The Role of Functions in the Stable Model Semantics

Michael Bartholomew

School of Computing, Informatics, and Decision Systems Engineering
Arizona State University, Tempe, USA
(*e-mail:* mjbartho@asu.edu)

submitted 12 May 2013; accepted 5 June 2013

Abstract

The rich modelling capabilities and efficient solvers of ASP have enabled the successful application of ASP to many domains. However, due to an issue called the "grounding bottleneck" that arises especially in numeric domains, ASP still faces some challenges in being applied to such domains. Recent work has considered hybrid approaches, using a combination of ASP and other KR formalisms including Satisfiability Modulo Theories, Constraint Satisfaction Programming, and Linear Programming. However, these are typically loose couplings and are not able to represent non-monotonic constructs in the non-ASP portions of those couplings. We propose a semantics that incorporates "intensional" functions in answer set programming. Intensional functions are those whose values can be described by other functions and predicates, rather than being pre-defined as in the standard answer set programming. We demonstrate that the functional stable model semantics plays an important role in the framework of "Answer Set Programming Modulo Theories (ASPMT)" —a tight integration of answer set programming and satisfiability modulo theories, under which existing integration approaches can be viewed as special cases where the role of functions is limited.

1 Introduction

Modeling complex real-world domains in a way that is both appealing from a knowledge representation standpoint and efficiently computable is a difficult task for several reasons. Many approaches use propositional symbols to represent knowledge which is reasonable for some characteristics but are not ideal for others. For example, the location of a car, the weight of a ball, or the volume of a container are all better described as functions rather than as a set of propositional constants among which only one can be true. Another commonly-faced difficulty is that even in a bounded domain, such as a range of speed for a spacecraft of 0 to 40000 kilometers per hour, a common discretization of integral values may be too coarse of a granularity to aptly model the domain for some plans. Two approaches that have seen some limited recent success are and Satisfiability Modulo Theories (SMT) Answer Set Programming (ASP).

While SMT improves upon SAT in these issues to some extent, one significant missing feature is non-monotonicity. Because of the lack of support for non-monotonic reasoning, expressing concepts such as inertia or default values for functional fluents requires explicit specification of when the inertia or default will not hold, thus making formulations containing these concepts not elaboration tolerant. Due to the non-monotonic nature of the semantics, ASP does not exhibit this weakness.

Though ASP has been successfully applied to many domains, it still suffers a significant decline in performance even in domains that are simple but have a large numeric domain. This poor

performance is due to the process of grounding the variables in the program–making variable-free copies of the rules by replacing variables with all possible combinations of values they can attain. Some optimization is possible to improve computation but as the Gears World example in Table 1 illustrates, ASP still suffers in a relatively simple domain compared to other KR formalisms such as SMT in some cases. However, these other formalisms lack non-monotonicity and are unable to benefit from the same elaboration tolerance and expressions of inertia and defaults that ASP enjoys.

Some attempts to address this poor performance in ASP have been attempted by integrating ASP with other declarative computing paradigms, such as constraint processing, satisfiability modulo theories, or mixed integer programming (Gebser et al. 2009 ; Balduccini 2009 ; Janhunen et al. 2011 ; Liu et al. 2012). However, the nature of these loose couplings mean that the vales of functions are determined by the other declarative computing paradigms rather than by ASP and so these functions lack the non-monotonicity needed for elaboration tolerant inertia and default values for these functions.

To take advantage of the success of these other formulations while still maintaining this non-monotonicity, we propose a modification to the stable model semantics. Several extensions of the stable model semantics have been proposed to allow "intensional" functions—functions that can be described by other functions and predicates (Cabalar 2011 ; Lifschitz 2012 ; Bartholomew and Lee 2012 ; Balduccini 2012b). Such functions significantly enhance the modeling capability of the language of answer set programming by providing natural representations of nonBoolean fluents, such as the location of an object, or the level of a water tank.

Example 1
The following program F describes the capacity of a container of water that has a leak but that can be refilled to the maximum amount, say 10, with the action FillUp.

$$\begin{array}{rcl} \{Amount_1 = x\} & \leftarrow & Amount_0 = x+1 \\ Amount_1 = 10 & \leftarrow & FillUp\ . \end{array} \tag{1}$$

Here $Amount_1$ is an intensional function constant, and x is a variable ranging over nonnegative integers. According to the semantics from (Bartholomew and Lee 2013), the first rule is a default rule (or choice rule) standing for $(Amount_1 = x) \vee \neg(Amount_1 = x) \leftarrow Amount_0 = x+1$, and expresses that the amount at next time decreases by default. However, if FillUp action is executed (if we add FillUp as a fact), this behavior is overridden, and the amount is set to the maximum value.

Recently, (Bartholomew and Lee 2013) showed that functional stable model semantics can be used as a natural basis of combining answer set programming and satisfiability modulo theories. For instance, the paper showed that if a program is "tight," as is the case in the example above, it can be turned into the input language of Satisfiability Modulo Theories (SMT) solvers, thereby allowing efficient constraint satisfaction methods to be applied to alleviate the grounding bottleneck involving functional fluents. For example, program (1) can be turned into the SMT instance

$$(Amount_0 = Amount_1 + 1) \vee (Amount_1 = 10 \wedge FillUp)$$
$$FillUp \rightarrow Amount_1 = 10\ .$$

The advantage of this approach is twofold: we can represent the functional fluents in a more natural way compared to the standard representation of these fluents as relations where we must write constraints to enforce uniqueness and existence so that these relations behave as functions. This approach also benefits from a significant improvement in runtime due to the grounding bottleneck incurred by the standard approach.

Similarly, (Balduccini 2012a) reports the computational efficiency of a system that computes the semantics of intensional functions defined in (Balduccini 2012b). However, the existing semantics of intensional functions were defined in very different styles under different intuitions, which obscures the relationships among them.

In order to fully understand the limitations and strengths of each semantics, a formal study of the relationships between these semantics is needed. Such a study will provide more clear insight into each semantics and will help to identify goals for future research. In addition, many theorems were established for the stable model semantics so a natural question is whether the functional stable model semantics enjoys these same results.

2 Background

2.1 Functional Stable Model Semantics

The functional stable model semantics can be concisely expressed in second-order logic in a way that is a proper extension of the second-order logic definition of the stable model semantics presented in (Ferraris et al. 2011):

Formulas are built the same as in first-order logic. A signature consists of function constants and predicate constants. Function constants of arity 0 are called object constants. We assume the following set of primitive propositional connectives and quantifiers:

$$\bot \text{ (falsity)},\ \wedge,\ \vee,\ \rightarrow,\ \forall,\ \exists\ .$$

We understand $\neg F$ as an abbreviation of $F \rightarrow \bot$; symbol $\top$ stands for $\bot \rightarrow \bot$, and $F \leftrightarrow G$ stands for $(F \rightarrow G) \wedge (G \rightarrow F)$.

For predicate symbols (constants or variables) u and c, we define $u \leq c$ 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

$$(\widehat{\mathbf{c}}^{pred} \leq \mathbf{c}^{pred}) \wedge \neg(\widehat{\mathbf{c}} = \mathbf{c})$$

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 as follows.

- 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;[1]
- $(F \wedge G)^* = F^* \wedge G^*$; $\quad (F \vee G)^* = F^* \vee G^*$;
- $(F \rightarrow G)^* = (F^* \rightarrow G^*) \wedge (F \rightarrow G)$;
- $(\forall x F)^* = \forall x F^*$; $\quad (\exists x F)^* = \exists x F^*$.

When F is a sentence, the models of SM$[F;\ \mathbf{c}]$ are called the $\mathbf{c}$-*stable* models of F. They are the models of F that are "stable" on $\mathbf{c}$.

[1] If an atomic formula F contains no intensional function constants, then F^* can be defined as F', as in (Ferraris et al. 2011).

If $\mathbf{c}$ contains predicate constants only, this definition of a stable model reduces to the one in (Ferraris et al. 2011). The definition of F^* above is the same as in (Ferraris et al. 2011) except for the case when F is an atomic formula.

2.2 Answer Set Programming Modulo Theories

Formally, an SMT instance is a formula in many-sorted first-order logic, where some designated function and predicate constants are constrained by some fixed background interpretation. SMT is the problem of determining whether such a formula has a model that expands the background interpretation (Barrett et al. 2009).

The syntax of ASPMT is the same as that of SMT. Let σ^{bg} be the (many-sorted) signature of the background theory bg. An interpretation of σ^{bg} is called a *background interpretation* if it satisfies the background theory. For instance, in the theory of reals, we assume that σ^{bg} contains the set $\mathcal{R}$ of symbols for all real numbers, the set of arithmetic functions over real numbers, and the set $\{<, >, \leq, \geq\}$ of binary predicates over real numbers. Background interpretations interpret these symbols in the standard way.

Let σ be a signature that is disjoint from σ^{bg}. We say that an interpretation I of σ satisfies F w.r.t. the background theory bg, denoted by $I \models_{bg} F$, if there is a background interpretation J of σ^{bg} that has the same universe as I, and $I \cup J$ satisfies F. For any ASPMT sentence F with background theory σ^{bg}, interpretation I is a stable model of F relative to $\mathbf{c}$ (w.r.t. background theory σ^{bg}) if $I \models_{bg} \text{SM}[F; \mathbf{c}]$.

It should be stressed that these background theories can be quite general and not just restricted to integers, rationals, or reals. Rather, just as SMT allows convenient concepts like bit-vectors and lists, the same background theories could be used to craft ASPMT solvers for these concepts in a similar fashion.

3 Related Work

The work on extending the stable model semantics to include functions can be categorized into two groups, each of which focuses only on one main issue.

One group is interested in enriching the modeling language by incorporating "intensional" functions (Cabalar 2011 ; Lifschitz 2012 ; Bartholomew and Lee 2012 ; Balduccini 2012b). Intensional functions are functions whose values can be described by other functions and predicates, rather than being pre-defined. Such semantics allow us to represent nonBoolean fluents by intensional functions without having to rely on predicates.

While stable model semantics described in (Ferraris et al. 2011) considers the minimality of predicates, the treatment of functions in (Lifschitz 2012 ; Bartholomew and Lee 2012) is instead about uniqueness of functions similar to Causal Logic (Giunchiglia et al. 2004). On the other hand, (Cabalar 2011 ; Balduccini 2012b) consider the minimality of functions by allowing a function to map to a value or to be undefined, which is considered less than mapping to a value. While the semantics treat functions differently and are defined in different ways, they coincide on a surprisingly large class of formula.

The other group of work focuses on improving the computational efficiency by integrating ASP with other declarative computing paradigms, such as constraint processing, satisfiability modulo theories, or mixed integer programming (Gebser et al. 2009 ; Balduccini 2009 ; Janhunen et al. 2011 ; Liu et al. 2012), and exploiting the efficient constraint processing techniques on functions (or called "variables" in CSP/SMT) without having to generate a large set of ground instances. Constraint variables are functions that are mapped to values in their domains. In SMT

with difference logic or linear constraints, arithmetic variables are functions that are mapped to numbers. However, these functions are not as expressive as intensional functions.

4 Research Goals

The first goal of this research is to execute a formal study of the relationships between the various proposals of incorporating more sophisticated functions in the stable model semantics. In doing so, this can help to concentrate research efforts in the case that some work may be closely related or if some defect in one proposal is shown to be absent in another.

The second goal is to investigate the properties of the functional stable model semantics especially concerning useful theorems that have been established for the traditional stable model semantics. Among these are the Theorem on Completion, the Theorem on Strong Negation, Safety, and the Splitting Theorem, each providing useful theoretical or practical properties from which the functional stable model semantics would also benefit.

Finally, the concept of Satisfiability Modulo Thoeries has proven successful in incorporating these functional fluents in the SAT community. Due to the known relationship between ASP and SAT, a natural question is whether ASP can benefit from a similar concept. The final goal is to investigate this question and provide a system that takes advantage of this concept, which we hope will allow more efficient computation of larger theories, especially those that use arithmetic, for which current ASP systems tend to perform poorly due to the grounding bottleneck.

5 Research Progress

This research started with an investigation into the effects Herbrand functions had on the ability to ground a formula with respect to a finite universe despte the formula have an infinite Herbrand Universe. This work (Bartholomew and Lee 2010) led to the identification of the largest decidable class of groundable formulas but even a simple formula like

$$p(1, f(f(1))) \wedge p(X, f(f(X))) \rightarrow p(f(X), f(X)),$$

which is finitely groundable, does not fall into this category. While this example may seem artificial, when we replace the Herbrand function f with the non-Herbrand function that is "increment by one", we get a perhaps more reasonable example

$$p(1, 3) \wedge p(X, X + 2) \rightarrow p(X + 1, X + 1)$$

and trying to better understand what role these functions might have in the stable model semantics led to further investigation.

During this investigation, we considered alternative semantics to the stable model semantics from (Ferraris et al. 2011). In particular, we studied the FLP semantics described in (Faber et al. 2004) and provided a reformulation in second-order logic that yielded a proper generalization of their semantics in order to more easily understand the relationship between the semantics. We also compared our reformulation with the reformulation in (Truszczyński 2009) in order to determine which would be the most reasonable semantics to pursue during the investigation into these functions. This investigation provided additional insight to the relationship between the several extensions of the FLP semantics, circumscription, and the stable model semantics from (Ferraris et al. 2011) and in the end, we decided to focus the investigation using the (Ferraris et al. 2011) semantics.

We concluded that the limitation to Herbrand functions was too strict to support even simple

Instance Size	CLINGO v3.0.5 Execution		iSAT v1.0 Execution		Z3 V4.3.0 Execution	
	Run Time (Grounding + Solving)	Atoms	Run Time	Variables	Run time	Memory
5	.02s (.02s + 0s)	3174	.03s	331	.03s	2.79
10	.3s (.3s + 0s)	10161	.19s	596	.09s	4.91
20	9.46s (4.02s + 5.11s)	36695	.79s	1126	.2s	8.65
30	42.56s (22.32s + 20.24s)	77627	2.05s	1656	.36s	12.22
50	923.74s (297.26 + 626.48s)	207706	14.35s	2716	1.09s	20.35
100	out of memory		494.77s	5366	5.52s	43.86

Table 1. *Gears World Experiment Results*

concepts like arithmetic and also that the prior treatment of functions as being pre-defined or non-intensional was also an undesirable limitation.

To address these limitations, we defined the concept of intensional functions in the context of the stable model semantics in (Bartholomew and Lee 2012) and compared this formalism to IF-programs (Lifschitz 2012). In addition, we performed an initial investigation into the properties of this semantics, an endeavor that yielded an initial extension of the Theorem on Completion to a restricted class ($\mathbf{f}$-plain) of formulas under the functional stable model semantics.

In (Bartholomew and Lee 2013), this investigation was continued as we compared the functional stable model semantics to Clingcon programs (Gebser et al. 2009) and ASP(LC) (Liu et al. 2012) programs. This work also generalized the Completion result for the traditional stable model semantics and the functional stable model semantics by considering a looser notion of a dependency graph. This work, as well as (Lee and Meng 2013), presented the notion of Answer Set Programming Modulo Theories (ASPMT) which can be understood as a special case of functional stable model semantics in which part of the interpretation is fixed so that some symbols are not intensional but rather are predefined. An example of this is ASMPT using the Theory of Reals as a background theory so that the set of real numbers $\mathcal{R}$ as well as the arithmetic and comparison operators are non-intensional and instead have the usual interpretation.

Most recently, we focused the research toward identifying the relationship between the functional stable model semantics and the two proposed semantics in (Balduccini 2012b) and (Cabalar 2011). This work, which was just recently accepted as a full paper in ICLP 2013, identified several large classes for which all three semantics coincide and also revealed that (Balduccini 2012b) and (Cabalar 2011), despite being described in rather different ways, are very similar semantically.

6 Preliminary Experimental Results

(Bartholomew and Lee 2013) presented a small domain–Gears World–for which the theory was hand-translated into an SMT instance and showed that the execution time of the SMT solvers was several orders of magnitude smaller than that of ASP as the instance size grew. The Gears World domain is one in which we have two gears, Gear1 with radius 7 and Gear2 with radius 17. Each gear is connected to a motor that has integral running speeds which can be incremented by 1 using the corresponding action. The gears can also be moved close together so that both gears

spin at the speed of the higher value (between *M1Speed* $\times$ *Radius1* or *M2Speed* $\times$ *Radius2*). The goal is to have Gear1 spinning at a multiple of Gear2's radius. That multiple is the instance size in Table 1, so for example, instance size 3 means at the end, we want Gear1 spinning at a speed of $51(= 3 \times 17)$. This domain can be expressed in the language of ASPMT with the theory of integers. Each system reports space consumption using a different metric: Clingo reports how many unique ground atoms occur in the ground program, iSAT reports how many distinct variables are used in the expanded program, and Z3 reports its memory usage. While these are not directly comparable, they are included in the table to demonstrate the scaling of memory usage as the instance size increases–while iSAT and Z3 both scale linearly with the instance size, Clingo scales superlinearly.

7 Open Issues and Expected Achievements

While some important questions have been addressed already, several important issues are still open questions. There are still several properties of the stable model semantics that have not been shown to hold for the functional stable model semantics (FSM), including the Theorem on Strong Negation, and the Splitting Theorem. In addition the concept of Skolemization has not been addressed. Each of these can provide some useful insight or computational advantage if they hold for the functional stable model semantics.

Further work must be done in the direction of ASPMT to allow for a satisfactory implementation. For instance, the semantics presented does not consider the aggregate construct. We expect that the incorporation of aggregates will not pose a problem. The expectation is that the algorithm used to compute answer sets can be modified to compute ASPMT instances in a way similar to the modification made to the SAT algorithm to compute SMT instances. We expect that this system will allow a more natural representation of functional fluents while also providing a computation method that addresses the poor performance of ASP in large numeric domain by avoiding grounding. We plan to perform several experiments to evaluate the merit of this approach.

What we hope to achieve by this work is to provide a successful framework for integrating KR formalisms through the use of the extension of the stable model semantics to include intensional functions. In doing so, we provide the framework for ASPMT which gives us a tight integration of ASP and SMT in order to take the best of both worlds–the non-monotonicity of ASP and the efficiency (especially in numeric domains) of SMT. This enables more efficient computation of challenging domains as well as currently infeasible reasoning such as reasoning about continuous, rather than discrete, change.

References

Marcello Balduccini. Representing Constraint Satisfaction Problems in Answer Set Programming. In *Working Notes of the Workshop on Answer Set Programming and Other Computing Paradigms (ASPOCP)* 2009.

Marcello Balduccini. An Answer Set Solver for non-Herbrand Programs: Progress Report. In *ICLP (Technical Communications)* 2012. Pages 49-60.

Marcello Balduccini. A "Conservative" Approach to Extending Answer Set Programming with Non-Herbrand Functions. In *Correct Reasoning - Essays on Logic-Based AI in Honour of Vladimir Lifschitz* 2012. Pages 24-39.

Clark W. Barrett, Roberto Sebastiani, Sanjit A. Seshia, and Cesare Tinelli. Satisfiability modulo theories. In Armin Biere, Marijn Heule, Hans van Maaren, and Toby Walsh, editors, *Handbook of Satisfiability*, volume 185 of *Frontiers in Artificial Intelligence and Applications*. Pages 825-885. IOS Press, 2009.

Pedro Cabalar. A Decidable Class of Groundable Formulas in the General Theory of Stable Models. In

Proceedings of International Conference on Principles of Knowledge Representation and Reasoning (KR) 2010. Pages 477-485.

Pedro Cabalar. Stable Models of Formulas with Intensional Functions. In *Proceedings of International Conference on Principles of Knowledge Representation and Reasoning (KR)* 2012. Pages 2-12.

Pedro Cabalar. Functional Stable Model Semantics and Answer Set Programming Modulo Theories[2]. In *Proceedings of International Joint Conference on Artificial Intelligence* 2013. To appear.

Pedro Cabalar. Functional answer set programming. In *Theory and Practice of Logic Programming* 2011, pages 203-233.

Faber, Wolfgang and Leone, Nicola and Pfeifer, Gerald. Recursive aggregates in disjunctive logic programs: Semantics and complexity. In *Proceedings of European Conference on Logics in Artificial Intelligence (JELIA)* 2004, pages 203-233.

Ferraris, Paolo and Lee, Joohyung and Lifschitz, Vladimir. Stable models and circumscription. In *Artificial Intelligence* 2011. Pages 236-263.

M. Gebser and M. Ostrowski and T. Schaub. Constraint Answer Set Solving. In *Proceedings of International Conference on Logic Programming (ICLP)* 2009, pages 235-249.

Giunchiglia, Enrico and Lee, Joohyung and Lifschitz, Vladimir and McCain, Norman and Turner, Hudson. Nonmonotonic causal theories. In *Artificial Intelligence* 2004. Pages 49-104.

Tomi Janhunen and Guohua Liu and Ilkka Niemelä. Tight Integration of Non-Ground Answer Set Programming and Satisfiability Modulo Theories. In *Working notes of the 1st Workshop on Grounding and Transformations for Theories with Variables* 2011.

Lee, Joohyung and Meng, Yunsong. Answer Set Programming Modulo Theories and Reasoning about Continuous Changes. In *Proceedings of International Joint Conference on Artificial Intelligence* 2013. To appear.

Vladimir Lifschitz. Logic Programs with Intensional Functions. In *Proceedings of International Conference on Principles of Knowledge Representation and Reasoning (KR)* 2012.

Guohua Liu and Tomi Janhunen and Ilkka Niemelä. Answer Set Programming via Mixed Integer Programming. In *Proceedings of International Conference on Principles of Knowledge Representation and Reasoning (KR)* 2012.

Miroslaw Truszczyński. Reducts of Propositional Theories, Satisfiability Relations, and Generalizations of Semantics of Logic Programs. In *Proceedings of International Conference on Logic Programming (ICLP)* 2009. Pages 175-189.

[2] `http://peace.eas.asu.edu/joolee/papers/fsm-aspmt-ijcai.pdf`