On Prudent Bravery and Other Abstractions

Melvin Fitting
fitting@alpha.lehman.cuny.edu
Dept. Mathematics and Computer Science
Lehman College (CUNY), Bronx, NY 10468
Depts. Computer Science, Philosophy, Mathematics
Graduate Center (CUNY), 33 West 42nd Street, NYC, NY 10036 *

October 13, 1994

Abstract

A special class of partial stable models, the *intrinsic* ones, is singled out for consideration, and attention is drawn to the largest one, which we designate as the *prudently brave* one. It is the largest partial stable model that is compatible with every partial stable model. As such, it is an object of natural interest. Its existence follows from general properties of monotonic functions, so it is a robust notion. The proofs given concerning intrinsic stable models are not new, but they appeared earlier in quite different contexts. What we do, essentially, is call the attention of the logic programming and non-monotonic reasoning community to them. We further show the entire development fits into the bilattice framework presented in [5], and thus applies very generally. Finally we show that extending logic programming machinery to allow embedded implications does not essentially change matters: stable model semantics, including the notion of a prudently brave one, continues to apply.

1 Introduction

This paper has three closely related purposes. The first is to propose a candidate from the family of stable models that deserves investigation, but which has been ignored up till now; we call it the *prudently brave* model. The second is to show that all the machinery developed for the investigation of stable and well-founded models extends directly to allow embedded implications. The third is to show that all this fits naturally into the general framework introduced in [5], and so really extends to a whole family of logic programming languages, with a variety of spaces chosen for truth values. The paper is not self-contained, but much of it can be read and understood on its own. Now we give a more detailed description of the work presented here.

The number of semantics introduced for logic programming seems to be increasing exponentially, and we hesitate to propose yet another candidate. However, we feel justified because what we propose is far from new — it has been available for many years, but has gone unrecognized. In addition, it is quite natural, and very robust — it depends not at all on particular details of the logic programming language in question.

*Research partly supported by NSF Grant CCR-9104015.

It is generally agreed that stable model semantics captures an important version of non-monotonic negation, and is consequently a success. Still, there are problems with the approach. If one uses only two-valued (or classical) stable models, some programs will not have any stable models, while others will have several. If one uses three-valued (or partial, or stationary) stable models every program will have at least one, but there is still the problem that some programs will have several. The most widespread solution to this difficulty has been to go to extremes, generally called *cautious* or *skeptical* and *brave* or *credulous*. The cautious version is quite straightforward: among the family of partial stable models there is a unique one containing the least information — the well-founded model. It assigns a classical truth value to a ground atom only if every partial stable model makes the same assignment. The brave version is somewhat more problematic since, in general, there is no largest partial stable model. Instead one asks whether there exists some partial stable model in which a proposition is true. This means we do not have a single model to deal with, but rather a whole family of them. Another plausible approach is to work with *maximal* partial stable models, assigning a classical truth value to a ground atom only if every maximal partial stable model makes that assignment. This gives us a single object to deal with, but in general it will not itself be a partial stable model.

In Section 5 we propose a candidate of interest which we call the *prudently brave* model: the *largest intrinsic partial stable model*. The meaning of this phrase is easily given. Call two partial stable models *compatible* if they do not contradict each other; if one assigns a classical truth value to a ground atom, the other will either assign the same truth value or leave the value of the ground atom undetermined. Call a partial stable model *intrinsic* if it is compatible with every partial stable model. Every logic program has intrinsic partial stable models — the well-founded model is the smallest such. In fact, being intrinsic is easily characterized: a partial stable model is intrinsic if and only if it is below every maximal partial stable model (we show this later). Not only does every logic program have intrinsic partial stable models but among them there is a unique largest one, which we call the *prudently brave* one. Whereas the cautious partial stable model assigns a classical truth value only if every other partial stable model agrees, the prudently brave one assigns a classical truth value whenever no other partial stable model can contradict it. In addition to proving existence, we give a construction of sorts for the largest intrinsic partial stable model. It is not as simple as those for the cautious version, but it is nonetheless a natural one.

The notion of an *intrinsic* fixed point for an operator goes back to Saul Kripke in his paper on the theory of truth [9]. It was discovered independently by Manna and Shamir in [10] under the name *optimal* fixed point. The particular construction of it given below comes from [2]. Although the formal settings in those papers were different, the same development and the same proofs carry over to the theory of stable models. In fact what we are really looking at is a general feature of a whole class of operators meeting certain monotonicity conditions, and the stability transform is such an operator. We take this as an argument that the largest intrinsic partial stable model is an extremely natural thing to consider. Its existence and general properties do not depend on sensitive details of particular logic programs; instead it will remain available through a wide variety of variations — it is a robust notion.

To make the argument accessible to the widest audience we present things specifically for logic programming in the standard sense, using the well-developed notion of partial, or three-valued semantics. Afterwards we observe that the entire development, with no changes in proofs, applies in the general setting of logic programming over a distributive bilattice of truth values, as given in [3, 5]. As we said above, we are really looking at general properties of operators meeting certain monotonicity conditions, and not at logic programming minutiae.

In [8] a three-valued semantics was developed for logic programming with embedded implications

and negation as failure. We conclude this paper by showing that even with embedded implications, the machinery of logic programming still fits the general bilattice paradigm of [3, 5] and so not only does the semantics of [8] result immediately, but the entire machinery of stable model semantics is also available, including that of prudent bravery. This seems to be unexplored territory, and we hope it finds investigators.

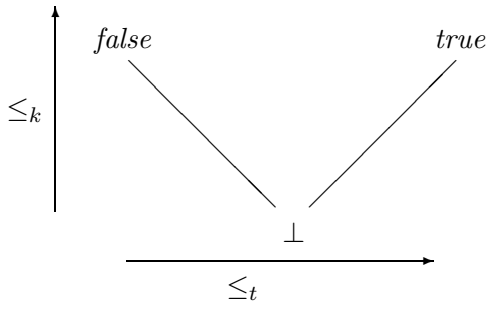
2 Basics

We sketch the syntactical and the semantical underpinnings we will be using. As we remarked above, the existence of a largest intrinsic stable model is quite robust, so to keep the presentation easy to follow, we adopt the simplest syntactical setting for now, the conventional one.

Definition 2.1 A *program* is a finite set of clauses. A *clause* is a formula of the form $A \leftarrow L_1 \wedge \ldots \wedge L_k$, where A is atomic and $L_1, \ldots, L_k$ are literals. It is allowed that k may be 0; identify the empty conjunction with *true*. If $\mathcal{P}$ is a program, by $\mathcal{P}^*$ we denote the set of all ground instances of $\mathcal{P}$ over the Herbrand base.

Thus the syntax of programs is quite basic: no explicit (classical) negation, no disjunctions in heads, no embedded implications. Next we turn to the elements of the semantics we will be using. We want to consider *partial* or *three-valued* models, and there is more than one kind of machinery that can be used for this purpose. We do things in a way that generalizes smoothly.

Definition 2.2 *Truth values* are *true*, *false*, and $\bot$, where $\bot$ is thought of as *undefined* or *unknown*. The collection of truth values is given two partial orderings, $\leq_k$ and $\leq_t$ according to the following diagram.



The ordering $\leq_t$ is intended to represent 'degree of truth.' Under this ordering the space of truth values is a complete lattice. We use $\wedge$ and $\vee$ for finitary meet and join in this lattice, and $\bigwedge$ and $\bigvee$ for the infinitary version. The $\leq_k$ ordering represents 'degree of knowledge.' This does not give a complete lattice, but it is a *complete semi-lattice*; that is, we have a partial ordering for which arbitrary non-empty meets exist, but joins exist only for directed sets. We use $\otimes$ and $\oplus$ for finitary meet and join (when it exists) with respect to this ordering, and $\prod$ and $\sum$ for the infinitary versions. In addition we have a unary operator, $\neg$, which is not lattice-theoretic. It is a symmetry with respect to $\leq_t$: $\neg \mathit{true} = \mathit{false}$; $\neg \mathit{false} = \mathit{true}$; $\neg\bot = \bot$.

Definition 2.3 A *valuation* is a mapping from the set of ground atoms to the space of truth values.

Valuations are often represented using other machinery. A valuation v can be identified with a set of ground *literals*: $\{A \mid v(A) = \mathit{true}\} \cup \{\neg A \mid v(A) = \mathit{false}\}$, where A represents a ground atom. Conversely any such set determines a valuation, provided it is consistent. Alternatively a valuation can be identified with a pair of sets of ground atoms, $\langle I^+, I^-\rangle$ where $I^+ = \{A \mid v(A) = \mathit{true}\}$ and $I^- = \{A \mid v(A) = \mathit{false}\}$. Again any such pair determines a valuation, provided $I^+ \cap I^- = \emptyset$. We have represented them as mappings because it is a version that generalizes readily.

Definition 2.4 The space of valuations is given two pointwise orderings as follows. $v \leq_t w$ if $v(A) \leq_t w(A)$ for every ground atom A. Likewise $v \leq_k w$ if $v(A) \leq_k w(A)$ for all ground atoms A.

If we represent valuations as sets of literals, as mentioned above, the $\leq_k$ ordering corresponds to subset. (The $\leq_t$ ordering is more complicated to characterize.) Also, the observations made above about the space of truth values being a complete lattice under $\leq_t$ and a complete semi-lattice under $\leq_k$ are pretty trivial, but it is easy to check that the space of valuations inherits these features, and it is here that they play their major role. We use $\wedge$, $\vee$, $\bigwedge$, $\bigvee$, $\otimes$, $\oplus$, $\prod$, and $\sum$ for the lattice and semi-lattice operations on the valuation space in the obvious way.

3 An Example

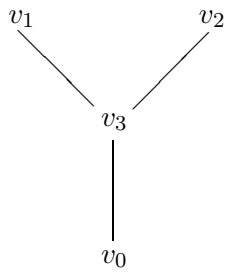
We present an example illustrating the general ideas involved. Definitions of partial stable models are fairly standard [14]; one characterization of them can be found in the next section. Consider the following logic program (in which we have combined clauses using disjunction, in the obvious way):

$$\begin{array}{lcl} \mathtt{A} & \leftarrow & \neg\mathtt{B} \\ \mathtt{B} & \leftarrow & \neg\mathtt{A} \\ \mathtt{C} & \leftarrow & \neg\mathtt{D} \vee \mathtt{A} \vee \mathtt{B} \\ \mathtt{D} & \leftarrow & \neg\mathtt{C} \\ \mathtt{E} & \leftarrow & \mathtt{A} \vee \mathtt{B} \end{array}$$

This program has four partial stable models, which we give in table form.

	A	B	C	D	E
v_0	$\bot$	$\bot$	$\bot$	$\bot$	$\bot$
v_1	*true*	*false*	*true*	*false*	*true*
v_2	*false*	*true*	*true*	*false*	*true*
v_3	$\bot$	$\bot$	*true*	*false*	$\bot$
$v_1 \otimes v_2$	$\bot$	$\bot$	*true*	*false*	*true*

The valuation v_0 is the well-founded model; it is the smallest in the $\leq_k$ ordering. Both v_1 and v_2 are maximal in the $\leq_k$ ordering; both are two-valued stable models. $v_1 \otimes v_2$, the meet of v_1 and v_2 in the $\leq_k$ ordering, is not a partial stable model. Valuation v_3 is the largest intrinsic partial stable model. The structure of partial stable models is given in the following diagram, where the vertical direction represents the $\leq_k$ ordering, and the $\leq_t$ ordering is ignored.



4 The Stability Operator

Stable models in the two-valued sense are generally characterized using the Gelfond-Lifschitz transform, which replaces a logic program using negation as failure by one that does not [6]. This approach was generalized to allow partial stable models in [14]. Here we follow an equivalent version from [5] partly because it works most smoothly for what we need here, partly because generalizations are immediate.

Definition 4.1 Suppose v_1 and v_2 are valuations. We define what we call a *pseudo-valuation*, $v_1 \triangle v_2$, as follows. For a ground atom A,

$$\begin{array}{rcl} (v_1 \triangle v_2)(A) & = & v_1(A) \\ (v_1 \triangle v_2)(\neg A) & = & \neg v_2(A) \end{array}$$

A pseudo-valuation differs from a valuation in that it treats a negative literal essentially as a different kind of positive literal, assigning a truth value to it independently. The intention is, in $v_1 \triangle v_2$, v_1 gives positive information while v_2 gives negative information, and these are independent of each other. The action of a pseudo-valuation is extended to conjunctions and disjunctions exactly as that of valuations is.

Definition 4.2 Let $\mathcal{P}$ be a program. We define the *extended immediate consequence operator*, $\Psi_{\mathcal{P}}$, mapping pairs of valuations to valuations, as follows. Let v_1, v_2 be valuations; $\Psi_{\mathcal{P}}(v_1, v_2)$ is the valuation such that:

$$\Psi_{\mathcal{P}}(v_1, v_2)(A) = \bigvee\{(v_1 \triangle v_2)(B) \mid A \leftarrow B \in \mathcal{P}^*\}.$$

Since the disjunction of an empty family is *false*, if A is not the head of any member of $\mathcal{P}^*$, the definition above sets $\Psi_{\mathcal{P}}(v_1, v_2)(A) = \mathit{false}$. In [5] it is shown that with respect to the $\leq_t$ ordering the extended immediate consequence operator is monotonic in its first argument and anti-monotonic in its second. Since the space of valuations is a complete lattice with respect to the $\leq_t$ ordering, least fixed points exist for monotone mappings, and if the second argument of the extended immediate consequence operator is held fixed, we have a monotone function of its first argument.

Definition 4.3 The *derived, or stability, operator* of $\Psi_{\mathcal{P}}$ is the single input mapping $\Psi'_{\mathcal{P}}$ given by: $\Psi'_{\mathcal{P}}(v)$ is the smallest fixed point, in the $\leq_t$ ordering, of the mapping $(\lambda x)\Psi_{\mathcal{P}}(x, v)$.

The derived operator generalizes the Gelfond-Lifschitz transformation quite directly. Its fixed points are exactly the partial stable models. A general theory of the behavior of the derived operator may be found in [5]. In particular, there are proofs given for the following items:

1. $\Psi'_{\mathcal{P}}$ is monotonic with respect to $\leq_k$;
2. since $\leq_k$ gives the space of valuations the structure of a complete semi-lattice, monotone mappings have least fixed points; the least fixed point of $\Psi'_{\mathcal{P}}$ is the well-founded model for $\mathcal{P}$;
3. $\Psi'_{\mathcal{P}}$ is anti-monotonic with respect to $\leq_t$ (this leads to the alternating fixpoint approach to the well-founded model [15], something we do not need here).

5 Results, Proofs, and Constructions

In this section we prove the existence of a largest intrinsic stable model, give a sort of construction for it, and prove some related results of interest. As a matter of fact, essentially nothing in this section depends on the details of logic programming (the arguments are taken directly from [2]). We have certain monotonicity features of the derived, or stability, operator and these are all that we will use. This makes explicit what we said earlier about the robustness of these ideas. We begin by summarizing the items we will be using.

1. The space of valuations, under $\leq_k$, is a complete semi-lattice, that is, $\leq_k$ is a partial ordering for which arbitrary non-empty meets exist, and joins exist for directed subsets (in particular, for chains).
2. For a program $\mathcal{P}$, the derived operator $\Psi'_{\mathcal{P}}$ is monotonic in the $\leq_k$ ordering.

These two items are all that will be used in this section. Recall, partial stable models of $\mathcal{P}$ are the fixed points of $\Psi'_{\mathcal{P}}$; an *intrinsic* fixed point is one that is compatible with every fixed point; and v and w are compatible if $v \oplus w$ is defined, where $\oplus$ is the join operation for the $\leq_k$ ordering (this is not the original definition, but is easily seen to be equivalent). For the rest of this section we refer to *fixed points* rather than to *partial stable models*.

We recall two familiar results. First, if $v \leq_k \Psi'_{\mathcal{P}}(v)$, then there is a smallest fixed point of $\Psi'_{\mathcal{P}}$ above v. This is a standard result with a standard proof: construct an ascending transfinite sequence, beginning with v, iterating $\Psi'_{\mathcal{P}}$, taking sups at limit ordinals. The existence of sups for chains is enough to guarantee the process will successfully continue. This produces a fixed point which must be above v and must be the smallest such. There is a companion result: if $\Psi'_{\mathcal{P}}(v) \leq_k v$, then there is a largest fixed point below v. The argument is essentially the same, but one constructs a descending sequence this time, using the assumed existence of arbitrary non-empty meets.

Proposition 5.1 *If $v_0 \leq_k \Psi'_{\mathcal{P}}(v_0)$ then there is a maximal fixed point above v_0; in particular, maximal fixed points exist.*

Proof Suppose $v_0 \leq_k \Psi'_{\mathcal{P}}(v_0)$. Let $\mathbf{E}$ be the set of valuations w such that $w \leq_k \Psi'_{\mathcal{P}}(w)$. Since $v_0 \in \mathbf{E}$, the set is not empty. The ordering $\leq_k$, restricted to $\mathbf{E}$, is still a partial order. Now we show that every chain in $\mathbf{E}$ has an upper bound in $\mathbf{E}$.

Let $\mathbf{C}$ be a chain in $\mathbf{E}$. Then $\sum \mathbf{C}$ exists, since the space of valuations is closed under directed joins. Suppose $w \in \mathbf{C}$; then $w \leq_k \sum \mathbf{C}$, so by monotonicity, $\Psi'_{\mathcal{P}}(w) \leq_k \Psi'_{\mathcal{P}}(\sum \mathbf{C})$. Since $w \in \mathbf{C}$, $w \leq_k \Psi'_{\mathcal{P}}(w)$. It follows that $w \leq_k \Psi'_{\mathcal{P}}(\sum \mathbf{C})$ for every $w \in \mathbf{C}$; consequently $\sum \mathbf{C} \leq_k \Psi'_{\mathcal{P}}(\sum \mathbf{C})$, so $\sum \mathbf{C} \in \mathbf{E}$ by definition, and we have established that $\mathbf{E}$ is closed under chain sups.

Since $\mathbf{E}$ is a partially ordered set in which each chain has an upper bound, by Zorn's Lemma each member of $\mathbf{E}$ can be extended to a maximal member; in particular this holds for v_0. All that is left is to show that each maximal member of $\mathbf{E}$ is a maximal fixed point of $\Psi'_{\mathcal{P}}$ in the space of

valuations. So, let m be a maximal member of $\mathbf{E}$. Since $m \in \mathbf{E}$, $m \leq_k \Psi'_{\mathcal{P}}(m)$. By monotonicity, $\Psi'_{\mathcal{P}}(m) \leq_k \Psi'_{\mathcal{P}}(\Psi'_{\mathcal{P}}(m))$, so $\Psi'_{\mathcal{P}}(m) \in \mathbf{E}$. Since m is maximal in $\mathbf{E}$ and $m \leq_k \Psi'_{\mathcal{P}}(m)$, it follows that $m = \Psi'_{\mathcal{P}}(m)$, so m is itself a fixed point. Any other fixed point must belong to $\mathbf{E}$, and so cannot be above m, since m is maximal in $\mathbf{E}$. It follows that m is a maximal fixed point in the space of valuations.

We have proved the first part of the Proposition. The second follows easily. There is a smallest valuation $v_\perp$ (it assigns every ground atom $\perp$, or more abstractly, take the meet of the space of all valuations). Since it is smallest, trivially $v_\perp \leq_k \Psi'_{\mathcal{P}}(v_\perp)$, so by the first part $v_\perp$ is below a maximal fixed point, hence maximal fixed points exist. ■

Definition 5.2 We denote by $\mathcal{M}$ the set of maximal fixed points (a non-empty collection, by the previous Proposition).

Proposition 5.3 *A fixed point v is intrinsic if and only if $v \leq_k \prod \mathcal{M}$.*

Proof First, suppose v is an intrinsic fixed point. Take an arbitrary $m \in \mathcal{M}$. Since m is a fixed point and v is intrinsic, $v \oplus m$ exists. Now $v \leq_k v \oplus m$, and v is a fixed point, so $v = \Psi'_{\mathcal{P}}(v) \leq_k \Psi'_{\mathcal{P}}(v \oplus m)$. Similarly $m \leq_k \Psi'_{\mathcal{P}}(v \oplus m)$, so $v \oplus m \leq_k \Psi'_{\mathcal{P}}(v \oplus m)$. It follows from the previous proposition that there is a maximal fixed point m_0 above $v \oplus m$. But $m \leq_k v \oplus m \leq_k m_0$, and m itself is a maximal fixed point, so $m = v \oplus m = m_0$. Since $m = v \oplus m$, $v \leq_k m$. Finally, since m was an arbitrary member of $\mathcal{M}$, $v \leq_k \prod \mathcal{M}$.

Conversely, suppose v is a fixed point and $v \leq_k \prod \mathcal{M}$. Let w be any fixed point. Again by the previous Proposition, w is below some maximal fixed point, say m. Then $w \leq_k m$, but also $v \leq_k \prod \mathcal{M} \leq_k m$. Thus v and w have a common upper bound, so $v \oplus w$ exists, since it is the meet of the set of all upper bounds for $\{v, w\}$, a set which has been established to be non-empty. ■

Proposition 5.4 *There is a largest intrinsic fixed point.*

Proof Let $m \in \mathcal{M}$, so that $\prod \mathcal{M} \leq_k m$. Then $\Psi'_{\mathcal{P}}(\prod \mathcal{M}) \leq_k \Psi'_{\mathcal{P}}(m) = m$. Since m was arbitrary, $\Psi'_{\mathcal{P}}(\prod \mathcal{M}) \leq_k \prod \mathcal{M}$. It follows that there is a largest fixed point below $\prod \mathcal{M}$, and hence, by the previous Proposition, a largest intrinsic fixed point exists. ■

Finally, we present a slightly more constructive approach to the largest intrinsic fixed point. First, recall one of the ways the well-founded model, which is the smallest fixed point of $\Psi'_{\mathcal{P}}$, can be produced. Start with the smallest valuation $v_\perp$, the one assigning $\perp$ to every ground atom. Produce a (generally transfinite) sequence of valuations as follows (where λ represents an arbitrary limit ordinal).

$$\begin{aligned} v_0 &= v_\perp \\ v_{\alpha+1} &= \Psi'_{\mathcal{P}}(v_\alpha) \\ v_\lambda &= \sum\{v_\alpha \mid \alpha < \lambda\} \end{aligned}$$

It is well-known that this sequence is increasing and converges to the smallest fixed point, hence to the well-founded model.

Now construct an alternative sequence, as follows.

$$\begin{aligned} w_0 &= \prod \mathcal{M} \\ w_{\alpha+1} &= \Psi'_{\mathcal{P}}(w_\alpha) \\ w_\lambda &= \prod\{w_\alpha \mid \alpha < \lambda\} \end{aligned}$$

In the proof of the previous Proposition it was shown that $\Psi'_{\mathcal{P}}(\prod \mathcal{M}) \leq_k \prod \mathcal{M}$. It follows easily that the w_α sequence is decreasing. In fact, it converges to the largest intrinsic fixed point, that is, to our candidate for the prudently brave model. The proof that it does so is straightforward, and is omitted.

6 Bilattices

In [3, 5] bilattices were used to provide a suitable abstract setting within which to discuss logic programming semantics. In this section we briefly give the basic definitions, and sketch how the notion of prudently brave stable model fits the general bilattice framework. For more detail see the papers just cited, along with [4], and [7], which is where bilattices originated in the first place.

Definition 6.1 A *pre-bilattice* is a structure $\langle \mathcal{B}, \leq_t, \leq_k \rangle$ where $\mathcal{B}$ is a non-empty set and $\leq_t$ and $\leq_k$ are each partial orderings giving $\mathcal{B}$ the structure of a lattice with a top and a bottom.

We call a pre-bilattice *complete* if both orderings have the structure of complete lattices. This is somewhat different from the original definition in [7].

Definition 6.2 In a pre-bilattice $\langle \mathcal{B}, \leq_t, \leq_k \rangle$, meet and join under $\leq_t$ are denoted $\wedge$ and $\vee$, and meet and join under $\leq_k$ are denoted $\otimes$ and $\oplus$. Top and bottom under $\leq_t$ are denoted *true* and *false*, and top and bottom under $\leq_k$ are denoted $\top$ and $\bot$. If the pre-bilattice is complete, infinitary meet and join under $\leq_t$ are denoted $\bigwedge$ and $\bigvee$, and infinitary meet and join under $\leq_k$ are denoted $\prod$ and $\sum$.

The operations $\wedge$ and $\vee$ should be thought of as generalizations of conjunction and disjunction in the classical space $\{$*false*, *true*$\}$. The operations $\otimes$, consensus, and $\oplus$, gullability, are not meaningful in the classical setting.

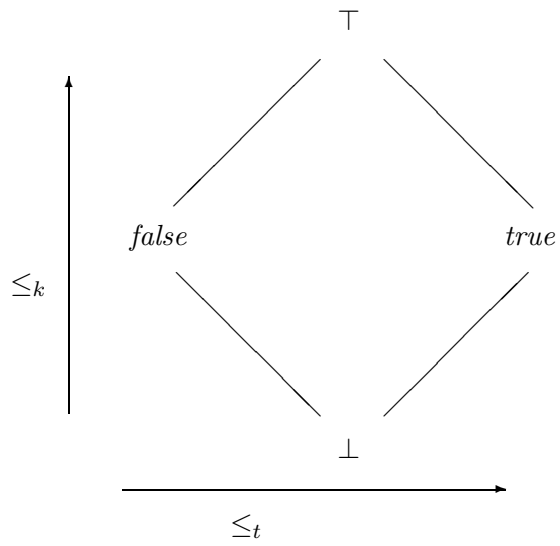
Definition 6.3 Bilattice versions:

- A *distributive bilattice* is a pre-bilattice $\langle \mathcal{B}, \leq_t, \leq_k \rangle$ in which all 12 distributive laws connecting $\wedge$, $\vee$, $\otimes$ and $\oplus$ hold. An *infinitely distributive bilattice* is a complete pre-bilattice in which all infinitary, as well as all finitary, distributive laws hold.

- The pre-bilattice $\langle \mathcal{B}, \leq_t, \leq_k \rangle$ satisfies the *interlacing conditions* if each of the lattice operations, $\wedge$, $\vee$, $\otimes$, $\oplus$, is monotone with respect to both orderings. If the pre-bilattice is complete, it satisfies the *infinitary interlacing conditions* if each of the infinitary meet and join operations is monotone with respect to both orderings.

- A bilattice has a *negation* if there is a mapping $\neg$ that reverses the $\leq_t$ ordering, leaves unchanged the $\leq_k$ ordering, and $\neg\neg x = x$. Similarly a bilattice has a *conflation* if there is a mapping $-$ that reverses the $\leq_k$ ordering, leaves unchanged the $\leq_t$ ordering, and $--x = x$. If a bilattice has both, we say they *commute* if $-\neg x = \neg - x$ for all x.

There are lots of bilattices meeting all the conditions above: infinitely distributive, satisfying the infinitary interlacing laws, with a negation and a conflation that commute. The simplest example is the well-known four-valued logic of Belnap [1], shown below. In it, negation is a left-right interchange, while conflation is similar vertically.



We do not give other examples of bilattices, or construction techniques here. These can be found elsewhere. But in [5] we showed partial stable model semantics, and well-founded semantics, found a kind of natural home in bilattices. That is, if one chooses a bilattice meeting the conditions above as the space of truth values, all the fundamental semantic facts about stable and well-founded models can be established with simple algebraic proofs. The question now is how does the notion of intrinsic stable models, and the prudently brave one fit into a general bilattice development? The answer is quite simple.

In bilattices having a conflation operation there is a notion of a member being *consistent*. Briefly, x is consistent if $x \leq_k -x$. The consistent members of Belnap's logic are the members of Kleene's strong three-valued logic, and in general, this notion of consistency is quite a natural one. The collection of consistent members of a distributive bilattice having a negation and a conflation that commute always constitutes a complete semi-lattice (a proof can be found in [3]). In all of the proofs of the previous section, the fact that valuations were a complete semi-lattice was essentially all we used about them. Thus the entire development carries over with no change whatsoever to the bilattice setting. In effect, what we presented was merely the special case where the truth value space was Belnap's four-valued logic. We presented only the special case because it was the one most familiar to readers, and we felt it adequately represented the whole class of examples.

7 Embedded implications

In [8] a three-valued semantics was given for a logic programming language containing embedded implications and negation as failure. This built on earlier work of [13], and was related to [11, 12]. In this section we show how logic programming with embedded implications can be generalized naturally to the bilattice context in a way that, when specialized to the consistent part of Belnap's logic, coincides with the semantics of [8]. In addition, once put into the bilattice framework the notion of stable model, well-founded model, and prudently brave model all become available. As far as we know, these have never been investigated for logic programming with embedded implications and negation as failure. We do no more than present the foundations here — we hope others find the material interesting and take up the investigation.

We begin with syntax, and in this we generally follow [8]. Let L be a fixed language, thus specifying the Herbrand universe once and for all. All programs we consider will be built up in the language L. Let A denote the set of atoms of L (including *true* for convenience, instead of the empty conjunction). Then the *goals*, G, and the *clauses*, D, are specified as follows:

$$\begin{array}{lcl} G & := & A \mid \neg A \mid G \wedge G \mid D \Rightarrow G \\ D & := & A \leftarrow G \mid D \wedge D \end{array}$$

In [8] universal quantification is allowed in clauses as well. We have omitted this in the interests of keeping the presentation relatively simple — it is straightforward to add it. Notice that negations are restricted to occur at the atomic level. The arrow $\leftarrow$ is intended to be the 'usual' arrow of logic programming; the arrow $\Rightarrow$ denotes the embedded implication we are specially interested in.

A program is a finite set of clauses. When considering programs we identify a conjunction of clauses with the set consisting of those clauses.

Now, let $\mathcal{B}$ be a bilattice of truth values meeting all the conditions of the previous section: it should be infinitely distributive, satisfy the infinitary interlacing conditions, and have a negation and a conflation that commute. Keep the Belnap four-valued logic in mind as a specific example. The general set-up now is inherently more complex than when embedded implications are not considered. A goal of the form $D \Rightarrow G$ is taken to mean we should attempt to establish the goal G, but with respect to the enlarged program resulting from the addition of the clause D. Thus we need to keep track of which program we are working with — it is not fixed once and for all.

Definition 7.1 $\mathbb{P}$ is the set of all programs in the language L. $[\mathbb{P} \rightarrow \mathcal{B}]$ is the space of all functions from $\mathbb{P}$ to the bilattice $\mathcal{B}$. It is given two pointwise orderings: $f \leq_k g$ if $f(\mathcal{P}) \leq_k g(\mathcal{P})$ for all $\mathcal{P} \in \mathbb{P}$, and similarly for $\leq_t$.

It is easy to check that $[\mathbb{P} \rightarrow \mathcal{B}]$ inherits the properties of $\mathcal{B}$; that is, it is a distributive bilattice since $\mathcal{B}$ is, it is complete, etc. Think of $[\mathbb{P} \rightarrow \mathcal{B}]$ as the space of *induced* truth values; it is to this space that valuations will map.

Definition 7.2 A *valuation* is a mapping v from ground atoms to members of $[\mathbb{P} \rightarrow \mathcal{B}]$. That is,

$$v : \mathit{atoms} \rightarrow [\mathbb{P} \rightarrow \mathcal{B}].$$

It is required that $v(\mathit{true})$ is the largest member of $[\mathbb{P} \rightarrow \mathcal{B}]$ under the $\leq_t$ ordering. (This amounts to saying that $v(\mathit{true})(\mathcal{P}) = \mathit{true}$ for all $\mathcal{P} \in \mathbb{P}$, where *true* is the largest member of $\mathcal{B}$ under $\leq_t$.)

What we call a valuation is called an interpretation in [8]. The difference in terminology is not important. What is important is the notion that valuations have to take all programs into account. This is the underlying idea in all versions of semantics allowing embedded implications. Next we need the notion of extending a program by adding a clause, and we need this reflected in the machinery of valuations.

Definition 7.3 Let D be a clause, and v be a valuation. We define a new valuation, v_{+D}, as follows:

$$v_{+D}(A) = \lambda\mathcal{P}.v(A)(\mathcal{P} \cup \{D\})$$

Definition 7.4 The action of valuations is extended to all goals in a straightforward way.

1. $v(\neg G) = \neg v(G)$ where the $\neg$ on the right is the negation of $[\mathbb{P} \rightarrow \mathcal{B}]$.

2. $v(G_1 \wedge G_2) = v(G_1) \wedge v(G_2)$ where the $\wedge$ on the right is the meet of $[\mathbb{P} \to \mathcal{B}]$ under $\leq_t$.

3. $v(D \Rightarrow G) = v_{+D}(G)$.

The following basic monotonicity condition can be shown, using structural induction: $v_1 \leq_k v_2 \Longrightarrow v_1(G) \leq_k v_2(G)$ for any goal G.

Now a mapping Φ is defined, sending valuations to valuations. Note, we still use $\mathcal{P}^*$ for the set of all ground instances of program $\mathcal{P}$ but over the fixed language L. $\Phi(v)$ is the valuation determined as follows:

- $\Phi(v)(A) = \lambda\mathcal{P}.\bigvee\{v(G) \mid A \leftarrow G \in \mathcal{P}^*\}$, where $\bigvee$ denotes the sup in the $\leq_t$ ordering.

Note: we assume an atom A in $\mathcal{P}$ is treated as if it were $A \leftarrow \mathit{true}$ in the definition above. Also the join of the empty set is the least element, so if A is not the head of any clause of $\mathcal{P}^*$, $\Phi(v)(A)(\mathcal{P}) = \mathit{false}$.

It follows from the monotonicity result stated earlier that Φ is monotonic in the $\leq_k$ ordering. Since the space of valuations is a complete lattice under $\leq_k$, Φ has a least and a greatest fixed point. The least is the generalization of the Kripke/Kleene semantics, provided one takes $\mathcal{B}$ to be Belnap's logic. This is equivalent to the semantics presented in [8]. If v is this least fixed point, the meaning it assigns to A with respect to program $\mathcal{P}$ is the truth value $v(A)(\mathcal{P})$.

Now the machinery for stable model semantics is exactly as presented before. A notion of pseudo-valuation is introduced:

$$\begin{array}{rcl}(v_1 \triangle v_2)(A) & = & v_1(A) \\ (v_1 \triangle v_2)(\neg A) & = & \neg v_2(A)\end{array}$$

Then a two-input version of the single-step operator is defined, more or less as earlier in Definition 4.2.

Definition 7.5 Let v_1, v_2 be valuations; $\Psi_{\mathcal{P}}(v_1, v_2)$ is the valuation such that:

- $\Psi_{\mathcal{P}}(v_1, v_2)(A) = \lambda\mathcal{P}.\bigvee\{(v_1 \triangle v_2)(G) \mid A \leftarrow G \in \mathcal{P}^*\}$.

It is not hard to show that:

1. Ψ is monotone in both inputs, under the $\leq_k$ ordering.

2. Ψ is monotone in the first input, under the $\leq_t$ ordering.

3. Ψ is anti-monotone in the second input, under the $\leq_t$ ordering.

Further, note that $\Phi(v) = \Psi(v, v)$. These monotonicity conditions are all that were used in [5] for the development of stable model semantics. Consequently the entire machinery of that paper applies even allowing embedded implications and negation as failure. In addition, the notion of a prudently brave model applies because of the observations in Section 6.

We do not carry the investigation of semantics for embedded implications any further than the mere basics here. We feel there is much to be explored and urge others to take up the investigation. We conclude with a small propositional example, to get things started. Consider the following program:

$$\begin{array}{lcl} \mathtt{A} & \leftarrow & (((\mathtt{B} \leftarrow \neg\mathtt{C}) \wedge (\mathtt{C} \leftarrow \neg\mathtt{B})) \Rightarrow \mathtt{A}) \\ \mathtt{A} & \leftarrow & \mathtt{B} \\ \mathtt{A} & \leftarrow & \mathtt{C} \end{array}$$

Take as the space of truth values Belnap's four-valued logic. Let the program be denoted $\mathcal{P}$, and let $\mathcal{Q}$ be the program $\mathcal{P}$ with the additional clauses $\mathtt{B} \leftarrow \neg\mathtt{C}$ and $\mathtt{C} \leftarrow \neg\mathtt{B}$. For convenience, if v is a valuation, we write $v(\mathtt{A}, \mathcal{P})$ for $v(\mathtt{A})(\mathcal{P})$, and similarly for other cases. Then v is a stable valuation where:

$$\begin{array}{rcl} v(\mathtt{A}, \mathcal{P}) & = & \mathit{true} \\ v(\mathtt{B}, \mathcal{P}) = v(\mathtt{C}, \mathcal{P}) & = & \bot \\ v(\mathtt{A}, \mathcal{Q}) = v(\mathtt{B}, \mathcal{Q}) & = & \mathit{true} \\ v(\mathtt{C}, \mathcal{Q}) & = & \mathit{false} \end{array}$$

There is another one with the roles of `B` and `C` reversed. Note that in these stable models, neither `B` nor `C` have classical truth values with respect to $\mathcal{P}$, though `A` does.

References

[1] Belnap, Jr., N. D. A useful four-valued logic. In *Modern Uses of Multiple-Valued Logic*, J. M. Dunn and G. Epstein, Eds. D. Reidel, 1977.

[2] Fitting, M. C. Notes on the mathematical aspects of Kripke's theory of truth. *Notre Dame Journal of Formal Logic 27* (1986), 75–88.

[3] Fitting, M. C. Bilattices and the semantics of logic programming. *Journal of Logic Programming 11* (1991), 91–116.

[4] Fitting, M. C. Kleene's logic, generalized. *Journal of Logic and Computation 1* (1992), 797–810.

[5] Fitting, M. C. The family of stable models. *Journal of Logic Programming 17* (1993), 197–225.

[6] Gelfond, M., and Lifschitz, V. The stable model semantics for logic programming. In *Proc. of the Fifth Logic Programming Symposium* (Cambridge, MA, 1988), R. Kowalski and K. Bowen, Eds., MIT Press, pp. 1070–1080.

[7] Ginsberg, M. L. Multivalued logics: a uniform approach to reasoning in artificial intelligence. *Computational Intelligence 4* (1988), 265–316.

[8] Giordano, L., and Olivetti, N. Negation as failure in intuitionistic logic programming. In *Logic Programming, Proceedings of the Joint International Conference and Symposium* (1992), MIT Press, pp. 430–445.

[9] Kripke, S. Outline of a theory of truth. *The Journal of Philosophy 72* (1975), 690–716. Reprinted in *New Essays on Truth and the Liar Paradox*, R. L. Martin, ed., Oxford (1983).

[10] Manna, Z., and Shamir, A. The optimal approach to recursive programs. *Comm. ACM 20* (1977), 824–831.

[11] McCarty, L. T. Clausal intuitionistic logic I. fixed point semantics. *Journal of Logic Programming 5* (1988), 1–31.

[12] McCarty, L. T. Clausal intuitionistic logic II. tableau proof procedures. *Journal of Logic Programming 5* (1988), 93–132.

[13] Miller, D. A theory of modules for logic programming. In *IEEE Symposium on Logic Programming* (1986), pp. 106–114.

[14] Przymusinski, T. C. Well-founded semantics coincides with three-valued stable-semantics. *Fundamenta Informaticae 13* (1990), 445–463.

[15] Van Gelder, A. The alternating fixpoint of logic programs with negation. In *Proc. 8th ACM Symp. on Principles of Database Systems* (Philadelphia, 1989), ACM, pp. 1–10.