) is different from saying that M satisfies $\neg p$ if $[\![p]\!] \not\subseteq M$ (weak negation). As we shall see, this leads to two natural interpretations of the negation symbol which are compatible with the basic logic programming paradigm. When the underlying space has a miop cl, one can get even more subsidiary types of negation by taking M to satisfy $\neg p$ if $cl([\![p]\!]) \cap M = cl(\emptyset)$, or by taking M to satisfy $\neg p$ if $cl([\![p]\!]) \not\subseteq M$.

The main contribution of this paper is to extend the spatial logic programming paradigm of [BMR01] to the full set based logic programming paradigm with associated miops. In particular, the main objectives of this paper are the following:

1. We show that logic programming with stable semantics lifts to lattices different from the power set lattice of the usual Herbrand interpretations[4]. It turns out that stable semantics generalizes to at least two different constructions in the general case, with subsidiary variations obtained by using closures under monotonic idempotent operators.
2. We show how several standard mathematical constructions such as separation properties, complementary subspaces, and continuity can be cast in the form of such generalized stable models relative to one of our two basic generalizations of stable semantics. This extends various representability results presented in [Ba03,GL02] to the continuous setting.

The ultimate purpose of this paper is to provide the foundations and basic techniques for crafting applications in the *answer set paradigm*, as described in [MT99,Nie99] and then [ASP01,Ba03], where topological, linear algebraic, or similar constructs associated with a given application are either necessary or at least natural. The use of miops allows for operations on what we call the *senses* of ground atoms that materially contribute to determining the models of programs. The expressive power of miops allows us to capture functions and relations intrinsic to the domain of a spatial logic program, but independent of the program. It is this feature that permits set based logic programs to seamlessly

[4] Other lattices were considered in [BS89], but stable semantics or the closure of models under miops were not studied.

serve as front-ends to other systems. Miops play the role of back-end, or "behind-the-scenes", procedures and functions.

The outline of this paper is as follows. In sections 2 and 3, we shall briefly review the spatial logic programming paradigm as given in [BMR01]. These sections relate to the objective (1) described above. In section 4, we introduce and develop the role of monotonic idempotent operators. In particular we give several examples where the same program can give different results depending on which miop and/or negation operator we use. Finally, in section 5, we discuss further work and extensions of the set based framework.

2 Spatial Logic Programs: syntax and semantics

Before giving the general definitions of our formalism for set based logic programming with miop operators, we shall first recall the relevant aspects of spatial logic programs as developed in [BMR01]. Fundamentally, the *set based* logic programs to be introduced in section 4 are spatial logic programs interleaved with miops.

The syntax of a spatial logic program is based on the syntax of the formulas of what we define as a *spatially augmented first-order logic*. Spatial augmentation is an intentional notion. The syntax of spatial programs will essentially be the syntax of DATALOG programs with negation.

Definition 1. A **spatially augmented first-order language** (**spatial language**, for short) $\mathcal{L}$ is a triple $(L, X, [\![\cdot]\!])$, where
(1) L is a language for first-order predicate logic (without function symbols other than constants),
(2) X is a nonempty (possibly infinite) set, called the **interpretation space**, and
(3) $[\![\cdot]\!]$ is a mapping from the ground atoms of L to the power set of X, called the *sense assignment*. If p is a ground atom, then $[\![p]\!]$ is called the *sense* of p.

At first glance, the mapping $[\![\cdot]\!]$ and the interpretation space X might seem to properly belong in the semantics of spatially augmented languages. However, these languages are to be thought of as having a fixed partial interpretation; hence the interpretation space and sense assignment should be fixed by the language analogously to fixing the interpretation of the equality symbol in ordinary first-order languages to be the identity relation.

Definition 2. A **spatial logic program** has three components.
1) The language $\mathcal{L}$ which includes the interpretation space and the sense assignment.
2) The IDB (**Intentional Database**): A finite set of program clauses, each of the form $A \leftarrow L_1, \ldots, L_n$, where each L_i is a *literal*, i.e. an atom or the negation of an atom, and A is an atom.
3) The EDB (**Extensional Database**): A finite set of ground atoms.

Given a spatial logic program P, the *Herbrand base* of P is the Herbrand base of the smallest spatial language over which P is a spatial logic program.

For the rest of this section, we shall assume that the classes of spatial logic programs that we consider are always over a language for first-order logic L with no function symbols except constants, and a fixed set X.

Informally, we think of the Herbrand universe Λ_L of the underlying language L, i.e. the set of constant symbols of L, as being a set of indices which we may employ to suit whatever purpose is at hand. We let HB_L denote the Herbrand base of L, i.e. the set of ground atoms of L. We omit the subscript L when the context is clear. Let X be a nonempty set, 2^X be the powerset of X, and $[\![\cdot]\!] : \mathrm{HB}_L \longrightarrow 2^X$. An *interpretation* I of the spatial language $\mathcal{L} = (L, X, [\![\cdot]\!])$ is a subset of X.

We note that sense assignments $[\![\cdot]\!]$ can also be used to partition the ground atoms into multiple sorts. For example, suppose X is the disjoint union of X_1 and X_2 and HB_L is the disjoint union of A_1 and A_2. Then we can achieve multiple sorts by letting $[\![\cdot]\!]$ be such that $[\![p]\!] \subseteq X_i$ for $p \in A_i$, $i = 1, 2$. However, we shall not pursue such uses in the paper.

Suppose that we are given a satisfaction relation between interpretations and ground literals, i.e. the criterion for how it is that an interpretation I satisfies a ground literal L, denoted by $I \models_{[\![\cdot]\!]} L$. We can then extend the given satisfaction relation $\models_{[\![\cdot]\!]}$ to all formulas generated from literals by $\wedge$, $\vee$ and $\rightarrow$ in the usual fashion. More formally, because of the diversity of notions of negation available, we will employ a mapping α_I corresponding to each $I \subseteq X$ from the set of sentences, i.e. the set of all formulas without free occurrences of variables, to three truth values $\mathbf{t}$, $\mathbf{f}$, and $\bot$. (Conventionally, we call the truth-value $\bot$ *bottom*.) We first define, for a given interpretation I, α_I on the ground literals and then extend α_I from ground literals to all other sentences that are not negations. For α_I to be well-defined, we must avoid situations where there are atoms A such that both $I \models_{[\![\cdot]\!]} A$ and $I \models_{[\![\cdot]\!]} \neg A$, as is the case for strong negation whenever $[\![A]\!] = \emptyset$.

For each ground atom A,

$$\alpha_I(A) = \begin{cases} \mathbf{t} & \text{if } I \models_{[\![\cdot]\!]} A \\ \mathbf{f} & \text{if } I \models_{[\![\cdot]\!]} \neg A \\ \bot & \text{otherwise.} \end{cases}$$

For each *negative literal* $\neg A$,

$$\alpha_I(\neg A) = \begin{cases} \mathbf{f} & \text{if } I \models_{[\![\cdot]\!]} A \\ \mathbf{t} & \text{if } I \models_{[\![\cdot]\!]} \neg A \\ \bot & \text{otherwise.} \end{cases}$$

We adopt a three-valued logic with truth values $\{\mathbf{t}, \mathbf{f}, \bot\}$. (Every sentence, i.e. the set of all formulas without free occurrences of variables, will turn out to have a truth-value other than $\bot$ if every ground atom has truth-value other than $\bot$.)

We adopt a standard set of *strong* interpretations of the 3-valued connectives.

$$\alpha_I(A \wedge B) = \begin{cases} \mathbf{t} & \text{if } \alpha_I(A) = \alpha_I(B) = \mathbf{t} \\ \mathbf{f} & \text{if } \alpha_I(A) = \mathbf{f} \text{ and } \alpha_I(B) \neq \bot \\ \mathbf{f} & \text{if } \alpha_I(B) = \mathbf{f} \text{ and } \alpha_I(A) \neq \bot \\ \bot & \text{otherwise.} \end{cases}$$

Similarly,

$$\alpha_I(A \vee B) = \begin{cases} \mathbf{f} & \text{if } \alpha_I(A) = \alpha_I(B) = \mathbf{f} \\ \mathbf{t} & \text{if } \alpha_I(A) = \mathbf{t} \\ \mathbf{t} & \text{if } \alpha_I(B) = \mathbf{t} \\ \bot & \text{otherwise.} \end{cases}$$

For the three-valued conditional:

$$\alpha_I(A \rightarrow B) = \begin{cases} \mathbf{t} & \text{if } \alpha_I(B) = \mathbf{t} \\ \mathbf{t} & \text{if } \alpha_I(A) = \mathbf{f} \\ \mathbf{f} & \text{if } \alpha_I(A) = \mathbf{t} \text{ and } \alpha_I(B) = \mathbf{f} \\ \bot & \text{otherwise.} \end{cases}$$

The quantifiers are interpreted schematically.

$$\alpha_I(\forall x\, \varphi(x)) = \begin{cases} \mathbf{t} & \text{if } \alpha_I(\varphi(e)) = \mathbf{t}, \text{ for all constants } e \\ \mathbf{f} & \text{if } \alpha_I(\varphi(e)) = \mathbf{f}, \text{ for some constant } e \\ \bot & \text{otherwise.} \end{cases}$$

$$\alpha_I(\exists x\, \varphi(x)) = \begin{cases} \mathbf{t} & \text{if } \alpha_I(\varphi(e)) = \mathbf{t}, \text{ for some constant } e \\ \mathbf{f} & \text{if } \alpha_I(\varphi(e)) = \mathbf{f}, \text{ for all constants } e \\ \bot & \text{otherwise.} \end{cases}$$

Finally, we define $I \models_{[\![\cdot]\!]} \varphi$ if and only if $\alpha_I(\varphi) = \mathbf{t}$. A *model* (not necessarily stable) of a spatial program is a model of the set of all formulas in the EDB and IDB. Again, α, and consequently $\models_{[\![\cdot]\!]}$ is not defined for negations other than negative literals.

For the rest of this paper, we shall assume for each atom p that $I \models_{[\![\cdot]\!]} p$ if and only if $[\![p]\!] \subseteq I$. Thus a model of a program must contain the sense of every ground instance of each atom in the EDB. Also note that if every ground literal in the language of program P has a non-bottom truth-value with respect to interpretation $\mathcal{I}$, then every clause in P will also have a non-bottom truth-value with respect to interpretation $\mathcal{I}$.

3 The consequence operator and stable models

The following operator generalizes the one-step consequence-operator of ordinary logic programs to spatial logic programs. Given a spatial program $\mathcal{P}$ with IDB P, let P' be the set of ground instances of the clauses in P and let

$$T_P(I) = I_1 \cup I_2$$

where

$$I_1 = \bigcup\{[\![A]\!] \mid A \leftarrow L_1, \ldots, L_n \ \in P', I \models_{[\![\cdot]\!]} L_i, i = 1, \ldots, n\} \text{ and}$$
$$I_2 = \bigcup\{[\![A]\!] \mid A \text{ is a ground atom in the EDB of } \mathcal{P}\}.$$

A *supported model* of $\mathcal{P}$ is a model of $\mathcal{P}$ that is a fixed point of $T_{\mathcal{P}}$. It should be noted that since clause bodies can contain negative literals, the supported models of $\mathcal{P}$ are partly dependent on the criteria used to determine the validity of negative literals in an interpretation.

A spatial logic program is *Horn* if its IDB is Horn. Our definitions allow us to generalize the familiar characterization of the least model of ordinary Horn programs. We iterate T_P in the usual manner:

$$\begin{aligned} T_P \uparrow^0 (I) &= I \\ T_P \uparrow^{\alpha+1} (I) &= T_P(T_P \uparrow^\alpha (I)) \\ T_P \uparrow^\lambda (I) &= \bigcup_{\alpha<\lambda} \{T_P \uparrow^\alpha (I)\},\ \lambda \text{ limit} \end{aligned}$$

It is clear that T_P is monotonic if P is a Horn program. Thus, the following result follows from the Tarski fixed point theorem.

Theorem 1. The least model of spatial Horn program P exists, is supported, and is given by $\mathbf{T}_P \uparrow^\alpha (\emptyset)$ for the least ordinal α at which a fixed point is obtained.

We note, however, that if the Herbrand universe of a spatial program is infinite (contains infinitely many constants) then, unlike the situation with ordinary Horn programs, T_P will not in general be upward continuous. That is, consider the following example.

Example 1. To specify a spatial program, we must specify the language, EDB and IDB. Let $\mathcal{L} = (L, X, [\![\cdot]\!])$ where L has four unary predicate symbols: p, q, r and s, and countably many constants $e_0, e_1, \ldots,$. X is the set $\mathbf{N} \bigcup \{\mathbf{N}\}$ where $\mathbf{N}$ is the set of natural numbers, $\{0, 1, 2, \ldots\}$. $[\![\cdot]\!]$ is specified by $[\![q(e_n)]\!] = \{0, \ldots, n\}$, $[\![p(e_n)]\!] = \{0, \ldots, n+1\}$, $[\![r(e_n)]\!] = \mathbf{N}$, $[\![s(e_n)]\!] = \{\mathbf{N}\}$.

The EDB is empty and the IDB is: $q(e_0) \leftarrow$, $p(X) \leftarrow q(X)$, and $s(e_0) \leftarrow r(e_0)$.

Now, after ω iterations upward from the empty interpretation, $r(e_0)$ becomes satisfied. One more iteration is required to reach an interpretation that satisfies $s(e_0)$, where the least fixed point is attained. △

What is different about the ascending iteration of T_P from the ordinary situation in logic programming is that, in the spatial case, the senses of ground body atoms can be satisfied by the union of the senses of infinitely many ground clause heads without any finite collection of these clause heads uniting to satisfy the body atom. But, if there are only finitely many primitive atoms, i.e. if the

Herbrand *universe* of the program is finite or the sense of each atom is a finite set, then this source of upward discontinuity vanishes. The proof of upward continuity is essentially the same in that case as that for ordinary Horn programs.

Theorem 2. *The least model of spatial Horn program P exists, is supported, and is given by $\mathbf{T}_P \uparrow^\omega (\emptyset)$, if the set of primitive ground atoms in the Herbrand base of P is finite or the sense of each atom is a finite set.*

In spatial logic programs, we allow clauses whose ground instances are of the following form:

$$A \leftarrow B_1, \ldots, B_n, \neg C_1, \ldots, \neg C_m. \tag{1}$$

In [BMR01], we defined two types of stable models depending upon how we interpreted the satisfaction relation for ground atoms. That is, consider the following two different definitions of the satisfaction relation $I \models_{[\![\cdot]\!]}$ for negative atoms $\neg p$.

$$I \models^s_{[\![\cdot]\!]} \neg p \iff [\![p]\!] \cap I = \emptyset \tag{2}$$

and

$$I \models^w_{[\![\cdot]\!]} \neg p \iff [\![p]\!] \not\subseteq I = \emptyset. \tag{3}$$

We shall refer to $I \models^s_{[\![\cdot]\!]}$ as *strong* negation and $I \models^w_{[\![\cdot]\!]}$ as *weak* negation. In both cases, if p is an atom, then we define

$$I \models^s_{[\![\cdot]\!]} p \iff I \models^w_{[\![\cdot]\!]} p \iff [\![p]\!] \subseteq I. \tag{4}$$

We can then define the stable model semantics relative to $I \models^a_{[\![\cdot]\!]}$ for $a \in \{s, w\}$ for such programs as follows. For any given set $J \subseteq X$, we define Gelfond-Lifschitz transform [GL88] of a program P, $GL^a_{J,[\![]\!]}(P)$, in two steps. First we consider all ground instances C of clauses in P as in (1). If it is not the case that $J \models^a_{[\![\cdot]\!]} \neg C_i$ for some C_i in the body of C, then the we eliminate clause C. If not, then we replace C by the Horn clause

$$A \leftarrow B_1, \ldots, B_n. \tag{5}$$

The $GL^a_{J,[\![\cdot]\!]}(P)$ consists of $EDB(P)$ plus the sets of all Horn clauses produced by this two step process. Thus $GL^a_{J,[\![\cdot]\!]}(P)$ is a Horn program so that the least fixed point of $T_{GL^a_{J,[\![\cdot]\!]}(P)}$ is defined. Then we say that J is an *a-stable model* of P if and only if J equals the least model of $GL^a_{J,[\![\cdot]\!]}(P)$.

Example 2. Let $\mathcal{L} = (L, X, [\![\cdot]\!])$ where L has five atoms x, y, z, m, and n. Let $X = \{1, 2, 3, 4\}$ and let $[\![\cdot]\!]$ is specified by

$[\![x]\!] = \{1\}$,
$[\![y]\!] = \{1, 2\}$,
$[\![z]\!] = \{2, 3\}$,
$[\![m]\!] = \{1, 2, 3, 4\}$, and
$[\![n]\!] = \{1, 2\}$.

The EDB is empty and the IDB is:
$x \leftarrow$
$y \leftarrow \neg m$
$z \leftarrow \neg n$.

Now it is easy to see that the only possible s-stable models or w-stable models must be unions of the senses of some of the heads of the clauses of P so that there are only three possible candidates of either s-stable or w-stable models for P, namely, $M_1 = \{1\}$, $M_2 = \{1, 2\}$, or $M_3 = \{1, 2, 3\}$.

Now it is easy to see that for strong negation, $M_i \not\models^s_{[\![\cdot]\!]} \neg m$ and $M_i \not\models^s_{[\![\cdot]\!]} \neg n$ for all $i \in \{1, 2, 3\}$. If follows that for each $i \in \{1, 2, 3\}$, $GL^a_{M_i,[\![\cdot]\!]}(P) = x \leftarrow$ so that the least model of $GL^a_{M_i,[\![\cdot]\!]}(P)$ is $\{1\}$. Thus only M_1 is an s-stable model of P.

For weak negation, it is easy to see that $M_i \models^s_{[\![\cdot]\!]} \neg m$ for all i and $M_i \models^s_{[\![\cdot]\!]} \neg n$ if and only if $i = 1$. Thus

$$GL^a_{M_1,[\![\cdot]\!]}(P) = \{x \leftarrow, y \leftarrow, z \leftarrow\}$$

and the least model of $GL^w_P(M_1)$ is $\{1, 2, 3\}$ so that M_1 is not w-stable. For $i = 2, 3$,

$$GL^a_{M_i,[\![\cdot]\!]}(P) = \{x \leftarrow, y \leftarrow\}$$

and the least model of $GL^w_P(M_i)$ is $\{1, 2\}$. Thus M_2 is a w-stable model of P and M_3 is not w-stable. This example shows that the notions of s-stable models and of w-stable models do not always coincide. △

Theorem 3. *For any spatial logic program P and any $a \in \{s, w\}$,*

1. *$I \subseteq X$ is a model of $\mathcal{P}$ iff $T_P(I) \subseteq I$ and*
2. *I is a-stable with respect to $\mathcal{P}$ implies that I is supported with respect to $\mathcal{P}$.*

Proof: The proofs of the two parts of the above proposition proceed exactly as they do for ordinary programs after noting that for *any* $I \subseteq X$, an element x of $T_P(I)$ is an element of the sense of at least one of the clause heads in $\mathcal{P}$ whose body is satisfied by I. □

The next theorem shows the relationship between stable models of a spatial program, and a natural topology induced by a spatial language on its interpretation space.

Theorem 4. *If $\mathcal{L}$ is a spatially augmented first-order language, then the set of senses of the ground atoms form a subbasis of a topology in which all supported models, a fortiori all a-stable models for $a \in \{s, w\}$, of all spatial programs over $\mathcal{L}$ are open subsets of the interpretation space.*

Proof: First we define the desired topology. Recall, [Ku66], that a topology in a set X is a family $\mathcal{O} \subseteq 2^X$ such that $\mathcal{O}$ contains the empty set and the entire X, and $\mathcal{O}$ is closed under finite intersections and arbitrary unions. Here is how

we define $\mathcal{O}$. We first define a *basis* $\mathcal{B}$ for $\mathcal{O}$. Let $\mathcal{A}$ be the set of ground atoms in $\mathcal{L}$.

$$\mathcal{B} = \{Y \subseteq X \mid \text{for some finite } \Gamma \subseteq \mathcal{A},\ Y = \bigcap_{p \in \Gamma} [\![p]\!]\}.$$

The topology $\mathcal{O}$ is the closure of $\mathcal{B}$ under arbitrary unions:

$$\mathcal{O} = \{Y \subseteq X \mid \text{for some subset } \mathcal{B}' \text{ of } \mathcal{B},\ Y = \bigcup \mathcal{B}'\}.$$

It immediately follows that $\mathcal{O}$ is closed under finite intersections and arbitrary unions.

Next let us observe that if Y is an open set in our topology associated with the program, and if Y is *any* set, then $T_P(Y)$ is open (for it is union of senses of intentional atoms). In particular fixed points of the operator T_P are open, and so also a-stable models are open. □

We will call the topology given by the previous theorem the *Herbrand topology*. This topology has a utility in finding stable models. Ordinarily one expects to recover a guess for a stable model as the least fixed point of the Gelfond-Lifschitz transform determined by the guess. The previous theorem allows one to recover merely the interior of the guess, or equivalently, confine one's guesses to open sets. In the next section, where we incorporate miops into the one-step consequence operator of a program, we can achieve even greater selectivity of stable models.

4 Set Based Logic Programming with Miops

In this section, we shall introduce miops on the underlying intentional space X of a family of logic programs and show how we can extend the spatial logic programming paradigm of the previous section to incorporate miops. We shall call this combination of spatial logic programming with miops *set based logic programming.*

4.1 Operators and stable models

Let us suppose that the underlying intentional space X is either $\mathbf{R}^n$ or $\mathbf{Q}^n$ were $\mathbf{R}$ is the reals and $\mathbf{Q}$ is the rationals. Then X is a topological vector space under the usual topology so that we have a number of natural miop operators:

1. $op_{id}(A) = A$, i.e. the identity map is simplest miop operator,
2. $op_c(A) = \bar{A}$ where $\bar{A}$ is the smallest closed set containing A,
3. $op_{int}(A) = int(A)$ where $int(A)$ is the interior of A,
4. $op_{convex}(A) = K(A)$ where $K(A)$ is the convex closure of A, i.e. the smallest set $K \subseteq X$ such that $A \subseteq K$ and whenever $x_1, \ldots, x_n \in K$ and $\alpha_1, \ldots, \alpha_n$ are elements of the underlying field ($\mathbf{R}$ or $\mathbf{Q}$) such that $\sum_{i=1}^n \alpha_i = 1$, then $\sum_{i=1}^n \alpha_i x_i$ is in K, and
5. $op_{subsp}(A) = (A)^*$ where $(A)^*$ is the subspace of X generated by A.

We should note that (5) is a prototypical example if we start with an *algebraic* structure. That is, we can let $op_{substr}(A) = (A)^*$ where $(A)^*$ is the substructure of X generated by A. For example,

(a) if X is a group, we can let $op_{subgrp}(A) = (A)^*$ where $(A)^*$ is the subgroup of X generated by A,
(b) if X is a ring, we can let $op_{subrg}(A) = (A)^*$ where $(A)^*$ is the subring of X generated by A,
(c) if X is a field, we can let $op_{subfld}(A) = (A)^*$ where $(A)^*$ is the subfield of X generated by A,
(d) if X is a Boolean algebra, we can let $op_{subalg}(A) = (A)*$ where $(A)^*$ is the subalgebra of X generated by A or we can let $op_{ideal}(A) = Id(A)$ where $Id(A)$ is the ideal of X generated by A, and
(e) if X is a partially ordered set, we can let $op_{lideal}(A) = Lid(A)$ where $Lid(A)$ is the lower order ideal of X (that is, the least subset of X containing A and closed under predecessors) generated by A.

Now let us suppose that we are given a miop $op^+ : 2^X \rightarrow 2^X$ and Horn set based logic program P over X. Recall that a set based logic program is *Horn* if its IDB is Horn. Then we can further generalize the one-step consequence-operator of ordinary logic programs with respect to 2-valued logic to spatial logic programs relative to miop operator op^+ as follows. First for any atom A and $I \subseteq X$, we say that $I \models_{[\![\cdot]\!],op^+} A$ if and only if $op^+([\![A]\!]) \subseteq I$. Then given a spatial program $\mathcal{P}$ with IDB P, let P' be the set of ground instances of a clauses in P and let

$$T_{P,op^+}(I) = op^+(I_1 \cup I_2)$$

where
$I_1 = \bigcup\{[\![A]\!] \mid A \leftarrow L_1, \ldots, L_n \ \in P', I \models_{[\![\cdot]\!],op^+} L_i, i = 1, \ldots, n\}$.
$I_2 = \bigcup\{[\![A]\!] \mid A$ is a ground atom in the EDB of $\mathcal{P}\}$. We then say that a *supported model relative to* op^+ of $\mathcal{P}$ is a fixed point of T_{P,op^+}.

We iterate T_{P,op^+} according to the following.

$$\begin{aligned} T_{P,op^+} \uparrow^0 (I) &= I \\ T_{P,op^+} \uparrow^{\alpha+1} (I) &= T_{P,op^+}(T_{P,op^+} \uparrow^\alpha (I)) \\ T_{P,op^+} \uparrow^\lambda (I) &= op^+(\bigcup_{\alpha<\lambda} \{T_{P,op^+} \uparrow^\alpha (I)\}),\ \lambda \text{ limit} \end{aligned}$$

Again it is easy to see that if P is a Horn spatial logic program and op^+ is a miop, then T_{P,op^+} is monotonic. Thus just like in the case a spatial logic programs, we have the following theorem.

Theorem 5. Given a miop op^+, the least model of a Horn set based logic program P exists and is closed under op^+ , is supported relative op^+, and is given by $\mathbf{T}_{P,op^+} \uparrow^\alpha (\emptyset)$ for the least ordinal α at which a fixed point is obtained.

Next we consider how we should deal with negation in the setting of miop operators. Suppose that we have a miop operator op^- on the underlying space

X. We do not require that op^- is the same as that miop op^+ but it may be. Our goal is to define two different satisfaction relations for negations of atoms relative to the miop operator op^- that correspond to the strong and weak satisfaction relations for negative atoms described in Section 2 above[5]. However, we can make the following more general definition.

Definition 3. Suppose that $\mathcal{P}$ is spatial logic program over X. Let $\mathcal{R}$ be any binary relation between subsets of X. Then, given any ground atom A and set $J \subseteq X$, then we say $J \models_{[\![\cdot]\!],op^+,\mathcal{R}} \neg A$ if and only if $\mathcal{R}(J, [\![A]\!])$.

The preceding definition is too general to play much of a role in a programming paradigm because it strips negation of negation's intended meaning except through special cases of the choice of $\mathcal{R}$, but there is still a point to it. That is, the Gelfond-Lifschitz transform works by marking atoms for special evaluation; it does not care what the mark means to achieve its fundamental objective which is to ensure that if I is a supported model of the Gelfond-Lifschitz transform with respect to I, then I is a supported model of P.

We will specialize $\mathcal{R}$ to two cases of interest for developing set based logic programming before moving on to fundamental canonical examples involving vector spaces and continuous functions, but first we give our main generalization of the one-step consequence operator and develop stable-model semantics.

Definition 4. Let op^+ be a miop and $\mathcal{R}$ be a binary relation between subsets of X. Then for any subset I of X, we say that I satisfies a positive literal A if

$$I \models_{[\![\cdot]\!],op^+,\mathcal{R}} A \text{ iff } op^+([\![A]\!]) \subseteq I$$

and I satisfies a negative literal $\neg A$ if

$$I \models_{[\![\cdot]\!],op^+,\mathcal{R}} \neg A \text{ iff } \mathcal{R}(I, [\![A]\!]).$$

Recall from Section 2 that whenever the satisfaction relation is defined for all ground literals we may extend it to all formulas.

Let

$$T_{P,op^+,\mathcal{R}}(I) = op^+(\bigcup_{A \in \Gamma} [\![A]\!])$$

where

$$\Gamma = \{A \mid A \leftarrow L_1, \ldots, L_n \in \text{ground}(P), I \models_{[\![\cdot]\!],op^+,\mathcal{R}} L_i, i = 1, \ldots, n\}$$
$$\bigcup \{A : A \text{ is in } EDB(P)\}.$$

We say that I is a *supported* model of P *relative to* op^+ and $\mathcal{R}$ iff I is a fixed point of $T_{P,op^+,\mathcal{R}}$ and $I \models P$.

Note that the definition of $T_{P,op^+,\mathcal{R}}$ agrees with that of T_{P,op^+} for Horn programs P. We have the following basic essentially classical result.

[5] Lifschitz [Li94] observed that different modalities, thus different operators, can be used to evaluate positive and negative part of bodies of clauses of normal programs.

Theorem 6. *Suppose that we are given X, a miop op^+ on X, and a binary relation $\mathcal{R}$ over 2^X. Then if $I \subseteq X$,*

$$I \models_{[\![\cdot]\!],op^+,\mathcal{R}} P \text{ iff } T_{P,op^+,\mathcal{R}}(I) \subseteq I$$

Note that we have defined the satisfaction of a positive literal A by a set I by the criterion that $op^+([\![A]\!]) \subseteq I$ rather than merely that $[\![A]\!] \subseteq I$. The following example shows why. Suppose that we defined satisfaction of positive literals by the weaker criterion that $I \models A$ iff $[\![A]\!] \subseteq I$.

Example 3. For the program P consisting of the single clause $p \leftarrow q$ take $X = 2^{\{p,q\}}$ with $[\![p]\!] = \{p\}$, $[\![q]\!] = \{q\}$, and let $op^+(\emptyset) = \emptyset$, $op^+(\{p\}) = op^+(\{q\}) = \{q\}$, $op^+(\{p,q\}) = \{p,q\}$. The choice of $\mathcal{R}$ is immaterial. For $I = \{q\}$, $T_{P,op^+,\mathcal{R}}(I) = op^+([\![p]\!]) = I$, but $I \not\models_{[\![\cdot]\!],op^+,\mathcal{R}} p$ since $[\![q]\!] \subseteq I$ while $[\![p]\!] \not\subseteq I$. △

The Example 3 indicates there would be substantial difficulties in generalizing results from logic programming to set based logic programming if we had used the weaker criteria. In effect, we have used a miop to alter the semantics of the spatially augmented first-order language L in the background of our discussion to be compatible with the miop $op+$ to be used with the set based programs over L by having the miop-closure of the sense of an atom be contained in a miop-closed set in order for the set to satisfy the atom. However because we use the definition that $I \models_{[\![\cdot]\!],op^+} A$ if and only if $op^+([\![A]\!]) \subseteq I$, we can set the senses of ground atoms at our convenience in applications, and the difficulties of the previous example vanish. This move resolves the difficulties in generalizing results from logic programming to set based logic programming and does so without requiring the programmer to arrange the atoms in his programs under the assumption that their senses are closed under op^+. The miop should be part of the inference engine, while premises (i.e. atoms) ideally shouldn't have to respect anything in the inference engine beyond syntax.

Definition 5. A set based logic program P consists of clauses of the form in (1). For any given set $J \subseteq X$, we define $GL_{J,[\![\cdot]\!],op^+,\mathcal{R}}(P)$, the Gelfond-Lifschitz transform of a program P with respect to miop op^+ and binary relation $\mathcal{R}$ on 2^X, in two steps. First we consider all ground instances of clauses C in P. If for some i, it is *not* the case that $J \models_{[\![\cdot]\!],op^+,\mathcal{R}} \neg C_i$, then the we eliminate clause C. Otherwise we replace C by the Horn clause

$$A \leftarrow B_1, \ldots, B_n. \tag{6}$$

The $GL_{J,[\![\cdot]\!],op^+,\mathcal{R}}(P)$ consists of $EDB(P)$ together with the set of all Horn clauses produced by this two step process.

Note that since $GL_{J,[\![\cdot]\!],op^+,\mathcal{R}}(P)$ is a Horn set based logic program, the least model of $GL_{J,[\![\cdot]\!],op^+,\mathcal{R}}(P)$ is defined. We can then define the stable model semantics for a set based logic program P over X relative to a miop op^+ on X and binary relation $\mathcal{R}$ on 2^X.

Definition 6. J is a *stable* model of P *relative* to op^+ and $\mathcal{R}$ iff J is the least fixed point of $T_{GL_{J,[\![\cdot]\!],op^+,\mathcal{R}}(P),op^+,\mathcal{R}}$.

We then have the following result.

Theorem 7. *Suppose that $\mathcal{P}$ is set based logic program over X, op^+ is a miop on X, and $\mathcal{R}$ is a binary relation on 2^X. Assume J is closed relative to op^+, i.e., $op^+(J) = J$. Then, if J is a supported model of $GL_{J,[\![\cdot]\!],op^+,\mathcal{R}}(P)$, then J is a supported model of P relative to op^+ and $\mathcal{R}$.*

Corollary 1. *If J is a stable model of P relative to op^+ and $\mathcal{R}$, then J is a supported model of P relative to op^+ and $\mathcal{R}$.*

For programming purposes, we are interested in special cases of the relation $\mathcal{R}$ on 2^X associated with a second miop, op^-. Here we do not assume that op^- is identical with op^+, but $op^- = op^+$ is allowed. Specifically, in what we shall call the strong semantics, we take $\mathcal{R}_1(J, K)$ iff $J \cap op^-(K) = op^-(\emptyset)$. For the weak semantics, we take $\mathcal{R}_2(J, K)$ iff $op^-(K) \not\subseteq J$. More formally, we make the following definition.

Definition 7. Suppose that $\mathcal{P}$ set based logic program over X and op^+ and op^- are miops on X. Let $a \in \{s, w\}$.

(I) Given any atom A and set $J \subseteq X$, then we say $J \models^a_{[\![\cdot]\!],op^+,op^-} A$ if and only if $op^+([\![A]\!]) \subseteq J$.

(II)$_s$ Given any atom A and set $J \subseteq X$, then we say $J \models^s_{[\![\cdot]\!],op^+,op^-} \neg A$ if and only if $op^-(A) \cap J = op^-(\emptyset)$.

(II)$_w$ Given any atom A and set $J \subseteq X$, then we say $J \models^w_{[\![\cdot]\!],op^+,op^-} \neg A$ if and only if $op^-(A) \not\subseteq J$.

The two types of satisfaction relations for negative literals immediately yield two types of supported models based on two types of one-step consequence operators, T^s_{P,op^+,op^-} and T^w_{P,op^+,op^-}, two types of Gelfond-Lifschitz transform, $GL^s_{J,[\![\cdot]\!],op^+,op^-}$ and $GL^w_{J,[\![\cdot]\!],op^+,op^-}$, and two types of stable model semantics.

Next we give a simple example to show that there is a difference between s-stable and w-stable models.

Example 4. Suppose that the underlying space $X = \mathcal{R}^2$ is the real plane. Our program will have two atoms $\{a, b\}, \{c, d\}$ where a, b, c and d are reals. We let $[a, b]$ and $[c, d]$ denote the line segments connecting a to b and c to d respectively. We let sense of the these atoms be the corresponding subsets, i.e. we let $[\![(\{a, b\}]\!] = \{a, b\}$ and $[\![\{c, d\}]\!] = \{c, d\}$. We let $op^+ = op^- = op_{convex}$. The consider the following program $\mathcal{P}$.

(1) $\{a, b\} \leftarrow \neg\{c, d\}$
(2) $\{c, d\} \leftarrow \neg\{a, b\}$

There are four possible candidate for stable models in this case, namely (i) $\emptyset$, (ii) $[a,b]$, (iii) $[c,d]$, and (iv) $op_{convex}\{a,b,c,d\}$. Let us recall that $op_{convex}(X)$ is the convex closure of X which, depending on a, b, c, and d may be either a quadrilateral, triangle, or a line segment.

If we are considering s-stable models where $J \models^{s}_{[\![\cdot]\!],op^+,op^-} \neg C$ if and only if $op^-(C) \cap J = op^-(\emptyset) = \emptyset$, then the only case where there are stable models if $[a,b]$ and $[c,d]$ are disjoint in which (ii) case and (iii) are s-stable models.

If we are considering w-stable models where $J \models^{w}_{[\![\cdot]\!],op^+,op^-} \neg C$ if and only if $op^-(C) \not\subseteq J$, then there are no w-stable models if $[a,b] = [c,d]$, (ii) is a w-stable model if $[a,b] \not\subseteq [c,d]$, (iii) is w-stable model if $[c,d] \not\subseteq [a,b]$ and (ii) and (iii) are w-stable models if neither $[a,b] \subseteq [c,d]$ nor $[c,d] \subseteq [a,b]$. $\triangle$

In the following subsections, 4.2-4.4, we shall give three examples to show how the stable models of a given spatial logic program can vary depending on how we define op^+ and op^-. We note that in the case where $op^- = op_{id}$ and the sense of any atom A such that $\neg A$ appears in $\mathcal{P}$ is a singleton, then there is no difference between s-stable and w-stable models. *Our examples in the next three subsections will all have this property so that we will not distinguish between s-stable and w-stable models.*

4.2 Separating sets and polymorphism

In this section, we shall give a number of example where set based logic programming yields natural polymorphisms. That is, suppose that that the underlying space S is the n-dimensional vector space $V = \mathbf{Q}^n$ with the usual topology. Let $\mathbf{0}$ denote the zero vector of V. Suppose A and B are subsets of V. Our idea is construct a program whose stable models correspond to *separating sets* S for A and B, that is sets S such that S is closed relative to op^+, $A \subseteq S$ and $S \cap B = op^-(\emptyset)$. We shall see that by picking the miop operators op^+ and op^- appropriately we can have a single spatial logic program P whose stable models have a variety of properties.

Formally, we shall assume that the underlying first order language has constant symbols a for each $a \in V$ and it has three unary predicate symbols S, $\overline{S}$ and A. Thus the ground atoms of the underlying Herbrand Base are all of the form $S(a)$, $\overline{S}(a)$ and $A(a)$ for some $a \in V$. We shall think of the interpretation space X as the set

$$X = \{S(a) : a \in V\} \cup \{\overline{S}(a) : a \in V\} \cup \{A(a) : a \in V\}.$$

The sense of any ground atom $S(a)$, $\overline{S}(a)$ and $A(a)$ will be just $\{S(a)\}$, $\{\overline{S}(a)\}$ and $\{A(a)\}$ respectively. That is: $[\![S(a)]\!] = \{S(a)\}$, $[\![\overline{S}(a)]\!] = \{\overline{S}(a)\}$, and $[\![A(a)]\!] = \{A(a)\}$. Let $op^- = op_{id}$ so that $op^-(\emptyset) = \emptyset$.

Now let us suppose that we are given three miop operators $op_S, op_{\overline{S}}, op_A$ on V. Then we can define a miop operator op^+ on X by the following:

$$op^+(T) = \{S(a) : a \in op_S(\{y \in V : S(y) \in T\})\} \cup \{\overline{S}(a) : a \in op_{\overline{S}}(\{y \in V : \overline{S}(y) \in T\})\} \cup \{A(a) : a \in op_A(\{y \in V : A(y) \in T\})\}. \quad (7)$$

The intuition here is that suppose we want for any stable model M of our program to have the property that $V_{S,M} = \{a : M \models S(a)\}$ is a subspace of V. Then we need only take $op_S = op_{subsp}$ because the fact that M is a fixed point of op^+ will automatically ensure that that $V_{S,M}$ is a subspace. Similarly if we want for any stable model M of our program to have the property that $V_{\overline{S},M} = \{a : M \models \bar{S}(a)\}$ is a subspace of V, then we need only take $op_{\bar{S}} = op_{subsp}$ because the fact that M is a fixed point of op^+ will automatically ensure that that $V_{\bar{S},M}$ is a subspace.

Now let us consider the following program $\mathcal{P}$, consisting of five sets of clauses.

(1) $S(a) \leftarrow$ (for all $a \in A$)
(2) $\overline{S}(b) \leftarrow$ (for all $b \in B$)
(3) $A(\mathbf{0}) \leftarrow S(x), \overline{S}(x), \neg A(\mathbf{0})$
(4) $S(x) \leftarrow \neg\overline{S}(x)$
(5) $\overline{S}(x) \leftarrow \neg S(x)$

We note that when we ground $\mathcal{P}$, the clauses of type (3), (4) and (5) will generate the following sets of ground clauses.

(3)′ $A(\mathbf{0}) \leftarrow S(v), \overline{S}(v), \neg A(\mathbf{0})$ (for all $v \in V$)
(4)′ $S(v) \leftarrow \neg\overline{S}(v)$ (for all $v \in V$)
(5)′ $\overline{S}(v) \leftarrow \neg S(v)$ (for all $v \in V$)

Before proceeding we should make an observation about the clauses of type (3)′ under our assumption that $op_A = op^- = op_{id}$. That is, if $op^- = op_{id}$, then $op^-([\![A(\mathbf{0})]\!]) = \{A(\mathbf{0})\}$. Now if $\{A(\mathbf{0})\} \subseteq M$, then it is not the case that $M \models^s_{[\![\cdot]\!],op^+,op^-} \neg A(\mathbf{0})$ nor is it the case that $M \models^w_{[\![\cdot]\!],op^+,op^-} \neg A(\mathbf{0})$. Note that every clause of $ground(\mathcal{P})$ which has $A(\mathbf{0})$ in the head has $\neg A(\mathbf{0})$ in the body. We claim that no matter how we define op_S and $op_{\overline{S}}$, it will be that case that any s-stable or w-stable model M of $\mathcal{P}$ will have $\{a : A(a) \in M\} = \emptyset$. That is, since the only clauses which have an $A(v)$ in the head come from the clauses of type (3)', it automatically follows that it must be the case that $\{a : A(a) \in M\}$ is either equal to $op_A(\emptyset) = \emptyset$ or $op_A(\{A(\mathbf{0})\}) = \{A(\mathbf{0})\}$. But it cannot be that $A(\mathbf{0}) \in M$ since otherwise all the clauses of type (3)' will be eliminated when we take $GL^s_{M,[\![\cdot]\!],op^+,op^-}(P)$ or $GL^w_{M,[\![\cdot]\!],op^+,op^-}(P)$ and hence there would be no way to generate $A(\mathbf{0})$ by iterating $T_{GL^s_{M,[\![\cdot]\!],op^+,op^-}(P),op^+,op^-}$ or $T_{GL^w_{M,[\![\cdot]\!],op^+,op^-}(P),op^+,op^-}$ starting at the empty set. Thus it must be that $\{a : A(a) \in M\} = \emptyset$. But then the effect of the clauses of type (3)′ is to say that it is impossible that both $S(v)$ and $\overline{S}(v)$ are elements of an a-stable model M of $\mathcal{P}$ for $a \in \{s, w\}$. Thus the effect of the clauses of type (3)′ is to say that

in any s-stable model or w-stable model M where $op_A = op^- = op_{id}$, the sets $\{a : S(a) \in M\}$ and $\{a : \overline{S}(a) \in M\}$ are disjoint.

Next it is easy to see that the clauses of type $(4)'$ and $(5)'$ ensure that that for any s-stable model or w-stable model M of $\mathcal{P}$, it is the case that $\{a : S(a) \in M\} \cup \{a : \overline{S}(a) \in M\} = V$. Similarly the clauses of type (1) and type (2) ensure that $A \subseteq \{a : S(a) \in M\}$ and $B \subseteq \{a : \overline{S}(a) \in M\}$. Thus it follows that no matter how we define op_S and $op_{\overline{S}}$, a s-stable model or w-stable model M of $\mathcal{P}$ must be of the form

$$M_C = \{a : S(a) \in C\} \cup \{a : \overline{S}(a) \in V - C\} \tag{8}$$

where $C \subseteq V$, $A \subseteq C$, $op_S(C) = C$, $B \subseteq V - C$ and $op_{\overline{S}}(V - C) = V - C$. Thus for any s-stable model or w-stable model M of $\mathcal{P}$, the set $\{a : S(a) \in M\}$ is a separating set for A and B.

This given, it is see that we have the following result which characterizes the set of a-stable models M_C as in (8) for $a \in \{s, w\}$ which can occur as an a-stable model of P assuming that $op_A = op^- = op_{id}$.

Proposition 1. *Suppose that $op_A = op^- = op_{id}$ and $a \in \{s, w\}$, then the following hold.*

1. *If $op_S = op_{\overline{S}} = op_{id}$, then M_C is an a-stable model of P iff $A \subseteq C$ and $B \subseteq V - C$.*
2. *If $op_S = op_c$ and $op_{\overline{S}} = op_{int}$, then M_C is an a-stable model of P iff C is a closed set such that $A \subseteq C$ and $B \subseteq V - C$.*
3. *If $op_S = op_{int}$ and $op_{\overline{S}} = op_c$, then M_C is an a-stable model of P iff C is an open set such that $A \subseteq C$ and $B \subseteq V - C$.*
4. *If $op_S = op_{\overline{S}} = op_{conv}$, then M_C is an a-stable model of P iff $K(C) = C$, $K(V - C) = V - C$, $A \subseteq C$ and $B \subseteq V - C$* [6].
5. *If $op_S = op_{subsp}$ and $op_{\overline{S}} = op_{id}$, then M_C is an a-stable model of P iff C is a subspace of V such that $A \subseteq C$ and $B \subseteq V - C$.*
6. *If $op_S = op_{id}$ and $op_{\overline{S}} = op_{subsp}$, then M_C is an a-stable model of P iff $V - C$ is a subspace of V such that $A \subseteq C$ and $B \subseteq V - C$.*

Now let us suppose that we are given a subspace $Y \subseteq X$. Then we can modify the program P to obtain a program R whose a-stable models for $a \in \{s, w\}$ produce separating sets over Y, i.e. all a-stable models of R are of the form

$$M_{C,D} = \{S(a) : a \in C\} \cup \{\overline{S}(a)a \in D\} \cup \{Y(a) : a \in A\}$$

where $C \cap D = Y$ and $C \cup D = V$. That is, extend the language so that we have an additional unary predicate symbol Y and assume that $[\![Y(a)]\!] = \{Y(a)\}$ for all $a \in V$. Define the miop operator $op_Y = op_{id}$. Then we can define a miop operator op^+ on X as by defining op^+ so that

[6] Recall the classical Convex Separation Theorem of Stone: if A and B are disjoint convex subsets of V, then there is a set C such that C and $V - C$ are convex subsets of V such that $A \subseteq C$ and $B \subseteq V - C$.

$$op^+(T) = \{S(a) : a \in op_S(\{y \in V : S(y) \in T\})\} \cup$$
$$\{\overline{S}(a) : a \in op_{\overline{S}}(\{y \in V : \overline{S}(y) \in T\})\} \cup$$
$$\{A(a) : a \in op_A(\{y \in V : A(y) \in T\})\} \cup$$
$$\{Y(a) : a \in op_Y(\{y \in V : Y(y) \in T\})\}. \quad (9)$$

Now let R be the program that results from P by adding the following clauses

(6.1) $Y(a) \leftarrow$ (for all $a \in Y$)
(7.1) $S(x) \leftarrow Y(s)$
(8.1) $\overline{S}(x) \leftarrow Y(s)$,

and by replacing clause (3) by

(3.1) $A(\mathbf{0}) \leftarrow S(x), \overline{S}(x), \neg Y(x), \neg A(\mathbf{0})$.

Then we can use the same type of analysis that we used for P to prove the following.

Proposition 2. *Suppose that $op_Y = op_A = op^- = op_{id}$ and $a \in \{s, w\}$, then the following hold.*

1. *If $op_S = op_{\overline{S}} = op_{id}$, then $M_{C,D}$ is an a-stable model of R if and only if $A \subseteq C$, $B \subseteq D$, $C \cap D = Y$, and $C \cup D = V$.*
2. *If $op_S = op_c$ and $op_{\overline{S}} = op_{int}$, then $M_{C,D}$ is an a-stable model of R if and only if C is a closed set and $A \subseteq C$, $B \subseteq D$, $C \cap D = Y$, and $C \cup D = V$.*
3. *If $op_S = op_{int}$ and $op_{\overline{S}} = op_c$, then $M_{C,D}$ is an a-stable model of R if and only if C is an open set and $A \subseteq C$, $B \subseteq D$, $C \cap D = Y$, and $C \cup D = V$.*
4. *If $op_S = op_{\overline{S}} = op_{conv}$, then $M_{C,D}$ is an a-stable model of R if and only if $K(C) = C$, $K(V - C) = V - C$, $A \subseteq C$, $B \subseteq D$, $C \cap D = Y$, and $C \cup D = V$.*
5. *If $op_S = op_{subsp}$ and $op_{\overline{S}} = op_{id}$, then $M_{C,D}$ is an a-stable model of R if and only if C is a subspace of V and $A \subseteq C$, $B \subseteq D$, $C \cap D = Y$, and $C \cup D = V$.*
6. *If $op_S = op_{id}$ and $op_{\overline{S}} = op_{subsp}$, then $M_{C,D}$ is and a-stable model of R if and only if D is a subspace of V and $A \subseteq C$, $B \subseteq D$, $C \cap D = Y$, and $C \cup D = V$.*

4.3 Complementary subspaces

In this example, we modify the previous program so that s-stable models and w-stable models are determined by a pair of subspaces U and W such that $A \subseteq U$, $B \subseteq W$, and U and V are complementary subspaces of V, that is, $U \cap W = \{\mathbf{0}\}$ and $op_{subsp}(U \cup W) = V$. To this end we add two more predicates $T, \overline{T}$ and define $[\![T(v)]\!] = \{T(v)\}$ and $[\![\overline{T}(v)]\!] = \{\overline{T}(v)\}$ for all $v \in X$. Next consider the program $\mathcal{Q}$ which consists of the clauses of $\mathcal{P}$ from the previous example enlarged by the following set of clauses:

(6) $T(b) \leftarrow$ (for all $b \in B$,)
(7) $\overline{T}(a) \leftarrow$ (for all $a \in A$,)
(8) $A(\mathbf{0}) \leftarrow T(x), \overline{T}(x), \neg A(\mathbf{0})$,
(9) $T(x) \leftarrow \neg\overline{T}(x)$
(10) $\overline{T}(x) \leftarrow \neg T(x)$.

Then as before we shall assume $op_A = op^- = op_{id}$ and define

$$\begin{aligned} op^+(R) = \{S(a) : a \in op_S(\{y \in V : S(y) \in R\})\} \cup \\ \{\overline{S}(a) : a \in op_{\overline{S}}(\{y \in V : \overline{S}(y) \in R\})\} \cup \\ \{T(a) : a \in op_T(\{y \in V : T(y) \in R\})\} \cup \\ \{\overline{T}(a) : a \in op_{\overline{T}}(\{y \in V : \overline{T}(y) \in R\})\} \cup \\ \{A(a) : a \in op_A(\{y \in V : A(y) \in T\})\}. \quad (10) \end{aligned}$$

Then if we let $op_S = op_T = op_{subsp}$ and $op_{\overline{S}} = op_{\overline{T}} = op_{id}$, then we can apply the reasoning to prove parts 5 and 6 of the Proposition 1 to show every a-stable model of $\mathcal{Q}$ for $a \in \{s, w\}$ relative to op^+ must be of the form

$$\begin{aligned} M_{U,W} = \{S(v) : v \in U\} \cup \{\overline{S}(v) : v \in V - U\} \\ \cup \{T(v) : v \in W\} \cup \{\overline{T}(v) : v \in V - W\}. \end{aligned}$$

where U and W are subspaces of V such that $A \subseteq U$, $B \subseteq V - U$, $A \subseteq V - W$, and $B \subseteq W$. Thus we need only add some clauses which ensure that U and W are complementary subspaces of V. We add three more predicates C, $\overline{C}$ and the equality predicate $(=)$, where $op_C = op_{subsp}$ and $op_= = op_{\overline{C}} = op_{id}$. Now, let us consider the following clauses (11)-(18):

(11) $A(\mathbf{0}) \leftarrow C(x), \overline{C}(x), \neg A(\mathbf{0})$,
(12) $C(x) \leftarrow \neg\overline{C}(x)$,
(13) $\overline{C}(x) \leftarrow \neg C(x)$,
(14) $A(\mathbf{0}) \leftarrow \overline{C}(x), \neg A(\mathbf{0})$,
(15) $C(x) \leftarrow S(x)$,
(16) $C(x) \leftarrow T(x)$,
(17) $= (v, v) \leftarrow$ (for all $v \in V$)
(18) $A(\mathbf{0}) \leftarrow S(x), T(x), \neg(x = \mathbf{0}), \neg A(\mathbf{0})$.

By our previous analysis, clauses (11), (12) and (13) ensure that in a stable model M the set $E = \{v \in V : C(v) \in M\}$ is a subspace and $V - E = \{v \in V : \overline{C}(v) \in M\}$. Clause (14) then ensures that in a stable model $V - E = \{v \in V : \overline{C}(v) \in M\} = \emptyset$ and hence it must be the case that $E = \{v \in V : C(v) \in M\} = V$. However the only way that we can generate in E is via applications the clauses of the form (15) and (16) so that in a stable model, we must have $op_{subsp}(U \cup W) = E = V$. Finally the clauses of the form (17) and (18) ensure that $U \cap W = \{\mathbf{0}\}$.

Thus if we assume $op_A = op^- = op_{id}$ and

$$\begin{aligned} op^+(R) = \{S(a) : a \in op_S(\{y \in V : S(y) \in R\})\} \cup \\ \{\overline{S}(a) : a \in op_{\overline{S}}(\{y \in V : \overline{S}(y) \in R\})\} \cup \\ \{T(a) : a \in op_T(\{y \in V : T(y) \in R\})\} \cup \\ \{\overline{T}(a) : a \in op_{\overline{T}}(\{y \in V : \overline{T}(y) \in R\})\} \cup \\ \{C(a) : a \in op_C(\{y \in V : C(y) \in R\})\} \cup \\ \{\overline{C}(a) : a \in op_{\overline{C}}(\{y \in V : \overline{C}(y) \in R\})\} \cup \\ \{A(a) : a \in op_A(\{y \in V : A(y) \in R\})\}, \end{aligned} \quad (11)$$

then every s-stable model or w-stable model of $\mathcal{Q}$, must be of the

$$\begin{aligned} M_{U,W,C} = \{S(v) : v \in U\} \cup \{\overline{S}(v) : v \in V - U\} \\ \cup \{T(v) : v \in W\} \cup \{\overline{T}(v) : v \in V - W\} \\ \cup \{C(v) : v \in V\}. \end{aligned}$$

Thus we have the following result.

Proposition 3. *Assume $op_A = op^- = op_{id}$ and*

$$\begin{aligned} op^+(R) = \{S(a) : a \in op_S(\{y \in V : S(y) \in R\})\} \cup \\ \{\overline{S}(a) : a \in op_{\overline{S}}(\{y \in V : \overline{S}(y) \in R\})\} \cup \\ \{T(a) : a \in op_T(\{y \in V : T(y) \in R\})\} \cup \\ \{\overline{T}(a) : a \in op_{\overline{T}}(\{y \in V : \overline{T}(y) \in R\})\} \cup \\ \{C(a) : a \in op_C(\{y \in V : C(y) \in R\})\} \cup \\ \{\overline{C}(a) : a \in op_{\overline{C}}(\{y \in V : \overline{C}(y) \in R\})\} \cup \\ \{A(a) : a \in op_A(\{y \in V : A(y) \in R\})\}. \end{aligned} \quad (12)$$

Then $M_{U,W,C}$ is an a-stable model of the program $\mathcal{Q}$ for $a \in \{s, w\}$ if and only if $A \subseteq U$, $B \subseteq W$, and U and V are complementary subspaces of V.

4.4 Continuous real functions

One of our original motivations for developing set based logic programming with miops was to have a framework were one could reason about multiple agents where any given agent A can pass information to other agents. In particular, if agent A is operating in a topological space like R and R^n, then he might want to pass information to another agent B who is operating in a similar space by taking the current closed or open set M which is a stable model of A's set based logic program and sending $f(M)$ to B were f is some continuous function or an approximation to a continuous function. While the details of such applications a well beyond the scope of this paper, we end this section

by constructing a set based program whose stable models represent continuous functions $F : [0,1] \to [0,1]$. This represent a first step in developing a systems of agents who can pass continuous information or an approximation of continuous information to each other via a set based logic programming paradigm.

Let $\mathcal{R}$ be the real line, equipped with its usual topology. Let $\mathcal{R}^+$ be the set of all positive real numbers. Let ω be the set of all natural numbers. Let $\mathbf{Z}^+$ be the set of all positive integers, that is $\omega \setminus \{0\}$.

It is easy to see that there $\mathcal{R}$ has a countable base $\{ U_a \mid a \in \omega \}$ such that

(a) $U_0 = \mathcal{R}$,
(b) for each $a > 0$, U_a is an open interval (p_a, q_a) whose endpoints are dyadic rationals,
(c) the endpoint sequences $\langle p_a \rangle_{a \in \omega}$ and $\langle q_a \rangle_{q \in \omega}$ are computable,
(d) there is a monotonic function $e : \mathbf{Z}^+ \longrightarrow \mathbf{Z}^+$ such that, for each positive integer m and each $a > e(m)$, the diameter of U_a is less than 2^{-m}, and
(e) for all natural numbers a and b, if $U_a \subseteq U_b$, then $a \geq b$.

For any positive integer n, the product space $\mathcal{R}^n$ also possesses such a base, with the obvious difference that the sets U_n for $n > 0$ are products of open intervals and that there are $2n$ of computable sequences of endpoints. Obviously, such a construction can be relativised to the product spaces $[0,1]^n$.

Given such a base for the topology of $\mathcal{R}^n$, we will say that a function $f : \omega \longrightarrow \omega$ is a *representing function* for a continuous function $F : \mathcal{R}^n \longrightarrow \mathcal{R}^n$ provided that

1. $U_m \subseteq U_n \to U_{f(m)} \subseteq U_{f(n)}$,
2. for all n, $F(U_n) \subseteq U_{f(n)}$, and
3. for all $x \in \mathcal{R}^n$ and $k \in \omega$, there exists an m such that $x \in U_m$ and $U_{f(m)}$ has diameter less than 2^{-k}.

In such a case, we can recover F from f, since, for each $x \in \mathcal{R}^n$,

$$F(x) \text{ is the unique member of } \bigcap_{a \in \omega, x \in U_a} U_{f(a)}. \tag{13}$$

In the case of compact space, for instance $[0,1]$, condition (3) may be replaced by the existence of a function $d : \omega \to \omega$, called the modulus of continuity of the function F, such that for all k, U_m has diameter less than $2^{-d(k)}$ implies $U_{f(m)}$ has diameter less than 2^{-k}.

Conversely, given such a base $\{ U_a \mid a \in \omega \}$ and an arbitrary function $f : \omega \longrightarrow \omega$, it is natural to ask when is it the case that there is a continuous function $F : \mathcal{R}^n \to \mathcal{R}^n$ such that $F(x)$ is defined by (13). One can show that $f : \omega \to \omega$ represents a continuous function F if

(a) $(\forall m, n)(U_m \subset U_n \to U_{f(m)} \subset U_{f(n)})$.
(b) $(\forall k, m)(\exists r)(\forall t)[(U_t \subseteq [-k,k]^n \ \&\ diam(U_t) < 2^{-r}) \to diam(U_{f(t)}) < 2^{-m}]$.

We shall consider a simplified version of this type of phenomenon. For example, let

$$\mathcal{A}_n = \{A_{n,k} : k = 0, \ldots, 2^n - 1\} \cup \{B_{n,k} : k = 1, \ldots, 2^{n-1} - 2\}. \qquad (14)$$

where

$$A_{n,k} = \begin{cases} [0, \frac{1}{2^n}) & \text{if } k = 0, \\ (\frac{2^n-1}{2^n}, 1] & \text{if } k = 2^n - 1 \text{ and} \\ (\frac{k}{2^n}, \frac{k+1}{2^n}) & \text{if } k = 1, \ldots, 2^{n-1} - 2 \end{cases}$$

and

$$B_{n,k} = (\frac{2k+1}{2^{n+1}}, \frac{2k+3}{2^{n+1}}) \text{ for } k = 0, \ldots, 2^{n-1} - 1.$$

The significance of the family $\mathcal{A}_n$ is that every $x \in [0, 1]$ lies in an open interval I of diameter $1/2^n$ for some $I \in \mathcal{A}_n$. Now suppose that our representing function f of a continuous function $F : [0, 1] \to [0, 1]$ has the property that if $U_a \in \mathcal{A}_{2n}$, then $f(a) = b$ where $b \in \mathcal{A}_n$. Thus the modulus of continuity function in this case is given by $d(k) = \frac{1}{2^{2k+2}}$. That is, whenever U_m has diameter $< d(k)$, there is a t such that $U_m \subseteq U_t$ where $U_t \in \mathcal{A}_{2k}$. Hence $U_{f(m)} \subseteq U_{f(t)} \in \mathcal{A}_k$ and $diam(U_{f(m)}) \leq diam(U_{f(t)}) = 2^{-k}$. In fact, it easy to see that we can specify F by merely defining f on the a such that $U_a \in \bigcup_{n\geq 1} \mathcal{A}_{2n}$.

This given, we consider the following set based logic program $\mathcal{P}$. The constants of the underlying program will be $A_{n,k}$ such that $k = 0, \ldots, 2^n - 1$ and $n \geq 1$ and $B_{n,k}$ such that $k = 0, \ldots 2^{n-1} - 2$ for $n \geq 1$. Our program will contain one binary relation symbol $f(x, y)$ and one 0-ary predicate symbol C. The sense of a ground atom $f(E_{m,k}, F_{n,l})$ where $E, F \in \{A, B\}$ will simply be the open set $E_{m,k} \times F_{n,l}$ contained in $[0, 1] \times [0, 1]$. The sense of C is just $\{C\}$ so that the underlying space X consists of all $\{C\} \cup \{E_{m,k} \times F_{n,l} : E, F \in \{A, B\}$ and $m, k, n, l \in \omega\}$. We will take the miop operator $op_f = op_C = op^- = op_{id}$. Then we consider the following propositional set based program $\mathcal{P}$.

(1) $C \leftarrow f(X, Y), \neg C$ for all $X \in \mathcal{A}_{2n}$ and $Y \notin \mathcal{A}_n$,
(2) $C \leftarrow f(X, Y), f(X, Z), \neg C$ for all $X \in \mathcal{A}_{2n}$ and $Y, Z \in \mathcal{A}_n$ with $n \geq 1$ and $X \neq Y$,
(3) $C \leftarrow f(X, U), f(Y, V), \neg C$ for all $X, Y \in \bigcup_{n\geq 1} \mathcal{A}_{2n}$ and $U, V \in \bigcup_{n\geq 1} \mathcal{A}_n$ such that $X \subseteq Y$ but $U \nsubseteq V$.
(4) $f(X, Y) \leftarrow \neg f(X, U_1), \ldots, \neg f(X, U_{2^n+2^{n-1}-1})$ for each $n \geq 1$, $X \in \mathcal{A}_{2n}$ and $Y \in \mathcal{A}_n$ where $\mathcal{A}_n - \{Y\} = \{U_1, \ldots, U_{2^n+2^{n-1}-1}\}$ and
(5) $C \leftarrow \neg f(X, U_0), \ldots, \neg f(X, U_{2^n+2^{n-1}-1}), \neg C$ for each $n \geq 1$, $X \in \mathcal{A}_{2n}$ and $Y \in \mathcal{A}_n$ where $\mathcal{A}_n = \{U_0, \ldots, U_{2^n+2^{n-1}-1}\}$.

It is then easy to see by the same type analysis that we used in Example 1, that C can never be an element of a stable model M for $\mathcal{P}$. It follows that effect of the clauses in (1), (2), (3) is to ensure that we can think of f as specifying a function defined on some subset of $\bigcup_{n\geq 1} \mathcal{A}_{2n}$ such that for each $n \geq 1$, (i) $X \in \mathcal{A}_{2n}$ implies $f(X) \in \mathcal{A}_n$ and (ii) if $X \subseteq Y$, then $f(X) \subseteq f(Y)$. Finally the

clauses of (4) and (5) say that f must be defined on all of $\bigcup_{n\geq 1} \mathcal{A}_{2n}$. Thus we have the following.

Proposition 4. *The a-stable models of $\mathcal{P}$ for $a \in \{s, w\}$ correspond to*

$$f : \bigcup_{n\geq 1} A_n \to \bigcup_{n\geq 1} A_n$$

such that for all n, $a \in A_{2n}$ implies $f(a) \in A_n$ and hence all such f define a continuous functions $F : [0,1] \longrightarrow [0,1]$ via (13).

We should note that we did not really need to use set based programming in this case as we could do the same thing in normal logic programming. The reason for presenting this construction is that by setting it in this framework, we can reason directly about the approximating interval $U_a \times U_{f(a)}$ for the function F in this case. Moreover, the framework of representing functions allows us to reason about continuous transformations between different agents. Of course, in actual practice, we can only reason about approximations of continuous functions since continuous functions and/or their representing functions are infinite objects. In our setting, we can reason about approximations of representing functions by fixing some n_0 and restricting our program clauses so that all indices involved must be smaller than or equal to n_0.

5 Conclusions and Further Work

In this paper, we defined a variant of logic programming, called set based logic programming, where the atoms have an associated *sense* which is a subset of a given space X, The usual one step consequence operator is modified so that it only produces fixed points of some monotonic idempotent operator op^+ on 2^X, and the satisfaction of negative atoms is determined by either weak or strong negation relative to a monotonic idempotent operator op^- on 2^X. We have illustrated how such programs embody a natural polymorphism and can be used to naturally express problems in the various continuous domains. We envision many other applications of our set based logic programming formalism in such areas as graphics, image and voice compression, and other domains where there are natural representation of processes that accept subsets of spaces as inputs and compute outputs which are also subsets of those spaces. For example, set based programs with miops can provide a logic-based approach to hybrid dynamical systems.

This paper is the first of a series of papers that will explore the set based logic programming paradigm. For example there are a number of concepts from logic programming such as, *well-founded model* [VGRS91], stratified programs, etc. that can be lifted to the present context almost verbatim. Thus one can develop a rich theory of set based logic programs. Our set based logic programs share certain features with Constraint Logic Programming [JM94] and the exact connections need to be explored. Third, one can think about the *senses* of atoms

as annotations of the kind discussed in [KS92]. While there are various differences between our approach and [JM94], for instance our use of negation as means to enforce constraints as in [Nie99], the relationship between these two approaches should be explored. Fourth, set based logic programming can be studied in the more general setting of nonmonotonic inductive definitions [Den00] (e.g. iterated inductive definitions of Feferman [Fef70]).

Finally we note that one there is third type of satisfaction relation for negative atoms that one might consider in the setting of set based logic programs. That is, suppose that the set of subsets of X which are closed under op^+ gives rise to a complete lattice $\mathcal{L}_{[\![\cdot]\!],op^+,X}$ where the meet $\wedge$, and join $\vee$, of any collection $\{A_i : i \in \Gamma\}$ of subsets of $\mathcal{L}_{[\![\cdot]\!],op^+,X}$ is given by equations

$$\wedge\{A_i : i \in \Gamma\} = op^+(\bigcap_{i\in\Gamma} A_i) \text{ and}$$
$$\vee\{A_i : i \in \Gamma\} = op^+(\bigcup_{i\in\Gamma} A_i).$$

For example, if $op^+ = op_{id}$, then we can just let $\mathcal{L}_{[\![\cdot]\!],op^+,X} = 2^X$. If X is a topological space and $op^+ = op_{cl}$, then $\mathcal{L}_{[\![\cdot]\!],op^+,X}$ is just lattice of closed subsets of X. If X is a vector space V and $op^+ = op_{subsp}$, then $\mathcal{L}_{[\![\cdot]\!],op^+,X}$ is just lattice of all subspaces of V.

Now assume that the sense of any atom A is closed under op^+. Then we can extend the sense map $[\![\cdot]\!]$ to all sentences via the following definition.

Definition 8. *(Senses of* closed *formulas)*

1. $[\![A \vee B]\!] = [\![A]\!] \wedge [\![B]\!]$
2. $[\![A \wedge B]\!] = [\![A]\!] \vee [\![B]\!]$
3. $[\![A \rightarrow B]\!] = \bigwedge\{W \in \mathcal{L}_{op} : (W \vee [\![A]\!]) \geq [\![B]\!])\}$
4. $[\![\neg A]\!] = \bigwedge\{W \in \mathcal{L}_{op} : (W \vee [\![A]\!]) = op(X)\}$
5. $[\![\exists x A(x)]\!] = \bigvee\{[\![A(t)]\!] : t \text{ a variable-free term}\}$
6. $[\![\forall x A(x)]\!] = \bigwedge\{[\![A(t)]\!] : t \text{ a variable-free term}\}$

Then one can define a new satisfaction relation $\models^h_{[\![\cdot]\!],op^+}$ by defining

$$I \models^h_{[\![\cdot]\!],op^+} A \iff [\![A]\!] \leq I.$$

We can then define a new one-step consequence operator and a new Gelfond-Lifschitz operator based on this new definition of $\models^h_{[\![\cdot]\!],op^+}$. We shall show in a subsequent paper that under certain circumstances, this new satisfaction relation is closely related to an intuitionistic logic type semantics based on complete Heyting algebras and Kripke models as described in [TvD88]. However, we should

note that such a definition does not always yield an interesting new semantics in the most general setting of this paper. That is, suppose that X is a vector space V and $op = op_{subsp}$. Suppose that A is non-trivial subspace of V. Let $a_1, a_2 \cdots$ be a basis of A. We claim that for any $x \in V$, there is a subspace W_x such that $W_x \vee A = V$ and $x \notin W$. This is clear if $x \in A$. If $x \notin A$, then x is independent over A so that we can extend $x, a_1, a_2, \cdots$ to a basis of V by adding say, $b_1, b_2, \cdots$. Then it is easy to see that if W_x is the subspace generated by $a_1 + x, b_1, b_2, \cdots$, then $W_x \vee A = V$, but $x \notin W_x$. In this case, condition (4) becomes

$$[\![\neg A]\!] = \bigwedge \{W \in \mathcal{L}_{[\![\cdot]\!], op_{subsp}, X} : (W \vee [\![A]\!]) = op_{subsp}(X)\} = op_{subsp}(\emptyset) = \{\mathbf{0}\}$$

where $\mathbf{0}$ is the zero vector of V. Thus in this case, $[\![\neg A]\!] = \{\mathbf{0}\}$ unless $A = \{\mathbf{0}\}$ in which case $[\![\neg A]\!] = V$ which would mean the negation is essentially trivial.

Acknowledgments: The second author has been partly supported by NSF grants IIS-0097278 and IIS-0325063. The third author has been partly supported by NSF grants DMS 0400507 and DMS 0654060. The authors wish to thank David Jakel and Angel Rivera for contributions and valuable discussion, particularly in regard to the description of the representation of continuous functions on the real numbers.

References

[ABW88] Apt, K., Blair, H.A., and Walker, A. Towards a theory of Declarative Knowledge. In *Foundations of Deductive Databases and Logic Programming*, J. Minker, Ed. Morgan Kaufmann, 89–148, 1988.

[ASP01] *Proceedings of the AAAI Spring 2001 Symposium on Answer Set Programming, Stanford, CA*, 2001.

[Ba03] Baral, C., *Knowledge Representation, Reasoning and Declarative Problem Solving*, Cambridge University Press, 2003.

[BS81] Burris, S. and Sankappanavar, H.P. *A Course in Universal Algebra*, Graduate Texts in Mathematics, no. 78, Springer-Verlag, 1981.

[BS89] Batarekh, A. and Subrahmanian, V.S. Topological Model Set Deformations in Logic Programming, Fundamenta Informaticae, Vol. XII, No. 3, pps 357–400, 1989.

[BMR01] Blair, H.A., Marek, V.W. and Remmel, J.B. Spatial Logic Programming, in *Proceedings SCI 2001*, Orlando, FL, July, 2001.

[Den00] Denecker, M. Extending classical logic with inductive definitions. In *First International Conference on Computational Logic (CL2000)*. Lecture Notes in Artificial Intelligence, vol. 1861. Springer, 703–717, 2000.

[Fef70] Feferman, S. Formal theories for transfinite iterations of generalized inductive definitions and some subsystems of analysis. In *Intuitionism and Proof theory*, A. Kino, J. Myhill, and R. Vesley, Eds. North Holland, 303–326, 1970.

[GL02] Gelfond, M. and Leone, N. Logic Programming and Knowledge Representation – A-Prolog perspective. *Artificial Intelligence* 138:3–38, 2002.

[GL88] Gelfond, M. and Lifschitz, V. The stable model semantics for logic programming. In *Proceedings of the International Joint Conference and Symposium on Logic Programming*. MIT Press, 1070–1080, 1988.

[JM94] Jaffar, J. and Maher, M. Constraint logic programming: A survey. *Journal of Logic Programming*, 19-20:503–581, 1994.

[Ku66] Kuratowski, K. *Topology*, Academic Press 1966.

[Li94] V. Lifschitz. Minimal belief and negation as failure. *Artificial Intelligence* 70:53–72, 1994.

[KS92] Kifer, M. and Subrahmanian, V.S. Theory of generalized annotated logic programming and its applications. *Journal of Logic Programming* 12:335–367, 1992.

[Kl67] Kleene, S.C. *Introduction to Metamathematics*, North-Holland, 1967.

[MT99] Marek, V.W., and Truszczyński, M. Stable Models and an Alternative Logic Programming Paradigm. *The Logic Programming Paradigm*, pp. 375–398. Series Artificial Intelligence, Springer-Verlag, 1999.

[Nie99] Niemelä, I. Logic programs with stable model semantics as a constraint programming paradigm. *Annals of Mathematics and Artificial Intelligence 25,* 3,4, 241–273, 1999

[TvD88] Troelstra, A.S. and van Dalen, D. *Constructivism in Mathematics*, volumes I and II, North-Holland, 1988.

[VGRS91] Van Gelder, A., Ross, K., and Schlipf, J. The Well-Founded Semantics for General Logic Programs. *Journal of the ACM 38,* 3, 620–650, 1991.

[W99] Weisstein, E.W., et.al. "Lattice." From MathWorld–A Wolfram Web Resource. http://mathworld.wolfram.com/Lattice.html