

Stable Models of Multi-Valued Formulas: Partial vs. Total Functions

Michael Bartholomew and Joohyung Lee

School of Computing, Informatics, and Decision Systems Engineering
Arizona State University, Tempe, AZ, USA
{mjbartho, joolee}@asu.edu

Abstract

Recent extensions of the stable model semantics that allow *intensional* functions—functions that can be specified by logic programs using other functions and predicates—can be divided into two groups. One group defines a stable model in terms of minimality on the values of partial functions, and the other defines it in terms of uniqueness on the values of total functions. We show that, in the context of multi-valued formulas, these two different approaches can be reduced to each other, and further, each of them can be viewed in terms of propositional formulas under the stable model semantics. Based on these results, we present a prototype implementation of different versions of functional stable model semantics by using existing answer set solvers.

Introduction

The original stable model semantics (Gelfond and Lifschitz 1988) and many extensions have been restricted to Herbrand models, where the role of functions is quite limited. Recently a few extensions of the stable model semantics were proposed to allow *intensional* functions—functions that can be specified by logic programs using other functions and predicates. Despite the different forms in which these semantics were defined, they can be essentially divided into two groups. One group defines a stable model in terms of “minimality on the values of partial functions” (Cabalar 2011; Balduccini 2013) and the other defines it in terms of “uniqueness on the values of total functions” (Lifschitz 2012; Bartholomew and Lee 2012). While it is known that they coincide on some class of formulas (Bartholomew and Lee 2013), it does not look obvious if they can be reduced to each other in full generality. Further, it is not obvious how mathematical results in answer set programming that have been established in the absence of intensional functions would carry over to these extensions.

In this note, we address such issues in the context of multi-valued formulas—a simple extension of standard propositional formulas where atoms can express functions mapping to finite domains. The convenience of using multi-valued formulas for knowledge representation is demonstrated in the context of nonmonotonic causal theories and action languages $\mathcal{C}+$ (Giunchiglia et al. 2004) and \mathcal{BC} (Lee, Lifschitz,

and Yang 2013). For this paper, multi-valued formulas serve as a simple but useful special case of first-order formulas to compare different extensions of the functional stable model semantics.

The total function based stable model semantics for multi-valued formulas is defined in (Bartholomew and Lee 2012). Here, following (Cabalar 2011; Balduccini 2013), we define the partial function based stable model semantics, which we call the CB-stable model semantics. This is essentially a generalization of the semantics from (Balduccini 2013). We show that multi-valued formulas under these functional stable model semantics can be viewed in terms of propositional formulas under the stable model semantics, with a slight difference to each other. This finding reveals the close relationship between the functional stable model semantics and their relationships to the propositional stable model semantics, and allows us to easily relate the mathematical results established for propositional formulas, such as the theorem on strong equivalence (Lifschitz, Pearce, and Valverde 2001) and the splitting theorem (Ferraris et al. 2009), to multi-valued formulas. In (Lee, Lifschitz, and Yang 2013), action language \mathcal{BC} is defined by a translation to multi-valued formulas and by a translation to logic programs. The equivalence between the two translations follows from our finding.

Given that both versions of the functional stable model semantics can be reduced to the propositional stable model semantics, one may wonder about the relationship between the two versions of the functional stable model semantics. Interestingly, we show that the functional stable model semantics that is based on partial functions can be fully embedded into the one that is based on total functions.

These results provide a way to implement the functional stable model semantics using existing ASP solvers. We present system MVSM based on this idea. The system is essentially a preprocessor to F2LP (Lee and Palla 2009), which in turn is a preprocessor to the ASP grounder GRINGO.

Multi-Valued Formulas under the Stable Model Semantics

Review: Stable Models of Multi-Valued Formulas

A (multi-valued) signature is a finite set σ of symbols called *constants*, along with a finite set $Dom(c)$ of symbols that is disjoint from σ and contains at least two elements, assigned

to each constant c . We call $Dom(c)$ the *domain* of c . A *multi-valued atom* of σ is \perp , or 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) formula* of σ is a propositional combination of multi-valued atoms.

A *(multi-valued) interpretation* of σ is a function that maps every element of σ to an element in its domain. A multi-valued 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. We often identify an interpretation with the set of atoms of σ that are satisfied by I . We say that I is a *(multi-valued) model* of F if it satisfies F .

We understand an expression of the form $c = d$, where both c and d are constants, as an abbreviation for the formula

$$\bigvee_{v \in Dom(c) \cap Dom(d)} (c=v \wedge d=v). \quad (1)$$

Let F be a multi-valued formula of signature σ , and let I be a multi-valued interpretation of σ . The *reduct* of F relative to I (denoted F^\perp) is the formula obtained from F by replacing each (maximal) subformula that is not satisfied by I with \perp . We call I a *multi-valued stable model* of F if I is the only multi-valued interpretation of σ that satisfies F^\perp .

Example 1 Take $\sigma = \{c\}$ and $Dom(c) = \{1, 2, 3\}$, and let F_1 be $c=1 \vee \neg(c=1)$ and let I_i ($i = 1, 2, 3$) be the interpretation that maps c to i . All three interpretations satisfy F_1 , but I_1 is the only stable model of F_1 : the reduct F_1^\perp is $c=1 \vee \perp$, and I_1 is the only model of the reduct; the reduct of F_1 relative to other interpretations is $\perp \vee \neg\perp$, which does not have a unique model.

If we conjoin $c = 2$ with F_1 , we can check that the only stable model is $c = 2$, which illustrates the nonmonotonicity of the semantics.

As shown in Example 1, formulas of the form $F \vee \neg F$ under the stable model semantics are useful for representing the concept of defaults involving functions. We abbreviate $F \vee \neg F$ as $\langle F \rangle$. For example, F_1 above can be written as $\langle c = 1 \rangle$.

Reducing Multi-Valued SM to Propositional SM

In this section we show that the multi-valued stable model semantics can be viewed as a special case of the propositional stable model semantics. Let σ be a multi-valued signature, and let σ^{prop} be the propositional signature consisting of all propositional atoms $c = v$ where $c \in \sigma$ and $v \in Dom(c)$. For example, for σ in Example 1, σ^{prop} is the set $\{c = 1, c = 2, c = 3\}$, where each element is understood as a propositional atom. We identify a multi-valued interpretation of σ with the corresponding set of propositional atoms from σ^{prop} . It is clear that a multi-valued interpretation I of signature σ satisfies a multi-valued formula F iff I satisfies F when F is viewed as a propositional formula of signature σ^{prop} . Also, it is not difficult to show that multi-valued formulas can be turned into standard propositional formulas having the same *models*. Less obvious is whether such a translation exists while keeping same *stable models*. Theorem 1 below shows such a translation.

Given a multi-valued signature σ , by UC_σ (“Uniqueness Constraint”) we denote the conjunction of

$$\bigwedge_{v \neq w \mid v, w \in Dom(c)} \neg(c = v \wedge c = w) \quad (2)$$

for all $c \in \sigma$, and by EC_σ (“Existence Constraint”) we denote the conjunction of

$$\neg\neg \bigvee_{v \in Dom(c)} c = v, \quad (3)$$

for all $c \in \sigma$. By UEC_σ we denote the conjunction of (2) and (3) for all $c \in \sigma$. For instance, in Example 1, UEC_σ is

$$\neg(c=1 \wedge c=2) \wedge \neg(c=2 \wedge c=3) \wedge \neg(c=1 \wedge c=3) \\ \wedge \neg\neg(c=1 \vee c=2 \vee c=3).$$

Theorem 1 Let F be a multi-valued formula of signature σ , which can be also viewed as a propositional formula of signature σ^{prop} .

- If an interpretation I of σ is a multi-valued stable model of F , then I can be viewed as an interpretation of σ^{prop} that is a propositional stable model of $F \wedge UEC_\sigma$ in the sense of (Ferraris 2005).
- If an interpretation I of σ^{prop} is a propositional stable model of $F \wedge UEC_\sigma$ in the sense of (Ferraris 2005), then I can be viewed as an interpretation of σ that is a multi-valued stable model of F .

Note that the presence of $\neg\neg$ in (3) is essential for Theorem 1 to be valid. For instance, consider the signature containing only one constant d whose domain is $\{1, 2\}$ and F to be \top . F has no multi-valued stable models, but $F \wedge \neg(d=1 \wedge d=2) \wedge (d=1 \vee d=2)$ has two propositional stable models: $\{d=1\}$ and $\{d=2\}$.

Multi-Valued Formulas under the CB-Stable Model Semantics

CB-Stable Models of Multi-Valued Formulas

In this section we introduce a variant of the stable model semantics in the previous section, which we call the CB-stable model semantics. Unlike the previous section, this section allows interpretations to be partially defined. That is, some constants might not be mapped to any values. By *complete* interpretations, we mean a special case of partial interpretations where all constants are defined, which can be identified with “total” interpretations in the previous section.

We consider the same syntax of a multi-valued formula as in the previous section. As with total interpretations, a partial interpretation I *satisfies* an atom $c = v$ if $I(c)$ is defined and is mapped to v . This implies that an interpretation that is undefined on c does not satisfy any atom of the form $c = w$ for any $w \in Dom(c)$. As before, it is convenient to identify a partial interpretation I with the set of atoms of σ that are satisfied by this interpretation. For instance, an interpretation of $\sigma = \{c\}$ which is undefined on c is identified with the empty set. Again, the satisfaction relation is extended from atoms to arbitrary formulas according to the usual truth tables

for the propositional connectives. We call I a *model* of F if it satisfies F .

The reduct F^{\perp} is defined to be the same as before. We say that a partial interpretation I is a *CB-stable model* of F if I satisfies F and no proper subset J of I satisfies F^{\perp} .

Example 1 Continued Under the CB-stable model semantics, $\langle c = 1 \rangle$ does not mean that c is mapped to 1 by default. Instead, it means that c can be mapped to 1 or undefined. As before, the reduct F_1^{\perp} relative to $I_1 = \{c = 1\}$ is $c = 1 \vee \perp$, and I_1 is the minimal model of the reduct.¹ Further, for I_0 that leaves c undefined, the reduct $F_1^{I_0}$ is $\perp \vee \neg \perp$, and I_0 is the minimal model of the reduct.

This difference in understanding $\langle F \rangle$ tells us that multi-valued stable models are more convenient for representing the commonsense law of inertia. For instance,

$$Loc_0 = Kitchen \rightarrow \langle Loc_1 = Kitchen \rangle$$

represents under the multi-valued stable model semantics that the location does not change by default, but under the CB-stable model semantics, the location may become unknown as well.

Reducing CB-Stable Models to Propositional SM

Similar to Theorem 1, the following theorem tells us that the CB-stable models of a multi-valued formula can be identified with the stable models of a propositional formula. The only difference is that we impose UC_{σ} in place of UEC_{σ} .

Theorem 2 Let F be a multi-valued formula of signature σ , which can be also viewed as a propositional formula of signature σ^{prop} .

- (a) If a partial interpretation I of σ is a CB-stable model of F , then I can be viewed as an interpretation of σ^{prop} that is a propositional stable model of $F \wedge UC_{\sigma}$ (in the sense of (Ferraris 2005)).
- (b) If an interpretation I of σ^{prop} is a propositional stable model of $F \wedge UC_{\sigma}$ (in the sense of (Ferraris 2005)), then I can be viewed as a partial interpretation of σ that is a CB-stable model of F .

Reducing CB Semantics to Multi-Valued SM

Reducing Multi-Valued SM to CB-Stable Models

The following corollary immediately follows from Theorems 1 and 2. It tells us that the multi-valued stable model semantics can be fully embedded into the CB-stable model semantics.

Corollary 1 For any multi-valued formula F of signature σ and any partial interpretation I , we have that I is a multi-valued stable model of F iff I is a CB-stable model of $F \wedge EC_{\sigma}$.

Unlike the way we treat $c = d$ as an abbreviation of (1), in (Balduccini 2013), it was called a *t-atom*, for which the notion of satisfaction was defined directly: I satisfies $c = d$ if I is defined on both c and d , and maps them to the same

value. This is essentially equivalent to the way we understand $c = d$ as shorthand for formula (1).²

Since I satisfies $c = c$ iff I is defined on c , the assertion in Corollary 1 remains valid when we replace EC_{σ} in the statement with $\neg \neg \left(\bigwedge_{c \in \sigma} c = c \right)$.

Reducing CB-Stable Models to Multi-Valued SM

Any multi-valued stable model of a formula is a CB-stable model, but not vice versa because an incomplete partial interpretation has no direct counterpart as a total interpretation. It may not look obvious how the CB-stable model semantics (based on partial functions) can be reduced to the multi-valued stable model semantics (based on total functions). Nevertheless, we show that it is possible.

Let σ be a multi-valued signature, and let σ^{none} be the signature that is the same as σ except that the domain of each constant has an additional new value NONE. Given a partial interpretation I of σ , by I^{none} we denote an interpretation of σ^{none} that agrees with I on all defined constants, and maps undefined constants to NONE. Recall that expression $\langle F \rangle$ stands for the formula $F \vee \neg F$.

Theorem 3 Let F be a multi-valued formula of signature σ .

- (a) If an interpretation I of σ is a CB-stable model of F , then I^{none} is a stable model of $F \wedge \bigwedge_{c \in \sigma} \langle c = \text{NONE} \rangle$.
- (b) If an interpretation J of σ^{none} is a stable model of $F \wedge \bigwedge_{c \in \sigma} \langle c = \text{NONE} \rangle$ then $J = I^{none}$ for some CB-stable model I of F .

For instance, in Example 1, the CB-stable models of F_1 are in a 1-1 correspondence with the stable models of $F_1 \wedge \langle c = \text{NONE} \rangle$.

System MVSM

System `mvsm`³ is a prototype implementation of multi-valued formulas under the stable model semantics. In fact, it is a script that invokes several software, such as `MVPF2LPCOMPILER`, `F2LP`, `GRINGO`, `CLASPD`, and `AS2TRANSITION`. `MVPF2LPCOMPILER` is an implementation of the translations in Theorem 1 and Theorem 2, which translates multi-valued formulas under the stable model semantics into standard propositional formulas under the stable model semantics. As the theorems show, the translations are very similar, and the user can choose which translation to use. `F2LP` then transforms the propositional formula into an ASP program in the input language of `GRINGO v3`. `AS2TRANSITION` takes the output of `CLASPD` and outputs propositional atoms in the form of multi-valued atoms. The composition of these software is depicted in Figure 1.

Shown below is a description of the blocks world domain in the language of `MVSM` assuming the multi-valued stable model semantics. The syntax of declarations follows the one in the input language of the Causal Calculator v2.⁴ Compared to the usual ASP encoding, explicit declaration of sorts

²In a sense, our treatment is more general because it allows the domains of c and d to be different.

³<http://sourceforge.net/projects/aspmt/>

⁴<http://www.cs.utexas.edu/~tag/cc/>

¹Minimality is understood in terms of set inclusion.

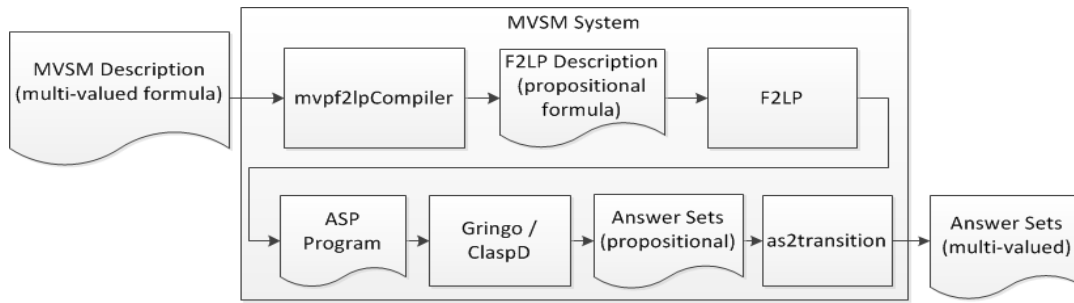


Figure 1: Architecture of MVSM

and type checking help reduce the programmer’s mistakes. The inertia and exogeneity assumptions in the last three rules have simple reading, once we understand $\langle F \rangle$ as representing defaults ($\{ \dots \}$ was used in place of $\langle \dots \rangle$). There is no need to use strong negation.

```

% File 'bw' : The blocks world

:- sorts
    step; astep; location >> block.

:- objects
    0..maxstep           :: step;
    0..maxstep-1        :: astep;
    1..6                 :: block;
    table                :: location.

:- variables
    ST                   :: step;
    T                    :: astep;
    Bool                 :: boolean;
    B,B1                 :: block;
    L                    :: location.

:- constants
    loc(block,step)     :: location;
    move(block,location,astep) :: boolean.

% two blocks can't be on the same block at the same time
<- loc(B1,ST)=B & loc(B2,ST)=B & B1!=B2.

% effect of moving a block
loc(B,T+1)=L <- move(B,L,T) .

% a block can be moved only when it is clear
<- move(B,L,T) & loc(B1,T)=B.

% a block can't be moved onto a block that is being
% moved also
<- move(B,B1,T) & move(B1,L,T) .

% initial location is exogenous
{loc(B,0)=L} .

% actions are exogenous
{move(B,L,T)=Bool} .

% fluents are inertial
{loc(B,T+1)=L} <- loc(B,T)=L.

```

Acknowledgements: This work was partially supported by NSF under Grant IIS-1319794 and by the South Korea IT R&D program MKE/KIAT 2010-TD-300404-001.

References

- Balduccini, M. 2013. ASP with non-Herbrand partial functions: a language and system for practical use. *TPLP* 13(4-5):547–561.
- Bartholomew, M., and Lee, J. 2012. Stable models of formulas with intensional functions. In *Proceedings of International Conference on Principles of Knowledge Representation and Reasoning (KR)*, 2–12.
- Bartholomew, M., and Lee, J. 2013. On the stable model semantics for intensional functions. *TPLP* 13(4-5):863–876.
- Cabalar, P. 2011. Functional answer set programming. *TPLP* 11(2-3):203–233.
- Ferraris, P.; Lee, J.; Lifschitz, V.; and Palla, R. 2009. Symmetric splitting in the general theory of stable models. In *Proceedings of International Joint Conference on Artificial Intelligence (IJCAI)*, 797–803. AAAI Press.
- Ferraris, P. 2005. Answer sets for propositional theories. In *Proceedings of International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR)*, 119–131.
- Gelfond, M., and Lifschitz, V. 1988. The stable model semantics for logic programming. In Kowalski, R., and Bowen, K., eds., *Proceedings of International Logic Programming Conference and Symposium*, 1070–1080. MIT Press.
- Giunchiglia, E.; Lee, J.; Lifschitz, V.; McCain, N.; and Turner, H. 2004. Nonmonotonic causal theories. *Artificial Intelligence* 153(1–2):49–104.
- Lee, J., and Palla, R. 2009. System F2LP – computing answer sets of first-order formulas. In *Proceedings of International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR)*, 515–521.
- Lee, J.; Lifschitz, V.; and Yang, F. 2013. Action language *BC*: Preliminary report. In *Proceedings of International Joint Conference on Artificial Intelligence (IJCAI)*.
- Lifschitz, V.; Pearce, D.; and Valverde, A. 2001. Strongly equivalent logic programs. *ACM Transactions on Computational Logic* 2:526–541.
- Lifschitz, V. 2012. Logic programs with intensional functions. In *Proceedings of International Conference on Principles of Knowledge Representation and Reasoning (KR)*, 24–31.