A Functional View of Strong Negation in Answer Set Programming

Michael Bartholomew and Joohyung Lee

School of Computing, Informatics, and Decision Systems Engineering
Arizona State University, Tempe, USA

Abstract. The distinction between strong negation and default negation has been useful in answer set programming. We present an alternative account of strong negation, which lets us view strong negation in terms of the functional stable model semantics by Bartholomew and Lee. More specifically, we show that, under complete interpretations, minimizing both positive and negative literals in the traditional answer set semantics is essentially the same as ensuring the uniqueness of Boolean function values under the functional stable model semantics. The same account lets us view Lifschitz's two-valued logic programs as a special case of the functional stable model semantics. In addition, we show how non-Boolean intensional functions can be eliminated in favor of Boolean intensional functions, and furthermore can be represented using strong negation, which provides a way to compute the functional stable model semantics using existing ASP solvers. We also note that similar results hold with the functional stable model semantics by Cabalar.

1 Introduction

The distinction between default negation and strong negation has been useful in answer set programming. In particular, it yields an elegant solution to the frame problem. The fact that block b stays at the same location l by inertia can be described by the rule

$$\mathit{On}(b,l,t{+}1) \;\leftarrow\; \mathit{On}(b,l,t),\ \mathit{not}\ \sim\!\mathit{On}(b,l,t{+}1) \tag{1}$$

along with the rule that describes the uniqueness of location values [Lifschitz, 2002],

$$\sim\!\mathit{On}(b,l_1,t) \;\leftarrow\; \mathit{On}(b,l,t),\ l \neq l_1\ . \tag{2}$$

Here '$\sim$' is the symbol for strong negation that represents explicit falsity while '*not*' is the symbol for default negation (negation as failure). Rule (1) asserts that without explicit evidence to the contrary, block b remains at location l. If we are given explicit conflicting information about the location of b at time $t+1$ then this conclusion will be defeated by rule (2), which asserts the uniqueness of location values.

An alternative representation of inertia, which uses choice rules instead of strong negation, was recently presented by Bartholomew and Lee [2012]. Instead of rule (1), they use the choice rule

$$\{\mathit{On}(b,l,t{+}1)\} \;\leftarrow\; \mathit{On}(b,l,t)\ , \tag{3}$$

which states that "if b is at l at time t, then decide arbitrarily whether to assert that b is at l at time $t+1$." Instead of rule (2), they write weaker rules for describing the functional property of *On*:

$$\begin{aligned} &\leftarrow\ \{On(b,l,t) : Location(l)\}0 && \text{(existence of location)} && (4)\\ &\leftarrow\ 2\{On(b,l,t) : Location(l)\} && \text{(uniqueness of location),} && (5)\end{aligned}$$

which can be also combined into one rule: $\leftarrow\ \ not\ 1\{On(b,l,t)\ :\ Location(l)\}1$. In the absence of additional information about the location of block b at time $t+1$, asserting $On(b,l,t+1)$ is the only option, in view of the existence of location constraint (4). But if we are given conflicting information about the location of b at time $t+1$ then not asserting $On(b,l,t+1)$ is the only option, in view of the uniqueness of location constraint (5).

Rules (3), (4), and (5) together can be more succinctly represented in the language of [Bartholomew and Lee, 2012] by means of intensional functions. That is, the three rules can be replaced by one rule

$$\{Loc(b,t+1)=l\}\ \leftarrow\ Loc(b,t)=l\ , \tag{6}$$

where *Loc* is an intensional function constant (the rule reads, "if block b is at location l at time t, by default, the block is at l at time $t+1$"). In fact, Corollary 2 of [Bartholomew and Lee, 2012] tells us how to eliminate intensional functions in favor of intensional predicates, justifying the equivalence between (6) and the set of rules (3), (4), and (5). The translation allows us to compute the language of [Bartholomew and Lee, 2012] using existing ASP solvers, such as SMODELS and GRINGO. However, DLV cannot be used because it does not accept choice rules. On the other hand, all these solvers accept rules (1) and (2), which contain strong negation.

The two representations of inertia involving intensional predicate *On* do not result in the same answer sets. In the first representation, which uses strong negation, each answer set contains only one atom of the form $On(b,l,t)$ for each block b and each time t; for all other locations l', negative literals $\sim On(b,l',t)$ belong to the answer set. On the other hand, such negative literals do not occur in the answer sets of a program that follows the second representation, which yields fewer ground atoms. This difference can be well explained by the difference between the symmetric and the asymmetric views of predicates that Lifschitz described in his message to Texas Action Group, titled "Choice Rules and the Belief-Based View of ASP": [1]

> The way I see it, in ASP programs we use predicates of two kinds, let's call them "symmetric" and "asymmetric." The fact that an object a does not have a property p is reflected by the presence of $\sim p(a)$ in the answer set if p is "symmetric," and by the absence of $p(a)$ if p is "asymmetric." In the second case, the strong negation of p is not used in the program at all.

According to these terminologies, predicate *On* is symmetric in the first representation, and asymmetric in the second representation.

This paper presents several technical results that help us understand the relationship between these two views. In this regard, it helps us to understand strong negation as a way of expressing intensional Boolean functions.

[1] http://www.cs.utexas.edu/users/vl/tag/choice_discussion

– Our first result provides an alternative account of strong negation in terms of Boolean intensional functions. For instance, (1) can be identified with

$$On(b,l,t{+}1)=\text{TRUE} \leftarrow On(b,l,t)=\text{TRUE} \wedge \neg(On(b,l,t{+}1)=\text{FALSE}) \ ,$$

and (2) can be identified with

$$On(b,l_1,t)=\text{FALSE} \leftarrow On(b,l,t)=\text{TRUE} \wedge l \neq l_1 \ .$$

Under complete interpretations, we show that minimizing both positive and negative literals in the traditional answer set semantics is essentially the same as ensuring the uniqueness of Boolean function values under the functional stable model semantics. In this sense, strong negation can be viewed as a mere disguise of Boolean functions.[2]

– We show how non-Boolean intensional functions can be eliminated in favor of Boolean functions. Combined with the result in the first bullet, this tells us a new way of turning the language of [Bartholomew and Lee, 2012] into traditional answer set programs with strong negation, so that system DLV, as well as SMODELS and GRINGO, can be used for computing the language of [Bartholomew and Lee, 2012]. As an example, it tells us how to turn (6) into the set of rules (1) and (2).

– Lifschitz [2012] recently proposed "two-valued logic programs," which modifies the traditional stable model semantics to represent complete information without distinguishing between strong negation and default negation. Using our result that views strong negation in terms of Boolean functions, we show that two-valued logic programs are in fact a special case of the functional stable model semantics in which every function is Boolean.

While the main results are stated for the language of [Bartholomew and Lee, 2012], similar results hold with the language of [Cabalar, 2011] based on the relationship between the two languages studied in [Bartholomew and Lee, 2013]. Furthermore, we note that the complete interpretation assumption in the first bullet can be dropped if we instead refer to the language of [Cabalar, 2011], at the price of introducing partial interpretations.

The paper is organized as follows. In Section 2 we review the two versions of the stable model semantics, one that allows strong negation, but is limited to express intensional predicates only, and the other that allows both intensional predicates and intensional functions. As a special case of the latter we also present multi-valued propositional formulas under the stable model semantics. Section 3 shows how strong negation can be viewed in terms of Boolean functions. Section 4 shows how non-Boolean functions can be eliminated in favor of Boolean functions. Section 5 shows how Lifschitz's two-valued logic programs can be viewed as a special case of the functional stable model semantics. Section 6 shows how strong negation can be represented in the language of [Cabalar, 2011].

2 Preliminaries

2.1 Review: First-Order Stable Model Semantics and Strong Negation

This review follows [Ferraris *et al.*, 2011]. A *signature* is defined as in first-order logic, consisting of *function constants* and *predicate constants*. Function constants of arity 0

[2] It is also well-known that strong negation can be also viewed in terms of auxiliary predicate constants [Gelfond and Lifschitz, 1991].

are also called *object constants*. We assume the following set of primitive propositional connectives and quantifiers: $\bot$ (falsity), $\wedge$, $\vee$, $\rightarrow$, $\forall$, $\exists$. The syntax of a formula is defined as in first-order logic. We understand $\neg F$ as an abbreviation of $F \rightarrow \bot$.

The stable models of a sentence F relative to a list of predicates $\mathbf{p} = (p_1, \dots, p_n)$ are defined via the *stable model operator with the intensional predicates* $\mathbf{p}$, denoted by SM$[F; \mathbf{p}]$. Let $\mathbf{u}$ be a list of distinct predicate variables $u_1, \dots, u_n$ of the same length as $\mathbf{p}$. By $\mathbf{u} = \mathbf{p}$ we denote the conjunction of the formulas $\forall \mathbf{x}(u_i(\mathbf{x}) \leftrightarrow p_i(\mathbf{x}))$, where $\mathbf{x}$ is a list of distinct object variables of the same length as the arity of p_i, for all $i = 1, \dots, n$. By $\mathbf{u} \leq \mathbf{p}$ we denote the conjunction of the formulas $\forall \mathbf{x}(u_i(\mathbf{x}) \rightarrow p_i(\mathbf{x}))$ for all $i = 1, \dots, n$, and $\mathbf{u} < \mathbf{p}$ stands for $(\mathbf{u} \leq \mathbf{p}) \wedge \neg(\mathbf{u} = \mathbf{p})$. For any first-order sentence F, expression SM$[F; \mathbf{p}]$ stands for the second-order sentence

$$F \wedge \neg \exists \mathbf{u}((\mathbf{u} < \mathbf{p}) \wedge F^*(\mathbf{u})),$$

where $F^*(\mathbf{u})$ is defined recursively:

- $p_i(\mathbf{t})^* = u_i(\mathbf{t})$ for any list $\mathbf{t}$ of terms;
- $F^* = F$ for any atomic formula F (including $\bot$ and equality) that does not contain members of $\mathbf{p}$;
- $(F \wedge G)^* = F^* \wedge G^*$; $(F \vee G)^* = F^* \vee G^*$;
- $(F \rightarrow G)^* = (F^* \rightarrow G^*) \wedge (F \rightarrow G)$;
- $(\forall x F)^* = \forall x F^*$; $(\exists x F)^* = \exists x F^*$.

A model of a sentence F (in the sense of first-order logic) is called $\mathbf{p}$-*stable* if it satisfies SM$[F; \mathbf{p}]$.

The traditional stable models of a logic program Π are identical to the Herbrand stable models of the *FOL-representation* of Π (i.e., the conjunction of the universal closures of implications corresponding to the rules).

Ferraris *et al.* [2011] incorporate strong negation into the stable model semantics by distinguishing between intensional predicates of two kinds, *positive* and *negative*. Each negative intensional predicate has the form $\sim\! p$, where p is a positive intensional predicate and '$\sim$' is a symbol for strong negation. In this sense, syntactically $\sim$ is not a logical connective, as it can appear only as a part of a predicate constant. An interpretation of the underlying signature is *coherent* if it satisfies the formula $\neg \exists \mathbf{x}(p(\mathbf{x}) \wedge \sim\! p(\mathbf{x}))$, where $\mathbf{x}$ is a list of distinct object variables, for each negative predicate $\sim\! p$. We consider coherent interpretations only.

Example 1 *The following is a representation of the Blocks World in the syntax of logic programs:*

$$\begin{array}{rcll}
\bot & \leftarrow & \mathit{On}(b_1, b, t), \mathit{On}(b_2, b, t) & (b_1 \neq b_2) \\
\mathit{On}(b, l, t+1) & \leftarrow & \mathit{Move}(b, l, t) & \\
\bot & \leftarrow & \mathit{Move}(b, l, t), \mathit{On}(b_1, b, t) & \\
\bot & \leftarrow & \mathit{Move}(b, b_1, t), \mathit{Move}(b_1, l, t) & \\
\mathit{On}(b, l, 0) & \leftarrow & \mathit{not}\ \sim\!\mathit{On}(b, l, 0) & \\
\sim\!\mathit{On}(b, l, 0) & \leftarrow & \mathit{not}\ \mathit{On}(b, l, 0) & \\
\mathit{Move}(b, l, t) & \leftarrow & \mathit{not}\ \sim\!\mathit{Move}(b, l, t) & \\
\sim\!\mathit{Move}(b, l, t) & \leftarrow & \mathit{not}\ \mathit{Move}(b, l, t) & \\
\mathit{On}(b, l, t+1) & \leftarrow & \mathit{On}(b, l, t), \mathit{not}\ \sim\!\mathit{On}(b, l, t+1) & \\
\sim\!\mathit{On}(b, l, t) & \leftarrow & \mathit{On}(b, l_1, t) & (l \neq l_1)\,.
\end{array} \tag{7}$$

Here On and Move are intensional predicate constants, b, b_1, b_2 are variables ranging over the blocks, l, l_1 are variables ranging over the locations (blocks and the table), and t is a variable ranging over the timepoints. The first rule asserts that at most one block can be on another block. The next three rules describe the effect and preconditions of action Move. The next four rules describe that fluent On is initially exogenous, and action Move is exogenous at each time. The next rule describes inertia, and the last rule asserts that a block can be at most at one location.

2.2 Review: The Functional Stable Model Semantics

The functional stable model semantics is defined by modifying the semantics in the previous section to allow "intensional" functions [Bartholomew and Lee, 2012]. For predicate symbols (constants or variables) u and c, we define $u \le c$ as $\forall \mathbf{x}(u(\mathbf{x}) \rightarrow c(\mathbf{x}))$. We define $u = c$ as $\forall \mathbf{x}(u(\mathbf{x}) \leftrightarrow c(\mathbf{x}))$ if u and c are predicate symbols, and $\forall \mathbf{x}(u(\mathbf{x}) = c(\mathbf{x}))$ if they are function symbols.

Let $\mathbf{c}$ be a list of distinct predicate and function constants and let $\widehat{\mathbf{c}}$ be a list of distinct predicate and function variables corresponding to $\mathbf{c}$. We call members of $\mathbf{c}$ *intensional* constants. By $\mathbf{c}^{pred}$ we mean the list of the predicate constants in $\mathbf{c}$, and by $\widehat{\mathbf{c}}^{pred}$ the list of the corresponding predicate variables in $\widehat{\mathbf{c}}$. We define $\widehat{\mathbf{c}} < \mathbf{c}$ as $(\widehat{\mathbf{c}}^{pred} \le \mathbf{c}^{pred}) \wedge \neg(\widehat{\mathbf{c}} = \mathbf{c})$ and SM$[F;\ \mathbf{c}]$ as

$$F \wedge \neg\exists\widehat{\mathbf{c}}(\widehat{\mathbf{c}} < \mathbf{c} \wedge F^*(\widehat{\mathbf{c}})),$$

where $F^*(\widehat{\mathbf{c}})$ is defined the same as the one in Section 2.1 except for the base case:

– When F is an atomic formula, F^* is $F' \wedge F$, where F' is obtained from F by replacing all intensional (function and predicate) constants in it with the corresponding (function and predicate) variables.

If $\mathbf{c}$ contains predicate constants only, this definition of a stable model reduces to the one in [Ferraris *et al.*, 2011], also reviewed in Section 2.1.

According to [Bartholomew and Lee, 2012], a *choice formula* $\{F\}$ is an abbreviation of the formula $F \vee \neg F$, which is also strongly equivalent to $\neg\neg F \rightarrow F$. A formula $\{\mathbf{t} = \mathbf{t}'\}$, where $\mathbf{t}$ contains an intensional function constant and $\mathbf{t}'$ does not, represents that $\mathbf{t}$ takes the value $\mathbf{t}'$ by default, as the following example demonstrates.

Example 2 *Let F_1 be $\{f = 1\}$, which stands for $(f = 1) \vee \neg(f = 1)$, and I_1 be an interpretation such that $I_1(f) = 1$. Let's assume that we consider only interpretations that map numbers to themselves. I_1 is an f-stable model of F_1: $F_1^*(\widehat{f})$ is equivalent to $((\widehat{f}=1) \wedge (f=1)) \vee \neg(f=1)$,[3] which is further equivalent to $(\widehat{f}=1)$ under the assumption I_1. It is not possible to satisfy this formula by assigning $\widehat{f}$ a different value from $I_1(f)$. On the other hand, I_2 such that $I_2(f) = 2$ is not f-stable since $F_1^*(\widehat{f})$ is equivalent to $\top$ under I_2, so that it is possible to satisfy this formula by assigning $\widehat{f}$ a different value from $I_2(f)$. If we let F_2 be $\{f = 1\} \wedge (f = 2)$, then I_2 is a f-stable of F_2, but I_1 is not: $F_2^*(\widehat{f})$ is equivalent to $\widehat{f}=2$ under I_2, so that $\widehat{f}$ has to map to 2 as well. This example illustrates the nonmonotonicity of the semantics.*

[3] It holds that $(\neg F)^*$ is equivalent to $\neg F$.

Example 3 *The Blocks World can be described in this language as follows. For readability, we write in a logic program like syntax:*

$$\begin{array}{rcl}
\bot & \leftarrow & Loc(b_1,t)\!=\!b \wedge Loc(b_2,t)\!=\!b \wedge (b_1 \neq b_2)\\
Loc(b,t\!+\!1)\!=\!l & \leftarrow & Move(b,l,t)\\
\bot & \leftarrow & Move(b,l,t) \wedge Loc(b_1,t)\!=\!b\\
\bot & \leftarrow & Move(b,b_1,t) \wedge Move(b_1,l,t)\\
\{Loc(b,0)\!=\!l\} & & \\
\{Move(b,l,t)\} & & \\
\{Loc(b,t\!+\!1)\!=\!l\} & \leftarrow & Loc(b,t)\!=\!l\ .
\end{array}$$

Here Loc is a function constant. The last rule is a default formula that describes the commonsense law of inertia. The stable models of this program are the models of SM$[F;\ Loc, Move]$, *where F is the FOL-representation of the program.*

2.3 Review: Stable Models of Multi-Valued Propositional Formulas

The following is a review of the stable model semantics of multi-valued propositional formulas from [Bartholomew and Lee, 2012], which can be viewed as a special case of the functional stable model semantics in the previous section.

The syntax of multi-valued propositional formulas is given in [Ferraris *et al.*, 2011]. A *multi-valued propositional signature* is a set σ of symbols called *constants*, along with a nonempty finite set $Dom(c)$ of symbols, disjoint from σ, assigned to each constant c. We call $Dom(c)$ the *domain* of c. A *Boolean* constant is one whose domain is the set $\{\text{TRUE}, \text{FALSE}\}$. An *atom* of a signature σ is an expression of the form $c\!=\!v$ ("the value of c is v") where $c \in \sigma$ and $v \in Dom(c)$. A *(multi-valued propositional) formula* of σ is a propositional combination of atoms.

A *(multi-valued propositional) interpretation* of σ is a function that maps every element of σ to an element of its domain. An interpretation I *satisfies* an atom $c\!=\!v$ (symbolically, $I \models c\!=\!v$) if $I(c) = v$. The satisfaction relation is extended from atoms to arbitrary formulas according to the usual truth tables for the propositional connectives. I is a *model* of a formula if it satisfies the formula.

The reduct F^I of a multi-valued propositional formula F relative to a multi-valued propositional interpretation I is the formula obtained from F by replacing each maximal subformula that is not satisfied by I with $\bot$. Interpretation I is a *stable model* of F if I is the only interpretation satisfying F^I.

Example 4 *Similar to Example 2, consider the signature $\sigma = \{f\}$ such that $Dom(c) = \{1,2,3\}$. Let I_1 be an interpretation such that $I_1(c) = 1$, and I_2 be such that $I_2(c) = 2$. Recall that $\{f\!=\!1\}$ is shorthand for $(f\!=\!1) \vee \neg(f\!=\!1)$. The reduct of this formula relative to I_1 is $(f\!=\!1) \vee \bot$, and I_1 is the only model of the reduct. On the other hand, the reduct of $\{f=1\}$ relative to I_2 is $(\bot \vee \neg\bot)$ and I_2 is not its unique model. Also, the reduct of $\{f=1\} \wedge (f=2)$ relative to I_1 is $(\bot \vee \neg\bot) \wedge \bot$ and I_1 is not a model. The reduct of $\{f=1\} \wedge (f=2)$ relative to I_2 is $(\bot \vee \neg\bot) \wedge (f=2)$, and I_2 is the only model of the reduct.*

3 Relating Strong Negation to Boolean Functions

3.1 Representing Strong Negation in Multi-Valued Propositional Formulas

Given a traditional propositional logic program Π of a signature σ [Gelfond and Lifschitz, 1991], we identify σ with the multi-valued propositional signature whose constants are the same symbols from σ and every constant is Boolean. By Π^{mv} we mean the multi-valued propositional formula that is obtained from Π by replacing negative literals of the form $\sim\!p$ with $p = \text{FALSE}$ and positive literals of the form p with $p = \text{TRUE}$.

We say that a set X of literals from σ is *complete* if, for each atom $a \in \sigma$, either a or $\sim a$ is in X. We identify a complete set of literals from σ with the corresponding multi-valued propositional interpretation.

Theorem 1 *A complete set of literals is an answer set of Π in the sense of [Gelfond and Lifschitz, 1991] iff it is a stable model of Π^{mv} in the sense of [Bartholomew and Lee, 2012].*

The theorem tells us that checking the minimality of positive and negative literals under the traditional stable model semantics is essentially the same as checking the uniqueness of corresponding function values under the stable model semantics from [Bartholomew and Lee, 2012].

Example 5 *Consider the program that describes a simple transition system consisting of two states depending on whether fluent p is true or false, and an action that makes p true (subscripts 0 and 1 represent time stamps).*

$$
\begin{array}{rcl@{\qquad\qquad}rcl}
p_0 & \leftarrow & \mathit{not}\ \sim\!p_0 & p_1 & \leftarrow & a \\
\sim\!p_0 & \leftarrow & \mathit{not}\ p_0 & & & \\
 & & & p_1 & \leftarrow & p_0, \mathit{not}\ \sim\!p_1 \\
a & \leftarrow & \mathit{not}\ \sim\!a & \sim\!p_1 & \leftarrow & \sim\!p_0, \mathit{not}\ p_1\ . \\
\sim\!a & \leftarrow & \mathit{not}\ a & & &
\end{array}
\qquad (8)
$$

The program has four answer sets, each of which corresponds to one of the four edges of the transition system. For instance, $\{\sim p_0, a, p_1\}$ is an answer set. This program can be encoded in the input languages of GRINGO *and* DLV*. In the input language of* DLV*, which allows disjunctions in the head of a rule, the four rules in the first column can be succinctly replaced by*

$$p_0 \vee \sim\!p_0 \qquad\qquad a \vee \sim\!a\ .$$

According to Theorem 1, the stable models of this program are the same as the stable models of the following multi-valued propositional formula (written in a logic program style syntax; '$\neg$' represents default negation):

$$
\begin{array}{rcl@{\qquad\qquad}rcl}
p_0 = \text{TRUE} & \leftarrow & \neg(p_0 = \text{FALSE}) & p_1 = \text{TRUE} & \leftarrow & a = \text{TRUE} \\
p_0 = \text{FALSE} & \leftarrow & \neg(p_0 = \text{TRUE}) & & & \\
 & & & p_1 = \text{TRUE} & \leftarrow & p_0 = \text{TRUE} \wedge \neg(p_1 = \text{FALSE}) \\
a = \text{TRUE} & \leftarrow & \neg(a = \text{FALSE}) & p_1 = \text{FALSE} & \leftarrow & p_0 = \text{FALSE} \wedge \neg(p_1 = \text{TRUE})\ . \\
a = \text{FALSE} & \leftarrow & \neg(a = \text{TRUE}) & & &
\end{array}
$$

3.2 Relation among Strong Negation, Default Negation, Choice Rules and Boolean Functions

In certain cases, strong negation can be replaced by default negation, and furthermore the expression can be rewritten in terms of choice rules, which often yields a succinct representation.

The following theorem, which extends the *Theorem on Double Negation* from [Ferraris *et al.*, 2009] to allow intensional functions, presents a condition under which equivalent transformations in classical logic preserve stable models.

Theorem 2 *Let F be a sentence, let $\mathbf{c}$ be a list of predicate and function constants, and let I be a (coherent) interpretation. Let F' be the sentence obtained from F by replacing a subformula $\neg H$ with $\neg H'$ such that $I \models \widetilde{\forall}(H \leftrightarrow H')$. Then*

$$I \models \mathrm{SM}[F;\ \mathbf{c}] \textit{ iff } I \models \mathrm{SM}[F';\ \mathbf{c}]\ .$$

We say that an interpretation is *complete* on a predicate p if it satisfies $\forall \mathbf{x}(p(\mathbf{x}) \vee \sim p(\mathbf{x}))$. It is clear that, for any complete interpretation I, we have $I \models \sim p(\mathbf{t})$ iff $I \models \neg p(\mathbf{t})$. This fact allows us to use Theorem 2 to replace strong negation occurring in H with default negation.

Example 5 continued *Each answer set of the first program in Example 5 is complete. In view of Theorem 2, the first two rules can be rewritten as $p_0 \leftarrow$ not not p_0 and $\sim p_0 \leftarrow$ not not $\sim p_0$, which can be further abbreviated as choice rules $\{p_0\}$ and $\{\sim p_0\}$. Consequently, the whole program can be rewritten using choice rules as*

$$\begin{array}{ll}
\{p_0\} & p_1 \leftarrow a \\
\{\sim p_0\} & \\
 & \{p_1\} \leftarrow p_0 \\
\{a\} & \{\sim p_1\} \leftarrow \sim p_0\ . \\
\{\sim a\} &
\end{array}$$

Similarly, since $I \models (p_0 = \text{FALSE})$ iff $I \models \neg(p_0 = \text{TRUE})$, in view of Theorem 2, the first rule of the second program in Example 5 can be rewritten as $p_0 = \text{TRUE} \leftarrow \neg\neg(p_0 = \text{TRUE})$ and further as $\{p_0 = \text{TRUE}\}$. This transformation allows us to rewrite the whole program as

$$\begin{array}{ll}
\{p_0 = B\} & p_1 = \text{TRUE} \leftarrow a = \text{TRUE} \\
\{a = B\} & \{p_1 = B\} \leftarrow p_0 = B\ ,
\end{array}$$

where B ranges over $\{\text{TRUE}, \text{FALSE}\}$. This program represents the transition system more succinctly than program (8).

3.3 Representing Strong Negation by Boolean Functions in the First-Order Case

Theorem 1 can be extended to the first-order case as follows.

Let f be a function constant. A first-order formula is called *f-plain* if each atomic formula

– does not contain f, or
– is of the form $f(\mathbf{t}) = u$ where $\mathbf{t}$ is a tuple of terms not containing f, and u is a term not containing f.

For example, $f = 1$ is f-plain, but each of $p(f)$, $g(f) = 1$, and $1 = f$ is not f-plain.

For a list $\mathbf{c}$ of predicate and function constants, we say that a first-order formula F is $\mathbf{c}$-plain if F is f-plain for each function constant f in $\mathbf{c}$. Roughly speaking, $\mathbf{c}$-plain formulas do not allow the functions in $\mathbf{c}$ to be nested in another predicate or function, and at most one function in $\mathbf{c}$ is allowed in each atomic formula. For example, $f = g$ is not (f, g)-plain, and neither is $f(g) = 1 \rightarrow g = 1$.

Let F be a formula whose signature contains both positive and negative predicate constants p and $\sim p$. Formula $F_b^{(p,\sim p)}$ is obtained from F as follows:

- in the signature of F, replace p and $\sim p$ with a new intensional function constant b of arity n, where n is the arity of p (or $\sim p$), and add two non-intensional object constants TRUE and FALSE;
- replace every occurrence of $\sim p(\mathbf{t})$, where $\mathbf{t}$ is a list of terms, with $b(\mathbf{t}) = \text{FALSE}$, and then replace every occurrence of $p(\mathbf{t})$ with $b(\mathbf{t}) = \text{TRUE}$.

By BC_b ("Boolean Constraint on b") we denote the conjunction of the following formulas, which asserts that b is a Boolean function:

$$\text{TRUE} \neq \text{FALSE} \ , \tag{9}$$

$$\neg\neg\forall\mathbf{x}(b(\mathbf{x}) = \text{TRUE} \lor b(\mathbf{x}) = \text{FALSE}) \ ,$$

where $\mathbf{x}$ is a list of distinct object variables.

Theorem 3 *Let* $\mathbf{c}$ *be a set of predicate and function constants, and let* F *be a* $\mathbf{c}$*-plain formula. Formulas*

$$\forall\mathbf{x}((p(\mathbf{x}) \leftrightarrow b(\mathbf{x}){=}\text{TRUE}) \land (\sim p(\mathbf{x}) \leftrightarrow b(\mathbf{x}){=}\text{FALSE})), \tag{10}$$

and BC_b *entail*

$$\text{SM}[F;\ p, \sim p, \mathbf{c}] \leftrightarrow \text{SM}[F^{(p,\sim p)}_{b};\ b, \mathbf{c}] \ .$$

If we drop the requirement that F be $\mathbf{c}$-plain, the statement does not hold as in the following example demonstrates.

Example 6 *Take* $\mathbf{c}$ *as* (f, g) *and let* F *be* $p(f) \land \sim p(g)$. $F^{(p,\sim p)}_{b}$ *is* $b(f) = \text{TRUE} \land b(g) = \text{FALSE}$. *Consider the interpretation* I *whose universe is* $\{1, 2\}$ *such that* I *contains* $p(1), \sim p(2)$ *and with the mappings* $b^I(1) = \text{TRUE}, b^I(2) = \text{FALSE}, f^I = 1, g^I = 2$. I *certainly satisfies* BC_b *and (10). I also satisfies* $\text{SM}[F;\ p, \sim p, f, g]$ *but does not satisfy* $\text{SM}[F^{(p,\sim p)}_{b};\ b, f, g]$*: we can let* I *be* $\widehat{b}^I(1) = \text{FALSE}, \widehat{b}^I(2) = \text{TRUE}, \widehat{f}^I = 2, \widehat{g}^I = 1$ *to satisfy both* $(\widehat{b}, \widehat{f}, \widehat{g}) < (b, f, g)$ *and* $(F^{(p,\sim p)}_{b})^*(\widehat{b}, \widehat{f}, \widehat{g})$*, which is*

$$b(f) = \text{TRUE} \land \widehat{b}(\widehat{f}) = \text{TRUE} \land b(g) = \text{FALSE} \land \widehat{b}(\widehat{g}) = \text{FALSE}.$$

Note that any interpretation that satisfies both (10) and BC_b is complete on p. Theorem 3 tells us that, for any interpretation I that is complete on p, minimizing the extents of both p and $\sim p$ has the same effect as ensuring that the corresponding Boolean function b have a unique value.

The following corollary shows that there is a 1–1 correspondence between the stable models of F and the stable models of $F^{(p,\sim p)}_{b}$. For any interpretation I of the signature of F that is complete on p, by $I^{(p,\sim p)}_{b}$ we denote the interpretation of the signature of $F^{(p,\sim p)}_{b}$ obtained from I by replacing the relation p^I with function b^I such that

$$\begin{aligned} b^I(\xi_1, \dots, \xi_n) &= \text{TRUE}^I \ \text{ if } \ p^I(\xi_1, \dots, \xi_n) = \text{TRUE}; \\ b^I(\xi_1, \dots, \xi_n) &= \text{FALSE}^I \ \text{ if } \ (\sim p)^I(\xi_1, \dots, \xi_n) = \text{TRUE} \ . \end{aligned}$$

(Notice that we overloaded the symbols TRUE and FALSE: object constants on one hand, and truth values on the other hand.) Since I is complete on p and coherent, b^I is well-defined. We also require that $I^{(p,\sim p)}_{b}$ satisfy (9). Consequently, $I^{(p,\sim p)}_{b}$ satisfies BC_b.

Corollary 1 *Let* $\mathbf{c}$ *be a set of predicate and function constants, and let* F *be a* $\mathbf{c}$*-plain sentence.* (I) *An interpretation* I *of the signature of* F *that is complete on* p *is a model of* $\mathrm{SM}[F;\ p, {\sim}p, \mathbf{c}]$ *iff* $I^{(p,\sim p)}_{b}$ *is a model of* $\mathrm{SM}[F^{(p,\sim p)}_{b};\ b, \mathbf{c}]$. (II) *An interpretation* J *of the signature of* $F^{(p,\sim p)}_{b}$ *is a model of* $\mathrm{SM}[F^{(p,\sim p)}_{b} \wedge BC_b;\ b, \mathbf{c}]$ *iff* $J = I^{(p,\sim p)}_{b}$ *for some model* I *of* $\mathrm{SM}[F;\ p, {\sim}p, \mathbf{c}]$.

The other direction, eliminating Boolean intensional functions in favor of symmetric predicates, is similar as we show in the following.

Let F be a $(b, \mathbf{c})$-plain formula such that every atomic formula containing b has the form $b(\mathbf{t}) = \text{TRUE}$ or $b(\mathbf{t}) = \text{FALSE}$, where $\mathbf{t}$ is any list of terms (not containing members from $(b, \mathbf{c})$). Formula $F^{b}_{(p,\sim p)}$ is obtained from F as follows:

- in the signature of F, replace b with predicate constants p and ${\sim}p$, whose arities are the same as that of b;
- replace every occurrence of $b(\mathbf{t}) = \text{TRUE}$, where $\mathbf{t}$ is any list of terms, with $p(\mathbf{t})$, and $b(\mathbf{t}) = \text{FALSE}$ with ${\sim}p(\mathbf{t})$.

Theorem 4 *Let* $\mathbf{c}$ *be a set of predicate and function constants, let* b *be a function constant, and let* F *be a* $(b, \mathbf{c})$*-plain formula such that every atomic formula containing* b *has the form* $b(\mathbf{t}) = \text{TRUE}$ *or* $b(\mathbf{t}) = \text{FALSE}$. *Formulas (10) and* BC_b *entail*

$$\mathrm{SM}[F;\ b, \mathbf{c}] \leftrightarrow \mathrm{SM}[F^{b}_{(p,\sim p)};\ p, {\sim}p, \mathbf{c}]\ .$$

The following corollary shows that there is a 1–1 correspondence between the stable models of F and the stable models of $F^{b}_{(p,\sim p)}$. For any interpretation I of the signature of F that satisfies BC_b, by $I^{b}_{(p,\sim p)}$ we denote the interpretation of the signature of $F^{b}_{(p,\sim p)}$ obtained from I by replacing the function b^I with predicate p^I such that

$$\begin{array}{l} p^I(\xi_1, \dots, \xi_n) = \text{TRUE} \ \text{ iff } \ b^I(\xi_1, \dots, \xi_n) = \text{TRUE}^I\ ; \\ ({\sim}p)^I(\xi_1, \dots, \xi_n) = \text{TRUE} \ \text{ iff } \ b^I(\xi_1, \dots, \xi_n) = \text{FALSE}^I\ . \end{array}$$

Corollary 2 *Let* $\mathbf{c}$ *be a set of predicate and function constants, let* b *be a function constant, and let* F *be a* $(b, \mathbf{c})$*-plain sentence such that every atomic formula containing* b *has the form* $b(\mathbf{t}) = \text{TRUE}$ *or* $b(\mathbf{t}) = \text{FALSE}$. (I) *An interpretation* I *of the signature of* F *is a model of* $\mathrm{SM}[F \wedge BC_b;\ b, \mathbf{c}]$ *iff* $I^{b}_{(p,\sim p)}$ *is a model of* $\mathrm{SM}[F^{b}_{(p,\sim p)};\ p, {\sim}p, \mathbf{c}]$. (II) *An interpretation* J *of the signature of* $F^{b}_{(p,\sim p)}$ *is a model of* $\mathrm{SM}[F^{b}_{(p,\sim p)};\ p, {\sim}p, \mathbf{c}]$ *iff* $J = I^{b}_{(p,\sim p)}$ *for some model* I *of* $\mathrm{SM}[F \wedge BC_b;\ b, \mathbf{c}]$.

An example of this corollary is shown in the next section.

4 Representing Non-Boolean Functions Using Strong Negation

In this section, we show how to eliminate non-Boolean intensional functions in favor of Boolean intensional functions. Combined with the method in the previous section, it gives us a systematic method of representing non-Boolean intensional functions using strong negation.

4.1 Eliminating non-Boolean Functions in Favor of Boolean Functions

Let F be an f-plain formula. Formula F^f_b is obtained from F as follows:

- in the signature of F, replace f with a new boolean intensional function b of arity $n+1$ where n is the arity of f;
- replace each subformula $f(\mathbf{t}) = c$ with $b(\mathbf{t}, c) = \text{TRUE}$.

By UE_b, we denote the following formulas that preserve the functional property:

$$\forall \mathbf{x} yz(y \neq z \wedge b(\mathbf{x}, y) = \text{TRUE} \rightarrow b(\mathbf{x}, z) = \text{FALSE}),$$

$$\neg\neg\forall \mathbf{x} \exists y(b(\mathbf{x}, y) = \text{TRUE}),$$

where $\mathbf{x}$ is a n-tuple of variables and all variables in $\mathbf{x}$, y, and z are pairwise distinct.

Theorem 5 *For any f-plain formula F,*

$$\forall \mathbf{x} y\big((f(\mathbf{x}) = y \leftrightarrow b(\mathbf{x}, y) = \text{TRUE}) \wedge (f(\mathbf{x}) \neq y \leftrightarrow b(\mathbf{x}, y) = \text{FALSE})\big)$$

and $\exists xy(x \neq y)$ entail

$$\text{SM}[F;\ f, \mathbf{c}] \ \leftrightarrow\ \text{SM}[F^f_b \wedge UE_b;\ b, \mathbf{c}]\ .$$

By I^f_b, we denote the interpretation of the signature of F^f_b obtained from I by replacing the mapping f^I with the mapping b^I such that

$$\begin{array}{ll} b^I(\xi_1, \ldots, \xi_n, \xi_{n+1}) = \text{TRUE}^I & \text{if } f^I(\xi_1, \ldots, \xi_n) = \xi_{n+1} \\ b^I(\xi_1, \ldots, \xi_n, \xi_{n+1}) = \text{FALSE}^I & \text{otherwise.} \end{array}$$

Corollary 3 *Let F be an f-plain sentence.* (I) *An interpretation I of the signature of F that satisfies $\exists xy(x \neq y)$ is a model of* $\text{SM}[F;\ f, \mathbf{c}]$ *iff I^f_b is a model of* $\text{SM}[F^f_b \wedge UE_b;\ b, \mathbf{c}]$. (II) *An interpretation J of the signature of F^f_b that satisfies $\exists xy(x \neq y)$ is a model of* $\text{SM}[F^f_b \wedge UE_b;\ b, \mathbf{c}]$ *iff $J = I^f_b$ for some model I of* $\text{SM}[F;\ f, \mathbf{c}]$.

Example 3 continued *In the program in Example 3, we eliminate non-Boolean function Loc in favor of Boolean function On as follows. The last two rules are UE_{On}.*

$$\begin{array}{rcl}
\bot & \leftarrow & \mathit{On}(b_1, b, t) = \text{TRUE} \wedge \mathit{On}(b_2, b, t) = \text{TRUE} \wedge b_1 \neq b_2 \\
\mathit{On}(b, l, t+1) = \text{TRUE} & \leftarrow & \mathit{Move}(b, l, t) \\
\bot & \leftarrow & \mathit{Move}(b, l, t) \wedge \mathit{On}(b_1, b, t) = \text{TRUE} \\
\bot & \leftarrow & \mathit{Move}(b, b_1, t) \wedge \mathit{Move}(b_1, l, t) \\
\{\mathit{On}(b, l, 0) = \text{TRUE}\} & & \\
\{\mathit{Move}(b, l, t)\} & & \\
\{\mathit{On}(b, l, t+1) = \text{TRUE}\} & \leftarrow & \mathit{On}(b, l, t) = \text{TRUE} \\
\mathit{On}(b, l, t) = \text{FALSE} & \leftarrow & \mathit{On}(b, l_1, t) = \text{TRUE} \wedge l \neq l_1 \\
\bot & \leftarrow & \mathit{not}\ \exists l(\mathit{On}(b, l, t) = \text{TRUE})\ .
\end{array}$$

For this program, it is not difficult to check that the last rule is redundant. Indeed, since the second to the last rule is the only rule that has $\mathit{On}(b, l, t) = \text{FALSE}$ in the head, one can check that any model that does not satisfy $\exists l(\mathit{On}(b, l, t) = \text{TRUE})$ is not stable even if we drop the last rule.

Corollary 2 tells us that this program can be represented by an answer set program containing strong negation (with the redundant rule dropped).

$$\begin{array}{rcll}
\bot & \leftarrow & \mathit{On}(b_1,b,t), \mathit{On}(b_2,b,t) & (b_1 \neq b_2) \\
\mathit{On}(b,l,t+1) & \leftarrow & \mathit{Move}(b,l,t) & \\
\bot & \leftarrow & \mathit{Move}(b,l,t), \mathit{On}(b_1,b,t) & \\
\bot & \leftarrow & \mathit{Move}(b,b_1,t), \mathit{Move}(b_1,l,t) & \\
\{\mathit{On}(b,l,0)\} & & & \\
\{\mathit{Move}(b,l,t)\} & & & \\
\{\mathit{On}(b,l,t+1)\} & \leftarrow & \mathit{On}(b,l,t) & \\
\sim\mathit{On}(b,l,t) & \leftarrow & \mathit{On}(b,l_1,t) & (l \neq l_1) .
\end{array} \tag{11}$$

Let us compare this program with program (7). *Similar to the explanation in Example 5 (continued), the 5th and the 7th rules of* (7) *can be represented using choice rules, which are the same as the 5th and the 6th rules of* (11). *The 6th and the 8th rules of* (7) *represent the closed world assumption. We can check that adding these rules to* (11) *extends the answer sets of* (7) *in a conservative way with the definition of the negative literals. This tells us that the answer sets of the two programs are in a 1-1 correspondence.*

As the example explains, non-Boolean functions can be represented using strong negation by composing the two translations, first eliminating non-Boolean functions in favor of Boolean functions as in Corollary 3 and then eliminating Boolean functions in favor of predicates as in Corollary 2. In the following we state this composition.

Let F be an f-plain formula where f is an intensional function constant. Formula F^f_p is obtained from F as follows:

- in the signature of F, replace f with two new intensional predicates p and $\sim p$ of arity $n+1$ where n is the arity of f;
- replace each subformula $f(\mathbf{t}) = c$ with $p(\mathbf{t}, c)$.

By UE_p, we denote the following formulas that preserve the functional property:

$$\forall \mathbf{x} yz(y \neq z \land p(\mathbf{x},y) \rightarrow \sim p(\mathbf{x},z)) \ ,$$
$$\neg\neg \forall \mathbf{x} \exists y \, p(\mathbf{x},y) \ ,$$

where $\mathbf{x}$ is an n-tuple of variables and all variables in $\mathbf{x}, y, z$ are pairwise distinct.

Theorem 6 *For any* $(f,\mathbf{c})$*-plain formula* F*, formulas*

$$\forall \mathbf{x} y(f(\mathbf{x}) = y \leftrightarrow p(\mathbf{x},y)), \ \ \forall \mathbf{x} y(f(\mathbf{x}) \neq y \leftrightarrow \sim p(\mathbf{x},y)), \ \ \exists xy(x \neq y)$$

entail

$$\mathrm{SM}[F;\ f,\mathbf{c}] \leftrightarrow \mathrm{SM}[F^f_p \land \mathit{UE}_p;\ p, \sim p, \mathbf{c}] \ .$$

By $I^f_{(p,\sim p)}$, we denote the interpretation of the signature of $F^f_{(p,\sim p)}$ obtained from I by replacing the function f^I with the relation p^I that consists of the tuples $\langle \xi_1, \ldots, \xi_n, f^I(\xi_1, \ldots, \xi_n) \rangle$ for all $\xi_1, \ldots, \xi_n$ from the universe of I. We then also add the set $(\sim p)^I$ that consists of the tuples $\langle \xi_1, \ldots, \xi_n, \xi_{n+1} \rangle$ for all $\xi_1, \ldots, \xi_n, \xi_{n+1}$ from the universe of I that do not occur in the set p^I.

Corollary 4 *Let F be an $(f, \mathbf{c})$-plain sentence.* (I) *An interpretation I of the signature of F that satisfies $\exists xy(x \neq y)$ is a model of* $\mathrm{SM}[F;\ f, \mathbf{c}]$ *iff $I^{f}_{(p,\sim p)}$ is a model of* $\mathrm{SM}[F^f_p \wedge \mathit{UE}_p;\ p, \sim p, \mathbf{c}]$. (II) *An interpretation J of the signature of F^f_p that satisfies $\exists xy(x \neq y)$ is a model of* $\mathrm{SM}[F^f_p \wedge \mathit{UE}_p;\ p, \sim p, \mathbf{c}]$ *iff $J = I^{f}_{(p,\sim p)}$ for some model I of* $\mathrm{SM}[F;\ f, \mathbf{c}]$.

Theorem 6 and Corollary 4 are similar to Theorem 8 and Corollary 2 from [Bartholomew and Lee, 2012]. The main difference is that the latter statements refer to the constraint called UEC_p that is weaker than UE_p. For instance, the elimination method from [Bartholomew and Lee, 2012] turns the Blocks World in Example 3 into almost the same program as (11) except that the last rule is turned into the constraint UEC_{On}:

$$\leftarrow\ \mathit{On}(b, l, t) \wedge \mathit{On}(b, l_1, t) \wedge l \neq l_1\ . \qquad (12)$$

It is clear that the stable models of $F^f_p \wedge \mathit{UE}_p$ are under the symmetric view, and the stable models of $F^f_p \wedge \mathit{UEC}_p$ are under the asymmetric view. To see how replacing UE_{On} by UEC_{On} turns the symmetric view to the asymmetric view, first observe that adding (12) to program (11) does not affect the stable models of the program. Let's call this program Π. It is easy to see that Π is a conservative extension of the program that is obtained from Π by deleting the rule with $\sim\mathit{On}(b, l, t)$ in the head.

5 Relating to Lifschitz's Two-Valued Logic Programs

Lifschitz [2012] presented a high level definition of a logic program that does not contain explicit default negation, but can handle nonmonotonic reasoning in a similar style as in Reiter's default logic. In this section we show how his formalism can be viewed as a special case of multi-valued propositional formulas under the stable model semantics in which every function is Boolean.

5.1 Review: Two-Valued Logic Programs

Let σ be a signature in propositional logic. A *two-valued rule* is an expression of the form

$$L_0\ \leftarrow\ L_1, \dots, L_n : F \qquad (13)$$

where $L_0, \dots, L_n$ are propositional literals formed from σ and F is a propositional formula of signature σ.

A *two-valued program* Π is a set of two-valued rules. An interpretation I is a function from σ to $\{\text{TRUE}, \text{FALSE}\}$. The *reduct* of a program Π relative to an interpretation I, denoted Π^I, is the set of rules $L_0\ \leftarrow\ L_1, \dots, L_n$ corresponding to the rules (13) of Π for which $I \models F$. Interpretation I is a stable model of Π if it is a minimal model of Π^I.

Example 7

$$a\ \leftarrow\ \ : a, \qquad \neg a\ \leftarrow : \neg a, \qquad b\ \leftarrow\ a\ :\ \top \qquad (14)$$

The reduct of this program relative to $\{a, b\}$ consists of rules a and $b\ \leftarrow\ a$. Interpretation $\{a, b\}$ is the minimal model of the reduct, so that it is a stable model of the program.

As described in [Lifschitz, 2012], if F in every rule (13) has the form of conjunctions of literals, then the two-valued logic program can be turned into a traditional answer set program containing strong negation when we consider complete answer sets only. For instance, program (14) can be turned into

$$a \;\leftarrow\; \textit{not}\ {\sim}a, \qquad {\sim}a \;\leftarrow\; \textit{not}\ a, \qquad b \;\leftarrow\; a\ .$$

This program has two answer sets, $\{a, b\}$ and ${\sim}a$, and only the complete answer set $\{a, b\}$ corresponds to the stable model found in Example 7.

5.2 Translation into SM with Boolean Functions

Given a two-valued logic program Π of a signature σ, we identify σ with the multi-valued propositional signature whose constants are from σ and the domain of every constant is Boolean values $\{\text{TRUE}, \text{FALSE}\}$. For any propositional formula G, $\mathit{Tr}(G)$ is obtained from G by replacing every negative literal ${\sim}A$ with $A = \text{FALSE}$ and every positive literal A with $A = \text{TRUE}$. By $\mathit{tv2sm}(\Pi)$ we denote the multi-valued propositional formula which is defined as the conjunction of

$$\neg\neg \mathit{Tr}(F) \wedge \mathit{Tr}(L_1) \wedge \cdots \wedge \mathit{Tr}(L_n) \rightarrow \mathit{Tr}(L_0)$$

for each rule (13) in Π.

For any interpretation I of σ, we obtain the multi-valued interpretation I' from I as follows. For each atom A in σ,

$$I'(A) = \begin{cases} \text{TRUE} & \text{if } I \models A \\ \text{FALSE} & \text{if } I \models \neg A \end{cases}$$

Theorem 7 *For any two-valued logic program Π, an interpretation I is a stable model of Π in the sense of [Lifschitz, 2012] iff I' is a stable model of tv2sm(Π) in the sense of [Bartholomew and Lee, 2012].*

Example 7 continued *For the program Π in Example 7, tv2sm(Π) is the following multi-valued propositional formula:*

$$\big(\neg\neg(a = \text{TRUE}) \rightarrow a = \text{TRUE}\big) \wedge \big(\neg\neg(a = \text{FALSE}) \rightarrow a = \text{FALSE}\big) \wedge \big(a = \text{TRUE} \rightarrow b = \text{TRUE}\big).$$

According to [Bartholomew and Lee, 2012], this too has only one stable model in which a and b are both mapped to TRUE*, corresponding to the only stable model of Π according to Lifschitz.*

Consider extending the rules (13) to contain variables. It is not difficult to see that the translation $\mathit{tv2sm}(\Pi)$ can be straightforwardly extended to non-ground programs. This accounts for providing the semantics of the first-order extension of two-valued logic programs.

6 Strong Negation and the Cabalar Semantics

There are other stable model semantics of intensional functions. Theorem 5 from [Bartholomew and Lee, 2013] states that the semantics by Bartholomew and Lee [2013] coincides with the semantics by Cabalar [2011] on **c**-plain formulas. Thus several theorems in this note stated for the Bartholomew-Lee semantics hold also under the Cabalar semantics.

A further result holds with the Cabalar semantics since it allows functions to be partial. This provides extensions of Theorem 3 and Corollary 1, which do not require the interpretations to be complete. Below we state this result. Due to lack of space, we refer the reader to [Bartholomew and Lee, 2013] for the definition of CBL, which is the second-order expression used to define the Cabalar semantics.

Similar to BC_b in Section 3.3, by BC'_b we denote the conjunction of the following formulas:

$$\text{TRUE} \neq \text{FALSE}, \tag{15}$$

$$\neg\neg\forall\mathbf{x}(b(\mathbf{x}) = \text{TRUE} \lor b(\mathbf{x}) = \text{FALSE} \lor b(\mathbf{x}) \neq b(\mathbf{x})),$$

where $\mathbf{x}$ is a list of distinct object variables.[4]

Theorem 8 *Let* $\mathbf{c}$ *be a set of predicate constants, and let F be a formula. Formulas*

$$\forall\mathbf{x}((p(\mathbf{x}) \leftrightarrow b(\mathbf{x})=\text{TRUE})\land(\sim p(\mathbf{x}) \leftrightarrow b(\mathbf{x})=\text{FALSE})\land(\neg p(\mathbf{x})\land\neg\sim p(\mathbf{x}) \leftrightarrow b(\mathbf{x}) \neq b(\mathbf{x})),$$

and BC'_b entail [5]

$$\text{SM}[F;\ p, \sim p, \mathbf{c}] \leftrightarrow \text{CBL}[F^{(p,\sim p)}_b;\ b, \mathbf{c}]\ .$$

The following corollary shows that there is a 1–1 correspondence between the stable models of F and the stable models of $F^{(p,\sim p)}_b$.[6] For any interpretation I of the signature of F, by $I^{(p,\sim p)}_b$ we denote the interpretation of the signature of $F^{(p,\sim p)}_b$ obtained from I by replacing the relation p^I with function b^I such that

$$\begin{array}{l} b^I(\boldsymbol{\xi}) = \text{TRUE}^I \ \text{ if } \ p^I(\boldsymbol{\xi}) = \text{TRUE}\ ; \\ b^I(\boldsymbol{\xi}) = \text{FALSE}^I \ \text{ if } \ (\sim p)^I(\boldsymbol{\xi}) = \text{TRUE}\ ; \\ b^I(\boldsymbol{\xi}) = u \ \text{ if } \ p^I(\boldsymbol{\xi}) = (\sim p)^I(\boldsymbol{\xi}) = \text{FALSE}\ . \end{array}$$

Since I is coherent, b^I is well-defined. We also require that $I^{(p,\sim p)}_b$ satisfy (15). Consequently, $I^{(p,\sim p)}_b$ satisfies BC'_b.

Corollary 5 *Let F be a sentence, and let* $\mathbf{c}$ *be a set of predicate constants.* (I) *An interpretation I of the signature of F is a model of* $\text{SM}[F;\ p, \sim p, \mathbf{c}]$ *iff* $I^{(p,\sim p)}_b$ *is a model of* $\text{CBL}[F^{(p,\sim p)}_b;\ b, \mathbf{c}]$. (II) *An interpretation J of the signature of* $F^{(p,\sim p)}_b$ *is a model of* $\text{CBL}[F^{(p,\sim p)}_b \land BC'_b;\ b, \mathbf{c}]$ *iff* $J = I^{(p,\sim p)}_b$ *for some model I of* $\text{SM}[F; p, \sim p, \mathbf{c}]$.

7 Conclusion

In this note, we showed that, under complete interpretations, symmetric predicates using strong negation can be alternatively expressed in terms of Boolean intensional functions in the language of [Bartholomew and Lee, 2012]. They can also be expressed in terms of Boolean intensional functions in the language of [Cabalar, 2011], but without requiring the complete interpretation assumption, at the price of relying on the notion of partial interpretations.

System CPLUS2ASP [Casolary and Lee, 2011; Babb and Lee, 2013] turns action language $\mathcal{C}$+ into answer set programs containing asymmetric predicates. The translation in

[4] Under partial interpretations, $b(\mathbf{t}) \neq b(\mathbf{t})$ is true if $b(\mathbf{t})$ is undefined. See [Cabalar, 2011; Bartholomew and Lee, 2013] for more details.

[5] The entailment is under partial interpretations and satisfaction.

[6] Recall the notation defined in Section 3.3.

this paper that eliminates intensional functions in favor of symmetric predicates provides an alternative method of computing $\mathcal{C}$+ using ASP solvers.

Acknowledgements: We are grateful to Vladimir Lifschitz for bringing attention to this subject, to Gregory Gelfond for useful discussions related to this paper, and to anonymous referees for useful comments. This work was partially supported by the National Science Foundation under Grant IIS-0916116 and by the South Korea IT R&D program MKE/KIAT 2010-TD-300404-001.

References

[Babb and Lee, 2013] Joseph Babb and Joohyung Lee. CPLUS2ASP: Computing action language C+ in answer set programming. In *Proceedings of International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR)*, 2013. To appear.

[Bartholomew and Lee, 2012] Michael Bartholomew and Joohyung Lee. Stable models of formulas with intensional functions. In *Proceedings of International Conference on Principles of Knowledge Representation and Reasoning (KR)*, pages 2–12, 2012.

[Bartholomew and Lee, 2013] Michael Bartholomew and Joohyung Lee. On the stable model semantics for intensional functions. *TPLP*, 2013. To appear.

[Cabalar, 2011] Pedro Cabalar. Functional answer set programming. *TPLP*, 11(2-3):203–233, 2011.

[Casolary and Lee, 2011] Michael Casolary and Joohyung Lee. Representing the language of the causal calculator in answer set programming. In *ICLP (Technical Communications)*, pages 51–61, 2011.

[Ferraris *et al.*, 2009] Paolo Ferraris, Joohyung Lee, Vladimir Lifschitz, and Ravi Palla. Symmetric splitting in the general theory of stable models. In *Proceedings of International Joint Conference on Artificial Intelligence (IJCAI)*, pages 797–803. AAAI Press, 2009.

[Ferraris *et al.*, 2011] Paolo Ferraris, Joohyung Lee, and Vladimir Lifschitz. Stable models and circumscription. *Artificial Intelligence*, 175:236–263, 2011.

[Gelfond and Lifschitz, 1991] Michael Gelfond and Vladimir Lifschitz. Classical negation in logic programs and disjunctive databases. *New Generation Computing*, 9:365–385, 1991.

[Lifschitz, 2002] Vladimir Lifschitz. Answer set programming and plan generation. *Artificial Intelligence*, 138:39–54, 2002.

[Lifschitz, 2012] Vladimir Lifschitz. Two-valued logic programs. In *ICLP (Technical Communications)*, pages 259–266, 2012.