First-Order Stable Model Semantics with Intensional Functions

Michael Bartholomew and Joohyung Lee

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

Abstract

In classical logic, nonBoolean fluents, such as the location of an object, can be naturally described by functions. However, this is not the case in answer set programs, where the values of functions are pre-defined, and nonmonotonicity of the semantics is related to minimizing the extents of predicates but has nothing to do with functions. We extend the first-order stable model semantics by Ferraris, Lee, and Lifschitz to allow intensional functions – functions that are specified by a logic program just like predicates are specified. We show that many known properties of the stable model semantics are naturally extended to this formalism and compare it with other related approaches to incorporating intensional functions. Furthermore, we use this extension as a basis for defining *Answer Set Programming Modulo Theories (ASPMT)*, analogous to the way that Satisfiability Modulo Theories (SMT) is defined, allowing for SMT-like effective first-order reasoning in the context of ASP. Using SMT solving techniques involving functions, ASPMT can be applied to domains containing real numbers and alleviates the grounding problem. We show that other approaches to integrating ASP and CSP/SMT can be related to special cases of ASPMT in which functions are limited to non-intensional ones.

Key words: Answer Set Programming, Intensional functions, Satisfiability Modulo Theories

1 Introduction

Answer set programming (ASP) is a widely used declarative computing paradigm oriented towards solving knowledge-intensive and combinatorial search problems [Lifschitz, 2008; Brewka *et al.*, 2011]. Its success is mainly due to the expressivity of its modeling language based on the concept of a stable model [Gelfond and Lifschitz, 1988] as well as the efficiency of ASP solvers thanks to intelligent grounding (the process that replaces schematic variables with variable-free terms) and efficient search methods that originated from propositional satisfiability (SAT) solvers.

The grounding and solving approach makes ASP highly effective for Boolean decision problems but becomes problematic when the domain contains a large number of numerical values or a set of real numbers. This is in part related to the limited role of functions in the stable model semantics [Lifschitz, 1988] in comparison with what is allowed in classical logic: either functions are eliminated in the process of grounding, or they are associated with fixed, pre-defined interpretations forming an Herbrand universe. Such a limitation forces us to represent *functional* fluents by *predicates*, but not by *functions*. For example, the following (non-ground) ASP rule represents that the water level does not change by default, where t is a variable for time stamps, l is a variable for integers, *not* stands for default negation, and $\sim$ stands for strong negation:

$$\begin{aligned} \mathit{WaterLevel}(t{+}1, l) \ \leftarrow\ & \mathit{WaterLevel}(t, l),\ \mathit{not}\ \ {\sim}\mathit{WaterLevel}(t{+}1, l), \\ & \mathit{Time}(t), \mathit{Level}(l). \end{aligned} \tag{1}$$

An attempt to replace the predicate $\mathit{WaterLevel}(t, l)$ by equality using a function, e.g. "$\mathit{WaterLevel}(t) = l$," does not work under the standard stable model semantics: "$\mathit{not}\ {\sim}(\mathit{WaterLevel}(t{+}1) = l)$" is not even syntactically valid because strong negation precedes equality, rather than an ordinary ASP atom. Besides, $\mathit{WaterLevel}(t) = l$ is false under any Herbrand interpretation unless l is the term $\mathit{WaterLevel}(t)$ itself, implying that $\mathit{WaterLevel}(t) = \mathit{WaterLevel}(t + 1)$ is always false.

While semantically correct, a computational drawback of using a rule like (1) is that a large set of ground rules needs to be generated when the water level ranges over a large integer domain. Moreover, real numbers are not supported at all because grounding cannot even be applied.

To alleviate the "grounding problem," there have been recent efforts in integrating ASP with constraint solving, where functional fluents can be represented by constraint variables and computed without fully grounding their value variables, e.g., [Mellarkod *et al.*, 2008; Gebser *et al.*, 2009; Balduccini, 2009; Janhunen *et al.*, 2011]. Constraint ASP solvers have demonstrated significantly better performance over traditional ASP solvers on many domains involving a large set of numbers, but they do not provide a fully satisfactory solution to the problem above because the concept of a function is not sufficiently general. For example, one may be tempted to rewrite rule (1) in the language of a constraint ASP solver, such as CLINGCON [1] —a combination of ASP solver CLINGO and constraint solver GECODE, as

$$\mathit{WaterLevel}(t{+}1) {=}^{\$}\, l \ \leftarrow\ \mathit{WaterLevel}(t) {=}^{\$}\, l,\ \mathit{not}\ \neg(\mathit{WaterLevel}(t{+}1) {=}^{\$}\, l) \tag{2}$$

where $=^{\$}$ indicates that the atom containing it is a constraint to be processed by constraint solver GECODE and not to be processed by ASP solver CLINGO. The

[1] `http://potassco.sourceforge.net/`

constraint variable $\mathit{WaterLevel}(t)$ is essentially a function that is mapped to a numeric value. However, this idea does not work either. [2] While it is possible to say that $\mathit{WaterLevel}(t) = 10$ and $\mathit{WaterLevel}(t+1) = \mathit{WaterLevel}(t)$ are true in the language of CLINGCON, negation as failure (*not*) in front of constraints does not work in the same way as it does when it is in front of standard ASP atoms. Indeed, rule (2) has no effect on characterizing the default value of $\mathit{WaterLevel}(t)$ and can be dropped without affecting answer sets. This is because nonmonotonicity of the stable model semantics (as well as almost all extensions, including those of Constraint ASP) is related to the minimality condition on predicates but has nothing to do with functions. Thus, unlike with predicates, they do not allow for directly asserting that functions have default values. Such an asymmetric treatment between functions and predicates in Constraint ASP makes the language of Constraint ASP less general than one might desire.

It is apparent that one of the main obstacles encountered in the above work is due to an insufficient level of generality regarding functions. Recently, the problem has been addressed in another, independent line of research to allow general first-order functions in ASP, although it was not motivated by efficient computation. Lifschitz [2012] called such functions "intensional functions"— functions whose values can be described by logic programs, rather than being pre-defined, thus allowing for defeasible reasoning involving functions in accordance with the stable model semantics. In [Cabalar, 2011], based on the notions of *partial functions* and *partial satisfaction*, functional stable models were defined by imposing minimality on the values of partial functions. The semantics presented in [Balduccini, 2012] is a special case of the semantics from [Cabalar, 2011] as shown in [Bartholomew and Lee, 2013c]. On the other hand, intensional functions defined in [Lifschitz, 2012] do not require the rather complex notions of partial functions and partial satisfaction but instead impose the uniqueness of values on *total functions* similar to the way nonmonotonic causal theories [Giunchiglia *et al.*, 2004] are defined. This led to a simpler semantics, but as we show later in this paper, the semantics is not a proper generalization of the first-order stable model semantics from [Ferraris *et al.*, 2011], and moreover, it exhibits some unintuitive behavior.

We present an alternative approach to incorporating intensional functions into the stable model semantics by a simple modification to the first-order stable model semantics from [Ferraris *et al.*, 2011]. It turns out that unlike the semantics from [Lifschitz, 2012], this formalism, which we call "Functional Stable Model Semantics (FSM)," is a proper generalization of the language from [Ferraris *et al.*, 2011], and avoids the unintuitive cases that the language from [Lifschitz, 2012] encounters.

[2] However, there is rather an indirect way to represent the assertion in the language of CLINGCON using *Ab* predicates:

$$\mathit{WaterLevel}(t+1) =^{\$} l \leftarrow \mathit{WaterLevel}(t) =^{\$} l, \mathit{not}\ \mathit{Ab}(t).$$

Furthermore, unlike the one from [Cabalar, 2011], it does not require the extended notion of partial interpretations that deviates from the notion of classical interpretations. Nevertheless, the semantics from [Cabalar, 2011] can be embedded into FSM by simulating partial interpretations by total interpretations with auxiliary constants [Bartholomew and Lee, 2013c].

Unlike the semantics from [Cabalar, 2011], as FSM properly extends the notion of functions in classical logic, its restriction to background theories provides a straightforward, seamless integration of ASP and Satisfiability Modulo Theories (SMT), which we call "Answer Set Programming Modulo Theories (ASPMT)," analogous to the known relationship between first-order logic and SMT. SMT is a generalization of SAT and, at the same time, a special case of first-order logic in which certain predicate and function symbols in background theories have fixed interpretations. Such background theories include difference logic, linear arithmetic, arrays, and non-linear real-valued functions. Likewise, ASPMT can be viewed as a generalization of the traditional ASP and, at the same time, a special case of FSM in which certain background theories are assumed as in SMT. On the other hand, unlike SMT, ASPMT is not only motivated by computational efficiency, but also by expressive knowledge representation. This is due to the fact that ASPMT is a natural extension of both ASP and SMT. Using SMT solving techniques involving functions, ASPMT can be applied to domains containing real numbers and alleviates the grounding problem. It turns out that constraint ASP can be viewed as a special case of ASPMT in which functions are limited to non-intensional ones.

Monotonic	Nonmonotonic
FOL	FSM
SMT	ASP Modulo Theories
SAT	Traditional ASP

Fig. 1. Analogy between SMT and ASPMT

The paper is organized as follows. Section 2 reviews the stable model semantics from [Ferraris *et al.*, 2011], which Section 3 extends to allow intensional functions. Section 4 shows that many known properties of the stable model semantics are naturally established for this extension. Section 5 shows how to eliminate intensional predicates in favor of intensional functions, and Section 6 shows the opposite elimination under a specific condition. Section 7 compares FSM to other approaches to defining intensional functions. Section 8 extends FSM to be many-sorted, and, based on it, Section 9 defines the concept of ASPMT as a special case of many-sorted FSM, and presents its reduction to SMT under certain conditions. Section 10 compares ASPMT to other approaches to combining ASP with CSP and SMT.

This article is an extended version of the conference papers [Bartholomew and Lee, 2012; Bartholomew and Lee, 2013a]. [3]

2 Review: First-Order Stable Model Semantics with Intensional Predicates

The proposed definition of a stable model in this paper is a direct generalization of the one from [Ferraris *et al.*, 2011], which we review in this section. Stable models are defined as classical models that satisfy a certain "stability" condition, which is expressed by ensuring a minimality condition on predicates.

The syntax of formulas is defined the same as in the standard first-order logic. A signature consists of *function constants* and *predicate constants*. Function constants of arity 0 are called *object constants*, and predicate constants of arity 0 are called *propositional constants*. A *term* of a signature σ is formed from object constants of σ and object variables using function constants of σ. An *atom* of σ is an n-ary predicate constant followed by a list of n terms; *atomic formulas* of σ are atoms of σ, equalities between terms of σ, and the 0-place connective $\bot$ (falsity). First-order formulas of σ are built from atomic formulas of σ using the primitive propositional connectives $\bot,\ \wedge,\ \vee,\ \rightarrow$, as well as quantifiers $\forall,\ \exists$. We understand $\neg F$ as an abbreviation of $F \rightarrow \bot$; symbol $\top$ stands for $\bot \rightarrow \bot$, and $F \leftrightarrow G$ stands for $(F \rightarrow G) \wedge (G \rightarrow F)$, and $t_1 \neq t_2$ stands for $\neg(t_1 = t_2)$.

In [Ferraris *et al.*, 2011], stable models are defined in terms of the SM operator, whose definition is similar to the CIRC operator used for defining circumscription [McCarthy, 1980; Lifschitz, 1994]. As in circumscription, for predicate symbols (constants or variables) u and p, expression $u \leq p$ is defined as shorthand for $\forall \mathbf{x}(u(\mathbf{x}) \rightarrow p(\mathbf{x}))$; expression $u = p$ is defined as $\forall \mathbf{x}(u(\mathbf{x}) \leftrightarrow p(\mathbf{x}))$. For lists of predicate symbols $\mathbf{u} = (u_1, \dots, u_n)$ and $\mathbf{p} = (p_1, \dots, p_n)$, expression $\mathbf{u} \leq \mathbf{p}$ is defined as $(u_1 \leq p_1) \wedge \dots \wedge (u_n \leq p_n)$, expression $\mathbf{u} = \mathbf{p}$ is defined as $(u_1 = p_1) \wedge \dots \wedge (u_n = p_n)$, and expression $\mathbf{u} < \mathbf{p}$ is defined as $\mathbf{u} \leq \mathbf{p} \wedge \neg(\mathbf{u} = \mathbf{p})$.

For any first-order formula F and any finite list of predicate constants $\mathbf{p} = (p_1, \dots, p_n)$, formula $\mathrm{SM}[F; \mathbf{p}]$ is defined as

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

where $\widehat{\mathbf{p}}$ is a list of distinct predicate variables $\widehat{p}_1, \dots, \widehat{p}_n$, and $F^*(\widehat{\mathbf{p}})$ is defined recursively as follows:

[3] Besides the complete proofs, this article contains some new results, such as the non-existence of translation from non-$\mathbf{c}$-plain formulas to $\mathbf{c}$-plain formulas, the usefulness of non-$\mathbf{c}$-plain formulas, reducibility of many-sorted FSM to unsorted FSM, and more complete formal comparison with related works.

- When F is an atomic formula, $F^*(\widehat{\mathbf{p}})$ is a formula obtained from F by replacing all predicate constants $\mathbf{p}$ in it with the corresponding predicate variables from $\widehat{\mathbf{p}}$;
- $(G \wedge H)^*(\widehat{\mathbf{p}}) = G^*(\widehat{\mathbf{p}}) \wedge H^*(\widehat{\mathbf{p}})$;
- $(G \vee H)^*(\widehat{\mathbf{p}}) = G^*(\widehat{\mathbf{p}}) \vee H^*(\widehat{\mathbf{p}})$;
- $(G \rightarrow H)^*(\widehat{\mathbf{p}}) = (G^*(\widehat{\mathbf{p}}) \rightarrow H^*(\widehat{\mathbf{p}})) \wedge (G \rightarrow H)$;
- $(\forall x G)^*(\widehat{\mathbf{p}}) = \forall x G^*(\widehat{\mathbf{p}})$;
- $(\exists x G)^*(\widehat{\mathbf{p}}) = \exists x G^*(\widehat{\mathbf{p}})$.

The predicate constants in $\mathbf{p}$ are called *intensional*: these are the predicates that we "intend to characterize" by F.[4] When F is a sentence (i.e., formula without free variables), the models of the second-order sentence SM$[F; \mathbf{p}]$ are called the *stable* models of F relative to $\mathbf{p}$: they are the models of F that are "stable" on $\mathbf{p}$.

Answer sets are defined as a special class of first-order stable models as follows. By $\sigma(F)$ we denote the signature consisting of the function and predicate constants occurring in F. If F contains at least one object constant, an Herbrand interpretation of $\sigma(F)$ that satisfies SM$[F; \mathbf{p}]$ is called an *answer set* of F, where $\mathbf{p}$ is the list of all predicate constants in $\sigma(F)$. The answer sets of a logic program Π are defined as the answer sets of the FOL-representation of Π, which is obtained from Π by

- replacing every comma by conjunction and every *not* by $\neg$ [5]
- turning every rule *Head* $\leftarrow$ *Body* into a formula rewriting it as the implication *Body* $\rightarrow$ *Head*, and
- forming the conjunction of the universal closures of these formulas.

For example, the FOL-representation of the program

$$
\begin{array}{l}
p(a)\\
q(b)\\
r(x) \leftarrow p(x), \textit{not}\ q(x)
\end{array}
$$

is

$$p(a) \wedge q(b) \wedge \forall x((p(x) \wedge \neg q(x)) \rightarrow r(x)) \tag{3}$$

[4] Intensional predicates are analogous to output predicates in Datalog, and non-intensional predicates are analogous to input predicates in Datalog [Lifschitz, 2011].
[5] Strong negation can be incorporated by introducing "negative" predicates as in [Ferraris *et al.*, 2011, Section 8], or can be represented by a Boolean function with the value FALSE [Bartholomew and Lee, 2013b]. For example, $\sim p$ can be represented by $p = \text{FALSE}$.

and $\mathrm{SM}[F;\ p, q, r]$ is

$$
\begin{aligned}
& p(a) \land q(b) \land \forall x((p(x) \land \neg q(x)) \rightarrow r(x)) \\
& \quad \land \neg \exists uvw \Big(\big((u, v, w) < (p, q, r) \big) \land u(a) \land v(b) \\
& \qquad\qquad\qquad \land \forall x \Big(\big((u(x) \land (\neg v(x) \land \neg q(x))) \rightarrow w(x) \big) \land \big((p(x) \land \neg q(x)) \rightarrow r(x) \big) \Big) \Big),
\end{aligned}
$$

which is equivalent to the first-order sentence

$$
\forall x(p(x) \leftrightarrow x = a) \land \forall x(q(x) \leftrightarrow x = b) \land \forall x(r(x) \leftrightarrow (p(x) \land \neg q(x))) \qquad (4)
$$

[Ferraris *et al.*, 2007, Example 3]. The stable models of F are any first-order models of (4). The only answer set of F is the Herbrand model $\{p(a),\ q(b),\ r(a)\}$.

Remark 1 *According to [Ferraris* et al.*, 2011], this definition of an answer set, when applied to the syntax of logic programs, is equivalent to the traditional definition of an answer set that is based on grounding and fixpoints as in [Gelfond and Lifschitz, 1988].*

It is also noted in [Ferraris et al.*, 2011] that if we replace $F^*(\widehat{\mathbf{p}})$ with a simpler expression $F(\widehat{\mathbf{p}})$ (which substitutes $\widehat{\mathbf{p}}$ for $\mathbf{p}$), then the definition of* $\mathrm{SM}[F; \mathbf{p}]$ *reduces to the definition of* $\mathrm{CIRC}[F; \mathbf{p}]$.

The definition of a stable model above is not limited to Herbrand models, so it allows general functions as in classical first-order logic. Indeed, in Section 10, we show that the previous approaches to combining answer set programs and constraint processing can be viewed as special cases of first-order formulas under the stable model semantics. However, these functions are "extensional," and cannot cover examples like (2).

3 Extending First-Order Stable Model Semantics to Allow Intensional Functions

In this section, we generalize the first-order stable model semantics to allow intensional functions in addition to intensional predicates.

3.1 Second-Order Logic Characterization of the Stable Model Semantics

We extend expression $u = c$ as $\forall \mathbf{x}(u(\mathbf{x}) = c(\mathbf{x}))$ if u and c are function symbols. For lists of predicate and function symbols $\mathbf{u} = (u_1, \ldots, u_n)$ and $\mathbf{c} = (c_1, \ldots, c_n)$, expression $\mathbf{u} = \mathbf{c}$ is defined as $(u_1 = c_1) \land \cdots \land (u_n = c_n)$.

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 $\mathbf{c}$. By $\mathbf{c}^{pred}$ ($\mathbf{c}^{func}$, respectively) we mean the list of all predicate constants (function constants, respectively) in $\mathbf{c}$, and by $\widehat{\mathbf{c}}^{pred}$ ($\widehat{\mathbf{c}}^{func}$, respectively) the list of the corresponding predicate variables (function variables, respectively) in $\widehat{\mathbf{c}}$. For any formula F, expression $\mathrm{SM}[F;\ \mathbf{c}]$ is defined as

$$F \wedge \neg\exists\widehat{\mathbf{c}}(\widehat{\mathbf{c}} < \mathbf{c} \wedge F^*(\widehat{\mathbf{c}})), \tag{5}$$

where $\widehat{\mathbf{c}} < \mathbf{c}$ is shorthand for $(\widehat{\mathbf{c}}^{pred} \leq \mathbf{c}^{pred}) \wedge \neg(\widehat{\mathbf{c}} = \mathbf{c})$, and $F^*(\widehat{\mathbf{c}})$ is defined recursively in the same way as $F^*(\widehat{\mathbf{p}})$ except for the base case, which is defined as follows.

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

As before, we say that an interpretation I that satisfies $\mathrm{SM}[F;\mathbf{c}]$ a *stable model* of F relative to $\mathbf{c}$. Clearly, every stable model of F is a model of F but not vice versa.

Remark 2 *It is easy to see that the definition of a stable model above is a proper generalization of the one from [Ferraris* et al.*, 2011], also reviewed in the previous section: the definition of* $\mathrm{SM}[F;\mathbf{c}]$ *in this section reduces to the one in the previous section when all intensional constants in* $\mathbf{c}$ *are predicate constants only.*

When all intensional constants are function constants only, the definition of $\mathrm{SM}[F;\mathbf{c}]$ *is similar to the first-order nonmonotonic causal theories defined in [Lifschitz, 1997]. The only difference is that, instead of* $F^*(\widehat{c})$*, a different expression is used there. A more detailed comparison is given in Section 7.1.*

We will often write $F \rightarrow G$ as $G \leftarrow F$ and identify a finite set of formulas with the conjunction of the universal closures of each formula in that set.

For any formula F, expression $\{F\}^{\mathrm{ch}}$ denotes the "choice" formula $(F \vee \neg F)$.

The following two lemmas are often useful in simplifying $F^*(\widehat{\mathbf{c}})$, as we demonstrate in Example 1 below. They are natural extensions of Lemmas 5 and 6 from [Ferraris *et al.*, 2011].

Lemma 1 *Formula*

$$(\widehat{\mathbf{c}} < \mathbf{c}) \wedge F^*(\widehat{\mathbf{c}}) \rightarrow F$$

is logically valid.

Proof. By induction on the structure of F. ∎

Lemma 2 *Formula*

$$\widehat{\mathbf{c}} < \mathbf{c} \rightarrow ((\neg F)^*(\widehat{\mathbf{c}}) \leftrightarrow \neg F)$$

is logically valid.

Proof. Immediate from Lemma 1. ∎

Example 1 *The following program F_1 describes the level of an unlimited water tank that is filled up unless it is flushed.*

$$\begin{aligned} \{Amt_1 = x+1\}^{\text{ch}} &\leftarrow Amt_0 = x, \\ Amt_1 = 0 &\leftarrow Flush\ . \end{aligned} \tag{6}$$

Here Amt_1 is an intensional function constant, and x is a variable ranging over nonnegative integers. Intuitively, the first rule asserts that the amount increases by one by default. [6] *However, if Flush action is executed (e.g., if we add the fact Flush to* (6)*), this behavior is overridden, and the amount is set to* 0*.*

Using Lemmas 1 and 2, under the assumption $\widehat{Amt_1} < Amt_1$, one can check that formula $F_1^*(\widehat{Amt_1})$ is equivalent to the conjunction consisting of (6) *and*

$$\begin{aligned} (\widehat{Amt_1} = x+1 \land Amt_1 = x+1) \lor \neg(Amt_1 = x+1) &\ \leftarrow\ Amt_0 = x, \\ \widehat{Amt_1} = 0 \land Amt_1 = 0 &\ \leftarrow\ Flush, \end{aligned} \tag{7}$$

so that

$$\begin{aligned} \text{SM}[F_1; Amt_1] &= F_1 \land \neg\exists\widehat{Amt_1}(\widehat{Amt_1} \neq Amt_1 \land F_1^*(\widehat{Amt_1})) \\ &\Leftrightarrow F_1 \land \neg\exists\widehat{Amt_1}(\widehat{Amt_1} \neq Amt_1 \land \\ &\qquad\qquad \forall x(Amt_0 = x \rightarrow \neg(Amt_1 = x+1)) \land (Flush \rightarrow \bot)). \end{aligned}$$

Consider the first-order interpretations that have the set of nonnegative integers as the universe, interprets integers, arithmetic functions, and comparison operators in the standard way, and maps the other constants in the following way.

	Amt_0	$Flush$	Amt_1
I_1	5	FALSE	6
I_2	5	FALSE	8
I_3	5	TRUE	0

- *Interpretation I_1 is in accordance with the intuitive reading of the rules above, and it is indeed a model of* $\text{SM}[F_1; Amt_1]$*.*

[6] Section 4.2 explains why choice formulas are read as specifying default values.

- *Interpretation I_2 is not intuitive (the amount suddenly jumps up with no reason). It is not a model of* $\mathrm{SM}[F_1; Amt_1]$ *though it is a model of* F_1.
- *Interpretation I_3 is in accordance with the intuitive reading of the rules above. It is a model of* $\mathrm{SM}[F_1; Amt_1]$.

3.2 Reduct-Based Characterization of the Stable Model Semantics

The second-order logic based definition of a stable model in the previous section is succinct, and is a natural extension of the first-order stable model semantics that is defined in [Ferraris *et al.*, 2011], but it may look distant from the usual definition of a stable model in the literature that is given in terms of grounding and fixpoints.

In [Bartholomew and Lee, 2013c], an equivalent definition of the functional stable model semantics in terms of *infinitary ground formulas* and *reduct* is given. Appendix A of this article contains a review of the definition.

4 Properties of Functional Stable Models

Many properties known for the stable model semantics can be naturally extended to the functional stable model semantics, which is a desirable feature of the proposed formalism.

4.1 Constraints

Following Ferraris *et al.* [2009], we say that an occurrence of a constant or any other subexpression in a formula F is *positive* if the number of implications containing that occurrence in the antecedent is even, and *negative* otherwise. We say that the occurrence is *strictly positive* if the number of implications in F containing that occurrence in the antecedent is 0. For example, in $\neg(f = 1) \rightarrow g = 1$, the occurrences of f and g are both positive, but only the occurrence of g is strictly positive. [7]

About a formula F we say that it is *negative* on a list $\mathbf{c}$ of predicate and function constants if F has no strictly positive occurrence of a constant from $\mathbf{c}$. Since any formula of the form $\neg H$ is shorthand for $H \rightarrow \bot$, such a formula is negative on any list of constants. The formulas of the form $\neg H$ are called *constraints* in the literature of ASP: adding a constraint to a program affects the set of its stable

[7] Recall that we understand $\neg F$ as shorthand for $F \rightarrow \bot$.

models in a particularly simple way by eliminating the stable models that "violate" the constraint. [8]

The following theorem is a generalization of Theorem 3 from [Ferraris *et al.*, 2011] for the functional stable model semantics.

Theorem 1 *For any first-order formulas F and G, if G is negative on $\mathbf{c}$, then* $\text{SM}[F \wedge G; \mathbf{c}]$ *is equivalent to* $\text{SM}[F; \mathbf{c}] \wedge G$.

Example 2 *Consider* $\text{SM}[F_2 \wedge \neg(f=1); fg]$ *where F_2 is* $(f=1 \vee g=1) \wedge (f=2 \vee g=2)$. *Since $\neg(f=1)$ is negative on $\{f, g\}$, according to Theorem 1,* $\text{SM}[F_2 \wedge \neg(f=1); fg]$ *is equivalent to* $\text{SM}[F_2; fg] \wedge \neg(f=1)$*, which is equivalent to* $f=2 \wedge g=1$.

4.2 Choice and Defaults

Similar to Theorem 2 from [Ferraris *et al.*, 2011], Theorem 2 below shows that making the set of intensional constants smaller can only make the result of applying SM weaker, and that this can be compensated by adding choice formulas. For any predicate constant p, by *Choice*(p) we denote the formula $\forall \mathbf{x}\{p(\mathbf{x})\}^{\text{ch}}$ (recall that $\{F\}^{\text{ch}}$ is shorthand for $F \vee \neg F$), where $\mathbf{x}$ is a list of distinct object variables. For any function constant f, by *Choice*(f) we denote the formula $\forall \mathbf{x}y\{f(\mathbf{x}) = y\}^{\text{ch}}$, where y is an object variable that is distinct from $\mathbf{x}$. For any finite list of predicate and function constants $\mathbf{c}$, the expression *Choice*$(\mathbf{c})$ stands for the conjunction of the formulas *Choice*(c) for all members c of $\mathbf{c}$. We sometimes identify a list with the corresponding set when there is no confusion.

The following theorem is a generalization of Theorem 7 from [Ferraris *et al.*, 2011] for the functional stable model semantics.

Theorem 2 *For any first-order formula F and any disjoint lists $\mathbf{c}$, $\mathbf{d}$ of distinct constants, the following formulas are logically valid:*

$$\text{SM}[F; \mathbf{cd}] \rightarrow \text{SM}[F; \mathbf{c}],$$

$$\text{SM}[F \wedge \mathit{Choice}(\mathbf{d}); \mathbf{cd}] \leftrightarrow \text{SM}[F; \mathbf{c}].$$

For example,

$$\text{SM}[(g=1 \rightarrow f=1) \wedge \forall y(g=y \vee \neg(g=y));\ fg]$$

is equivalent to

$$\text{SM}[g=1 \rightarrow f=1;\ f].$$

[8] Note that the term "constraint" here is different from the one used in CSP.

A formula $\{f(\mathbf{t}) = \mathbf{t}'\}^{\rm ch}$, where f is an intensional function constant and $\mathbf{t}$, $\mathbf{t}'$ contain no intensional function constants, intuitively represents that $f(\mathbf{t})$ *takes the value* $\mathbf{t}'$ *by default*. For example, the stable models of $\{g = 1\}^{\rm ch}$ relative to g map g to 1. On the other hand, the default behavior is overridden when we conjoin the formula with $g = 2$: the stable models of

$$\{g = 1\}^{\rm ch} \wedge g = 2$$

relative to g map g to 2, and no longer to 1.

The treatment of $\{g = 1\}^{\rm ch}$ as $(g = 1) \vee \neg(g = 1)$ is similar to the choice rule $\{p\}^{\rm ch}$ in ASP for propositional constant p, which stands for $p \vee \neg p$, with an exception that g has to satisfy a functional requirement, i.e., it is mapped to a unique value. Under that requirement, an interpretation that maps g to 1 is a stable model but another assignment to g is not a stable model because the choice rule itself does not force one to believe that g is mapped to that other value. This makes the choice rule for the function work as assigning a default value to the function.

With this understanding, the commonsense law of inertia can be succinctly represented using choice formulas for functions. For instance, the formula

$$Loc(b,t) = l \;\rightarrow\; \{Loc(b,t+1) = l\}^{\rm ch}, \tag{8}$$

where *Loc* is an intensional function constant, represents that the location of a block b at next step retains its value by default. The default behavior can be overridden if some action moves the block. In contrast, the standard ASP representation of the commonsense law of inertia, such as (1), uses both default negation and strong negation, and requires the user to be aware of the subtle difference between them.

4.3 Strong Equivalence

Strong equivalence [Lifschitz *et al.*, 2001] is an important notion that allows us to replace a subformula with another subformula without affecting the stable models. The theorem on strong equivalence can be extended to formulas with intensional functions as follows.

For first-order formulas F and G, we say that F is *strongly equivalent* to G if, for any formula H, any occurrence of F in H, and any list $\mathbf{c}$ of distinct predicate and function constants, $\text{SM}[H; \mathbf{c}]$ is equivalent to $\text{SM}[H'; \mathbf{c}]$, where H' is obtained from H by replacing the occurrence of F by G.

The following theorem tells us that strong equivalence can be characterized in terms of equivalence in classical logic.

Theorem 3 *Let F and G be first-order formulas, let $\mathbf{c}$ be the list of all predicate* *and function constants occurring in F or G, and let $\widehat{\mathbf{c}}$ be a list of distinct predicate and function variables corresponding to $\mathbf{c}$. The following conditions are equivalent to each other.*

- *F and G are strongly equivalent to each other;*
- *Formula*
$$(F \leftrightarrow G) \wedge (\widehat{\mathbf{c}} < \mathbf{c} \rightarrow (F^*(\widehat{\mathbf{c}}) \leftrightarrow G^*(\widehat{\mathbf{c}}))) \tag{9}$$
is logically valid.

For instance, choice formula $\{F\}^{\mathrm{ch}}$ is strongly equivalent to $\neg\neg F \rightarrow F$. This can be shown, in accordance with Theorem 3, by checking that not only they are classically equivalent but also
$$(F \vee \neg F)^*(\widehat{\mathbf{c}})$$
and
$$(\neg\neg F \rightarrow F)^*(\widehat{\mathbf{c}})$$
are classically equivalent under $\widehat{\mathbf{c}} < \mathbf{c}$. Indeed, in view of Lemma 2, $(F \vee \neg F)^*(\widehat{\mathbf{c}})$ is equivalent to $(F^*(\widehat{\mathbf{c}}) \vee \neg F)$ and $(\neg\neg F \rightarrow F)^*(\widehat{\mathbf{c}})$ is equivalent to $F \rightarrow F^*(\widehat{\mathbf{c}})$. This fact allows us to rewrite formula (8) as an implication in which the consequent is an atomic formula:

$$\mathit{Loc}(b,t)\!=\!l \wedge \neg\neg(\mathit{Loc}(b,t+1)\!=\!l) \ \rightarrow \ \mathit{Loc}(b,t\!+\!1)\!=\!l.$$

For another example, $(G \rightarrow F) \wedge (H \rightarrow F)$ is strongly equivalent to $(G \vee H) \rightarrow F$. This is useful for rewriting a theory into "Clark normal form," to which we can apply completion as presented in the next section.

4.4 Completion

Completion [Clark, 1978] is a process that turns formulas under the stable model semantics to formulas under the standard first-order logic.

We say that a formula F is in *Clark normal form* (relative to a list $\mathbf{c}$ of intensional constants) if it is a conjunction of sentences of the form

$$\forall \mathbf{x}(G \rightarrow p(\mathbf{x})) \tag{10}$$

and

$$\forall \mathbf{x}y(G \rightarrow f(\mathbf{x})\!=\!y) \tag{11}$$

one for each intensional predicate constant p in $\mathbf{c}$ and each intensional function constant f in $\mathbf{c}$, where $\mathbf{x}$ is a list of distinct object variables, y is another object variable, and G is a formula that has no free variables other than those in $\mathbf{x}$ and y.

The *completion* of a formula F in Clark normal form relative to $\mathbf{c}$, denoted by COMP$[F;\mathbf{c}]$, is obtained from F by replacing each conjunctive term (10) with

$$\forall\mathbf{x}(p(\mathbf{x}) \leftrightarrow G) \tag{12}$$

and each conjunctive term (11) with

$$\forall\mathbf{x}y(f(\mathbf{x})\!=\!y \leftrightarrow G). \tag{13}$$

The *dependency graph* of F (relative to $\mathbf{c}$), denoted by $\mathrm{DG}_{\mathbf{c}}[F]$, is the directed graph that

- has all members of $\mathbf{c}$ as its vertices, and
- has an edge from c to d if, for some strictly positive occurrence of $G \to H$ in F,
  - c has a strictly positive occurrence in H, and
  - d has a strictly positive occurrence in G.

We say that F is *tight* (on $\mathbf{c}$) if the dependency graph of F (relative to $\mathbf{c}$) is acyclic. The following theorem, which generalizes Theorem 11 from [Ferraris *et al.*, 2011] for the functional stable model semantics, tells us that, for a tight formula, completion is a process that allows us to reclassify intensional constants as non-intensional ones. It is similar to the main theorem of [Lifschitz and Yang, 2013], which describes functional completion in the context of nonmonotonic causal logic.

Theorem 4 *For any formula F in Clark normal form relative to* $\mathbf{c}$ *that is tight on* $\mathbf{c}$*, an interpretation I that satisfies $\exists xy(x \neq y)$ is a model of* SM$[F;\mathbf{c}]$ *iff I is a model of* COMP$[F;\mathbf{c}]$.

Example 1 Continued *Formula F_1 is not in Clark normal Form relative to Amt_1, but it is strongly equivalent to*

$$\begin{array}{l}
Amt_1\!=\!y \;\leftarrow\; y\!=\!x\!+\!1 \wedge Amt_0\!=\!x \wedge \neg\neg(Amt_1 = y),\\
Amt_1\!=\!y \;\leftarrow\; y\!=\!0 \wedge Flush\;.
\end{array}$$

and further to

$$Amt_1\!=\!y \;\leftarrow\; \Big(y\!=\!x\!+\!1 \wedge Amt_0\!=\!x \wedge \neg\neg(Amt_1\!=\!y)\Big) \vee \Big(y\!=\!0 \wedge Flush\Big),$$

which is in Clark normal form relative to Amt_1 and is tight on Amt_1. In accordance with Theorem 4, the stable models of F_1 relative to Amt_1 coincide with the classical models of

$$Amt_1\!=\!y \;\leftrightarrow\; \Big(y\!=\!x\!+\!1 \wedge Amt_0\!=\!x \wedge \neg\neg(Amt_1\!=\!y)\Big) \vee \Big(y\!=\!0 \wedge Flush\Big).$$

The assumption $\exists xy(x \neq y)$ in the statement of Theorem 4 is essential to avoid the mismatch between "trivial" stable models and models of completion when the universe is a singleton. Recall that in order to dispute the stability of a model I in the presence of intensional function constants, one needs another interpretation that is different from I on intensional function constants. If the universe contains only one element, the stability of a model is trivial. For example, take F to be $\top$ and $\mathbf{c}$ to be an intensional function constant f. If the universe $|I|$ of an interpretation I is a singleton, then I satisfies $\mathrm{SM}[F]$ because there is only one way to interpret $\mathbf{c}$, but I does not satisfy the completion formula $\forall \mathbf{x} y(f(\mathbf{x}) = y \leftrightarrow \bot)$.

5 Eliminating Intensional Predicates in Favor of Intensional Functions

In first-order logic, it is known that predicate constants can be replaced by function constants and vice versa. This section and the next section show similar transformations under the functional stable model semantics.

5.1 Eliminating Intensional Predicates

Intensional predicate constants can be eliminated in favor of intensional function constants as follows.

Given a formula F and an intensional predicate constant p, formula F^p_f is obtained from F as follows:

- in the signature of F, replace p with a new intensional function constant f of arity n, where n is the arity of p, and add two new non-intensional object constants 0 and 1 (rename if necessary);
- replace each subformula $p(\mathbf{t})$ in F with $f(\mathbf{t}) = 1$.

By FC_f ("Functional Constraint on f") we denote the conjunction of the following formulas, which enforces f to be two-valued:

$$0 \neq 1, \tag{14}$$

$$\neg\neg\forall \mathbf{x}(f(\mathbf{x}) = 0 \lor f(\mathbf{x}) = 1), \tag{15}$$

where $\mathbf{x}$ is a list of distinct object variables. By DF_f ("Default False on f") we denote the formula

$$\forall \mathbf{x}\{f(\mathbf{x}) = 0\}^{\mathrm{ch}}. \tag{16}$$

Example 3 *Let F be the conjunction of the universal closures of the following* *formulas:*

$$Loc(b,t)=l \to \{Loc(b,t+1)=l\}^{\mathrm{ch}},$$
$$Move(b,l,t) \to Loc(b,t+1)=l$$

(lower case symbols are variables). We eliminate the intensional predicate constant Move in favor of an intensional function constant $Move_f$ to obtain $F^{Move}_{Move_f} \wedge FC_{Move_f} \wedge DF_{Move_f}$, which is the conjunction of the universal closures of the following formulas:

$$Loc(b,t)=l \to \{Loc(b,t{+}1)=l\}^{\mathrm{ch}},$$
$$Move_f(b,l,t)=1 \to Loc(b,t{+}1)=l,$$
$$0 \neq 1,$$
$$\neg\neg(Move_f(b,l,t)=0 \vee Move_f(b,l,t)=1),$$
$$\{Move_f(b,l,t)=0\}^{\mathrm{ch}}.$$

The following theorem asserts the correctness of the elimination method.

Theorem 5 *The set of formulas*

$$\{\forall \mathbf{x}(f(\mathbf{x})=1 \leftrightarrow p(\mathbf{x})),\ \ FC_f\}$$

entails

$$\mathrm{SM}[F;p\mathbf{c}] \leftrightarrow \mathrm{SM}[F^p_f \wedge DF_f; f\mathbf{c}].$$

The following corollary to Theorem 5 tells us that there is a 1–1 correspondence between the stable models of F and the stable models of its "functional image" $F^p_f \wedge DF_f \wedge FC_f$. For any interpretation I of the signature of F, by I^p_f we denote the interpretation of the signature of F^p_f obtained from I by replacing the set p^I with the function $f^{I^p_f}$ such that, for all $\xi_1,\dots,\xi_n$ in the universe of I,

$$f^{I^p_f}(\xi_1,\dots,\xi_n)=1^I \text{ if } p^I(\xi_1,\dots,\xi_n)=\text{TRUE}$$
$$f^{I^p_f}(\xi_1,\dots,\xi_n)=0^I \text{ otherwise .}$$

Furthermore, we assume that I^p_f satisfies (14). Consequently, I^p_f satisfies FC_f.

Corollary 6 *Let F be a first-order sentence.*

(a) An interpretation I of the signature of F is a model of $\mathrm{SM}[F;p\mathbf{c}]$ *iff I^p_f is a model of* $\mathrm{SM}[F^p_f \wedge DF_f \wedge FC_f; f\mathbf{c}]$.

(b) An interpretation J of the signature of F^p_f is a model of $\mathrm{SM}[F^p_f \wedge DF_f \wedge FC_f; f\mathbf{c}]$ *iff $J = I^p_f$ for some model I of* $\mathrm{SM}[F;p\mathbf{c}]$.

In Corollary 6 (b), it is clear by the construction of I_f^p that, for each J, there is exactly one I that satisfies the statement.

Repeated applications of Corollary 6 allow us to completely eliminate intensional predicate constants in favor of intensional function constants, thereby turning formulas under the stable model semantics from [Ferraris *et al.*, 2011] into formulas under FSM whose intensional constants are function constants only.

Note that $\neg\neg$ in (15) cannot be dropped in general. The formula $\neg\neg F$ is not strongly equivalent to F. The former is a weaker assertion than the latter under the stable model semantics. Indeed, if it is dropped, in Corollary 6, when F is $\top$, the empty set is the only model of SM$[F; p]$ whereas SM$[F_f^p \wedge DF_f \wedge FC_f; f]$ has two models where f is mapped to 0 or 1.

6 Eliminating Intensional Functions in favor of Intensional Predicates

We show how to eliminate intensional function constants in favor of intensional predicate constants. Unlike in the previous section, the result is established for "f-plain" formulas only. It turns out that there is no elimination method for arbitrary formulas that is both modular and signature-preserving.

6.1 Eliminating Intensional Functions from $\mathbf{c}$*-Plain Formulas in favor of Intensional Predicates*

Let f be a function constant. A first-order formula is called f*-plain* [Lifschitz and Yang, 2011] if each atomic formula in it

- does not contain f, or
- is of the form $f(\mathbf{t}) = t_1$ where $\mathbf{t}$ is a tuple of terms not containing f, and t_1 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 any list $\mathbf{c}$ of predicate and function constants, we say that F is $\mathbf{c}$*-plain* if F is f-plain for each function constant f in $\mathbf{c}$.

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

- in the signature of F, replace f with a new intensional predicate constant p of arity $n + 1$, where n is the arity of f;
- replace each subformula $f(\mathbf{t}) = t_1$ in F with $p(\mathbf{t}, t_1)$.

The following theorem asserts the correctness of the elimination.

Theorem 7 *For any f-plain formula F, the set of formulas*

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

entails

$$\mathrm{SM}[F; f\mathbf{c}] \leftrightarrow \mathrm{SM}[F^f_p; p\mathbf{c}].$$

The theorem tells us how to eliminate an intensional function constant f from an f-plain formula in favor of an intensional predicate constant. By UEC_p we denote the following formulas that enforce the "functional image" on the predicate p,

$$\begin{gathered}\forall \mathbf{x} yz(p(\mathbf{x}, y) \wedge p(\mathbf{x}, z) \wedge y \neq z \rightarrow \bot), \\ \neg\neg\forall \mathbf{x} \exists y\, p(\mathbf{x}, y),\end{gathered} \tag{17}$$

where $\mathbf{x}$ is an n-tuple of variables, and all variables in $\mathbf{x}$, y, and z are pairwise distinct. Note that each formula is negative on any list of constants, so they work as constraints (Section 4.1) to eliminate the stable models that violate them.

Example 4 *Consider the same formula F in Example 3. We eliminate the function constant Loc in favor of the intensional predicate constant Loc_p to obtain $F^{\mathit{Loc}}_{\mathit{Loc}_p} \wedge \mathit{UEC}_{\mathit{Loc}_p}$, which is the conjunction of the universal closures of the following formulas:*

$$\begin{gathered}\mathit{Loc}_p(b, t, l) \rightarrow \{\mathit{Loc}_p(b, t{+}1, l)\}^{\mathrm{ch}}, \\ \mathit{Move}(b, l, t) \rightarrow \mathit{Loc}_p(b, t{+}1, l), \\ \mathit{Loc}_p(b, t, l) \wedge \mathit{Loc}_p(b, t, l') \wedge l \neq l' \rightarrow \bot, \\ \neg\neg\forall b\, t\, \exists l(\mathit{Loc}_p(b, t, l)).\end{gathered} \tag{18}$$

The following corollary shows that there is a simple 1–1 correspondence between the stable models of F and the stable models of $F^f_p \wedge \mathit{UEC}_p$. Recall that the signature of F^f_p is obtained from the signature of F by replacing f with p. For any interpretation I of the signature of F, by I^f_p we denote the interpretation of the signature of F^f_p obtained from I by replacing the function f^I with the predicate p^I that consists of the tuples

$$\langle \xi_1, \dots, \xi_n, f^I(\xi_1, \dots, \xi_n)\rangle$$

for all $\xi_1, \dots, \xi_n$ from the universe of I.

Corollary 8 *Let F be an f-plain sentence.*

(a) 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 is a model of $\mathrm{SM}[F^f_p \wedge \mathit{UEC}_p;\; p\mathbf{c}]$.

(b) An interpretation J of the signature of F_p^f that satisfies $\exists xy(x \neq y)$ is a model of $\mathrm{SM}[F_p^f \wedge \mathit{UEC}_p;\ p\mathbf{c}]$ *iff* $J = I_p^f$ *for some model I of* $\mathrm{SM}[F; f\mathbf{c}]$.

In Corollary 8 (b), it is clear by the construction of I_p^f that, for each J, there is exactly one I that satisfies the statement.

Theorem 7 and Corollary 8 are similar to Theorem 3 and Corollary 5 from [Lifschitz and Yang, 2011], which are about eliminating "explainable" functions in nonmonotonic causal logic in favor of "explainable" predicates.

Similar to Theorem 4, the condition $\exists xy(x \neq y)$ is necessary in Theorem 7 and Corollary 8 because in order to dispute the stability of a model I in the presence of intensional function constants, one needs another interpretation that is different from I on intensional function constants. Such an interpretation simply does not exist if the condition is missing, so I becomes trivially stable. For example, consider the formula $\top$ with signature $\sigma = \{f\}$ and the universe $\{1\}$. There is only one interpretation, which maps f to 1. This is a stable model of $\top$. On the other hand, the formula $\top \wedge \mathit{UEC}_p$, which is $\top \wedge \neg\neg\exists y\, p(y)$, has no stable models.

The method above eliminates only one intensional function constant at a time, but repeated applications can eliminate all intensional function constants from a given $\mathbf{c}$-plain formula in favor of intensional predicate constants. In other words, it tells us that the stable model semantics for $\mathbf{c}$-plain formulas can be reduced to the stable model semantics from [Ferraris *et al.*, 2011] by adding uniqueness and existence of value constraints.

The elimination method described in Corollary 8 has shown to be useful in a special class of FSM, known as *multi-valued propositional formulas* [Giunchiglia *et al.*, 2004]. [9] In [Lee *et al.*, 2013], the method allows us to relate the two different translations of action language $\mathcal{BC}$ into multi-valued propositional formulas and into the usual ASP programs. Also, it led to the design of MVSM, [10] which computes stable models of multi-valued propositional formulas using F2LP and CLINGO, and the design of CPLUS2ASP [Babb and Lee, 2013], [11] which computes action languages using ASP solvers.

Interestingly, the elimination method results in a new way of formalizing the commonsense law of inertia using choice rules instead of using strong negation, e.g., (1). The formulas (18) can be more succinctly represented in the language of ASP

[9] We discuss the relationship in Section 8.2.

[10] `http://reasoning.eas.asu.edu/mvsm/`

[11] `http://reasoning.eas.asu.edu/cplus2asp/`

as follows.

$$\{Loc_p(b, t{+}1, l)\}^{\rm ch} \leftarrow Loc_p(b, t, l)$$
$$Loc_p(b, t{+}1, l) \leftarrow Move(b, l, t)$$
$$\leftarrow not\ 1\{Loc_p(b, t, l) : Location(l)\}1, Block(b), Time(t)$$

where *Location*, *Block*, and *Time* are domain predicates. The first rule says that if the location of b at time t is l, then decide arbitrarily whether to assert $Loc_p(b, t{+}1, l)$ at time $t+1$. In the absence of additional information about the location of b at time $t+1$, asserting $Loc_p(b, t{+}1, l)$ will be the only option, as the third rule requires one of the location l to be associated with the block b at time $t+1$. But if we are given conflicting information about the location at time $t+1$ due to the *Move* action, then not asserting $Loc_p(b, t{+}1, l)$ will be the only option, and the second rule will tell us the new location of b at time $t+1$.

6.2 Non-$\mathbf{c}$*-plain formulas vs.* $\mathbf{c}$*-plain formulas*

One may wonder if the method of eliminating intensional function constants in the previous section can be extended to non-$\mathbf{c}$-plain formulas, possibly by first rewriting the formulas into $\mathbf{c}$-plain formulas. In classical logic, this is easily done by "unfolding" nested functions by introducing existential quantifiers, but this is not the case under the stable model semantics because nested functions in general express weaker assertions than unfolded ones.

Example 5 *Consider F to be $a + b = 5$, where a and b are object constants. The formula F is equivalent to $\exists xy(a{=}x \land b{=}y \land x + y{=}5)$ under classical logic, but this is not the case under FSM. The former has no stable models, and the latter has many stable models, including I such that $a^I = 1, b^I = 4$.*

Gelfond and Kahl [2014] describe the intuitive meaning of stable models in terms of *rationality principle*: "believe nothing you are not forced to believe." In the example above, it is natural to understand that $a + b = 5$ does not force one to believe $a = 1$ and $b = 4$.

The weaker assertion expressed by function nesting is useful for specifying the range of a function using a domain predicate, or expressing the concept of synonymity between the two functions without forcing the functions to have specific values.

Example 6 *Consider F to be $Dom(a)$ where Dom is a predicate constant and a is an object constant. The formula F can be viewed as applying the sort predicate (i.e., domain predicate) Dom to specify the value range of a, but it does not force one to believe that a has a particular value. In classical logic, F is equivalent* *to $\exists x(Dom(x) \wedge x = a)$, but their stable models are different. The former has no stable models, and the latter has many stable models, including I such that $Dom^I = \{1, 2, 3\}$ and $a^I = 1$.*

Example 7 *A "synonymity" rule [Lifschitz and Yang, 2011] has the form*

$$B \;\rightarrow\; f_1(\mathbf{t}_1) = f_2(\mathbf{t}_2), \tag{19}$$

where f_1, f_2 are intensional function constants in $\mathbf{f}$, and $\mathbf{t}_1$, $\mathbf{t}_2$ are tuples of terms not containing members of $\mathbf{f}$. This rule expresses that we believe $f_1(\mathbf{t}_1)$ to be "synonymous" to $f_2(\mathbf{t}_2)$ under condition B, but it does not force one to assign particular values to $f_1(\mathbf{t_1})$ and $f_2(\mathbf{t_2})$. As a special case, consider $f_1 = f_2$ vs. $\exists x(f_1 = x \wedge f_2 = x)$. The latter forces one to assign some values to f_1 and f_2, and does not express the intended weaker assertion that they are synonymous.

To sum up, in Examples 5, 6, and 7, the classically equivalent transformations do not preserve strong equivalence. They affect the beliefs, forcing one to believe more than what the original formulas assert.

On the other hand, there is some special class of formulas for which the process of "unfolding" preserves stable models. We first define precisely the process.

Definition 1 *The process of* unfolding *F w.r.t. a list $\mathbf{c}$ of constants, denoted by $UF_{\mathbf{c}}(F)$, is recursively defined as follows.*

- *If F is an atomic formula that is $\mathbf{c}$-plain, $UF_{\mathbf{c}}(F)$ is F;*
- *If F is an atomic formula of the form $p(t_1, \dots, t_n)$ ($n \geq 0$) such that $t_{k_1}, \dots, t_{k_j}$ are all the terms in $t_1, \dots, t_n$ that contain some members of $\mathbf{c}$, then $UF_{\mathbf{c}}(p(t_1, \dots, t_n))$ is*
$$\exists x_1 \dots x_j \Big(p(t_1, \dots, t_n)'' \wedge \bigwedge_{1 \leq i \leq j} UF_{\mathbf{c}}(t_{k_i} = x_i)\Big),$$
where $p(t_1, \dots, t_n)''$ is obtained from $p(t_1, \dots, t_n)$ by replacing each t_{k_i} with a new variable x_i.
- *If F is an atomic formula of the form $f(t_1, \dots, t_n) = t_0$ ($n \geq 0$) such that $t_{k_1}, \dots, t_{k_j}$ are all the terms in $t_0, \dots, t_n$ that contain some members of $\mathbf{c}$, then $UF_{\mathbf{c}}(f(t_1, \dots, t_n) = t_0)$ is*
$$\exists x_1 \dots x_j \Big((f(t_1, \dots, t_n) = t_0)'' \wedge \bigwedge_{1 \leq i \leq j} UF_{\mathbf{c}}(t_{k_i} = x_i)\Big),$$
where $(f(t_1, \dots, t_n) = t_0)''$ is obtained from $f(t_1, \dots, t_n) = t_0$ by replacing each t_{k_i} with a new variable x_i.
- *$UF_{\mathbf{c}}(F \odot G)$ is $UF_{\mathbf{c}}(F) \odot UF_{\mathbf{c}}(G)$, where $\odot \in \{\wedge, \vee, \rightarrow\}$.*
- *$UF_{\mathbf{c}}(QxF)$ is $Qx\ UF_{\mathbf{c}}(F(x))$, where $Q \in \{\forall, \exists\}$.*

In Example 6, $UF_{Dom}(F)$ is $\exists x(Dom(x) \wedge a = x)$, and in Example 5, $UF_{(a,b)}(F)$ is $\exists xy(a = x \wedge b = y \wedge x + y = 5)$. In Example 7, $UF_{(f_1,f_2)}(f_1 = f_2)$ is $\exists x(f_1 =$

$x \wedge f_2 = x$). We already observed that the process of unfolding does not preserve the stable models of the formulas.

Theorem 9 below presents a special class of formulas, for which the process of unfolding does preserve stable models, or in other words, unfolding does not affect the beliefs.

Definition 2 *We say that a formula is* head-$\mathbf{c}$-plain *if every strictly positively occurrence of an atomic formula in it is* $\mathbf{c}$*-plain.*

For instance, $f(g)=1 \rightarrow h=1$ is head-(f,g,h)-plain, though it is not (f,g,h)-plain.

Theorem 9 *For any head-*$\mathbf{c}$*-plain sentence F that is tight on* $\mathbf{c}$ *and any interpretation I satisfying $\exists xy(x \neq y)$, we have $I \models \mathrm{SM}[F;\mathbf{c}]$ iff $I \models \mathrm{SM}[UF_{\mathbf{c}}(F);\mathbf{c}]$.*

One may wonder if there is any other translation that would work to unfold nested functions. However, it turns out that there is no modular, signature-preserving translation from arbitrary formulas to $\mathbf{c}$-plain formulas while preserving stable models.

Theorem 10 *For any set* $\mathbf{c}$ *of constants, there is no strongly equivalent transformation that turns an arbitrary formula into a* $\mathbf{c}$*-plain formula.*

The proof follows from the following lemma.

Lemma 3 *There is no f-plain formula that is strongly equivalent to $p(f) \wedge p(1) \wedge p(2) \wedge \neg p(3)$.*

Theorem 10 tells us that the set of arbitrary formulas is strictly more expressive than the set of $\mathbf{c}$-plain formulas of the same signature. One application of this greater expressivity is in reducing many-sorted FSM to unsorted FSM in Section 8.1 later.

7 Comparing FSM with Other Approaches to Intensional Functions

7.1 Relation to Nonmonotonic Causal Logic

A *(nonmonotonic) causal theory* is a finite list of rules of the form

$$F \Leftarrow G$$

where F and G are formulas as in first-order logic. We identify a rule with the universal closure of the implication $G \rightarrow F$. A *causal model* of a causal theory T

is defined as the models of the second-order sentence

$$\text{CM}[T; \mathbf{f}] = T \wedge \neg \exists \widehat{\mathbf{f}}(\widehat{\mathbf{f}} \neq \mathbf{f} \wedge T^{\dagger}(\widehat{\mathbf{f}}))$$

where $\mathbf{f}$ is a list of *explainable* function constants, and $T^{\dagger}(\widehat{\mathbf{f}})$ denotes the conjunction of the formulas [12]

$$\widetilde{\forall}(G \to F(\widehat{\mathbf{f}})) \tag{20}$$

for all rules $F \Leftarrow G$ of T. By a *definite* causal theory, we mean the causal theory whose rules have the form either

$$f(\mathbf{t}) = t_1 \Leftarrow B \tag{21}$$

or

$$\bot \Leftarrow B, \tag{22}$$

where f is an explainable function constant, $\mathbf{t}$ is a list of terms that does not contain explainable function constants, and t_1 is a term that does not contain explainable function constants. By *Tr*(T) we denote the theory consisting of the following formulas:

$$\widetilde{\forall}(\neg\neg B \to f(\mathbf{t}) = t_1)$$

for each rule (21) in T, and

$$\widetilde{\forall}\neg B$$

for each rule (22) in T. The causal models of such T coincide with the stable models of *Tr*(T).

Theorem 11 *For any definite causal theory T, $I \models \text{CM}[T; \mathbf{f}]$ iff $I \models \text{SM}[\mathit{Tr}(T); \mathbf{f}]$.*

For non-definite theories, they do not coincide as shown by the following example.

Example 8 *Consider the following non-definite causal theory T:*

$$\neg(f = 1) \Leftarrow \top$$
$$\neg(f = 2) \Leftarrow \top$$

An interpretation I where $|I| = \{1, 2, 3\}$, and $f^I = 3$ is a causal model of T. However, the corresponding formula Tr(T) is equivalent to

$$\neg(f = 1) \wedge \neg(f = 2),$$

which has no stable models.

The following example, a variant of Lin's suitcase example [Lin, 1995], demonstrates some unintuitive behavior of definite causal theories in representing indirect effects of actions, which is not present in the functional stable model semantics.

[12] $\widetilde{\forall}F$ represents the universal closure of F.

Example 9 *Consider the two switches which can be flipped but cannot be both up or down at the same time. If one of them is down and the other is up, the direct effect of flipping only one switch is changing the status of that switch, and the indirect effect is changing the status of the other switch. Let Up*(s,t)*, where* s *is switch* A *or* B*, and* t *is a time stamp* 0 *or* 1*, be object constants whose values are Boolean, let Flip*(s)*, where* s *is switch* A *or* B*, be function constants whose values are Boolean, and let* x, y *be variables ranging over Boolean values. The domain can be formalized in a causal theory as*

$$
\begin{array}{rcll}
\mathit{Up}(s,1)=x & \Leftarrow & \mathit{Up}(s,0)=y \wedge \mathit{Flip}(s)=\mathrm{TRUE} & (x \neq y) \\
\mathit{Up}(s,1)=x & \Leftarrow & \mathit{Up}(s',1)=y & (s \neq s', x \neq y) \\
\mathit{Up}(s,1)=x & \Leftarrow & \mathit{Up}(s,1)=x \wedge \mathit{Up}(s,0)=x & \\
\mathit{Flip}(s)=x & \Leftarrow & \mathit{Flip}(s)=x & \\
\mathit{Up}(A,0)=\mathrm{FALSE} & \Leftarrow & \top & \\
\mathit{Up}(B,0)=\mathrm{TRUE} & \Leftarrow & \top &
\end{array}
$$

There are five causal models as shown in the following table.

	Up(A,0)	*Up(B,0)*	*Flip(A)*	*Flip(B)*	*Up(A,1)*	*Up(B,1)*
I_1	FALSE	TRUE	FALSE	FALSE	FALSE	TRUE
I_2	FALSE	TRUE	FALSE	TRUE	TRUE	FALSE
I_3	FALSE	TRUE	TRUE	FALSE	TRUE	FALSE
I_4	FALSE	TRUE	TRUE	TRUE	TRUE	FALSE
I_5	FALSE	TRUE	FALSE	FALSE	TRUE	FALSE

I_2 *and* I_3 *exhibit the indirect effect of the action Flip. Only* I_5 *is not intuitive because the fluent Up changes its value for no reason.*

In the functional stable model semantics, the domain can be represented as

$$
\begin{array}{rcll}
\mathit{Up}(s,1)=x & \leftarrow & \mathit{Up}(s,0)=y \wedge \mathit{Flip}(s)=\mathrm{TRUE} & (x \neq y) \\
\mathit{Up}(s,1)=x & \leftarrow & \mathit{Up}(s',1)=y & (s \neq s', x \neq y) \\
\{\mathit{Up}(s,1)=x\}^{\mathrm{ch}} & \leftarrow & \mathit{Up}(s,0)=x & \\
\{Flip(s)=x\}^{\mathrm{ch}} & \leftarrow & \top & \\
\mathit{Up}(A,0)=\mathrm{FALSE} & \leftarrow & \top & \\
\mathit{Up}(B,0)=\mathrm{TRUE} & \leftarrow & \top &
\end{array}
$$

The program has four stable models I_1, I_2, I_3, I_4; The unintuitive causal model I_5 is not its stable model.

7.2 Relation to Cabalar Semantics

As mentioned earlier, the stable model semantics by Cabalar [2011] is defined in terms of partial satisfaction, which deviates from classical satisfaction. Bartholomew and Lee [2013c] show its relationship to FSM. There, it is shown that when we consider stable models to be total interpretations only, both semantics coincide on $\mathbf{c}$-plain formulas. Also, F and $UF_{\mathbf{c}}(F)$ have the same stable models under the Cabalar semantics, so any complex formula under the Cabalar semantics can be reduced to a $\mathbf{c}$-plain formula by preserving stable models. Furthermore, partial stable models under the Cabalar semantics can be embedded into FSM by introducing an auxiliary object constant NONE to denote that the function is undefined. Consequently, the Cabalar semantics can be fully embedded into FSM by unfolding using an auxiliary constant. We refer the reader to [Bartholomew and Lee, 2013c, Section 4] for the details.

On the other hand, Theorem 10 of this paper shows that the reverse direction is not possible because the class of $\mathbf{c}$-plain formulas is a restricted subset in the functional stable model semantics, which is not the case with the Cabalar semantics. In other words, non-$\mathbf{c}$-plain formulas are weaker than $\mathbf{c}$-plain formulas under FSM whereas the Cabalar semantics does not distinguish them. For instance, under the Cabalar semantics, the formula $a + b = 5$ in Example 5 has many stable models I as long as $a^I + b^I = 5$; in Example 6, $Dom(a)$ has many stable models rather than simply restricting the value of a to the extent of *Dom*; in Example 7, $f_1 = f_2$ has stable models as long as the functions are assigned the same values instead of merely stating that the functions are synonymous.

We observe that the weaker assertions by non-$\mathbf{c}$-plain formulas are often useful but they are not allowed in the Cabalar semantics. In particular, the use of "sort predicates" as in Example 6 is important in specifying the range of an intensional function, rather than a particular value. [13] The synonymity rule like Example 7 is useful for the design of modular action languages as described in [Lifschitz and Yang, 2011].

[13] In Section 8.1 below, we formally show how to reduce many-sorted FSM into unsorted FSM and notes that the axioms used there is not expressible in the Cabalar semantics.

7.3 Relation to IF-Programs

The functional stable model semantics presented here is inspired by IF-programs from [Lifschitz, 2012], where intensional functions were defined without requiring the complex notion of partial functions and partial satisfaction but instead by imposing the uniqueness of values on *total functions*. It turns out that neither semantics is stronger than the other while they coincide on a certain syntactically restricted class of programs. However, the semantics of IF-programs exhibits an unintuitive behavior.

7.3.1 Review of IF-Programs

We consider rules of the form

$$H \leftarrow B, \tag{23}$$

where H and B are formulas that do not contain $\rightarrow$. As before, we identify a rule with the universal closure of the implication $B \rightarrow H$. An IF-program is a finite conjunction of those rules.

An occurrence of a symbol in a formula is *negated* if it belongs to a subformula that begins with negation, and is *non-negated* otherwise. Let F be a formula, let $\mathbf{f}$ be a list of distinct function constants, and let $\widehat{\mathbf{f}}$ be a list of distinct function variables similar to $\mathbf{f}$. By $F^{\diamond}(\widehat{\mathbf{f}})$ we denote the formula obtained from F by replacing each non-negated occurrence of a member of $\mathbf{f}$ with the corresponding function variable in $\widehat{\mathbf{f}}$. By IF$[F; \mathbf{f}]$ we denote the second-order sentence

$$F \wedge \neg\exists\widehat{\mathbf{f}}(\widehat{\mathbf{f}} \neq \mathbf{f} \wedge F^{\diamond}(\widehat{\mathbf{f}})).$$

According to [Lifschitz, 2012], the $\mathbf{f}$-*stable models* of an IF-program Π are defined as the models of IF$[F; \mathbf{f}]$, where F is the FOL-representation of Π.

7.3.2 Comparison

The definition of the IF operator above looks close to our definition of the SM operator. However, they often behave quite differently.

Example 10 *Let F be the following program*

$$d = 2 \leftarrow c = 1,$$

$$d = 1$$

and let I be an interpretation such that $|I| = \{1, 2\}$, $c^I = 2$ and $d^I = 1$. I is a model of IF$[F; cd]$*, but not a model of* SM$[F; cd]$*. The former is not intuitive from the rationality principle because c does not even appear in the head of a rule.*

Example 11 *Let F be the following program*

$$(c = 1 \vee d = 1) \wedge (c = 2 \vee d = 2)$$

and let I_1 and I_2 be interpretations such that $|I_1| = |I_2| = \{1, 2, 3\}$ and $I_1(c) = 1$, $I_1(d) = 2$, $I_2(c) = 2$, $I_2(d) = 1$. The interpretations I_1 and I_2 are models of $\mathrm{SM}[F; cd]$. *On the other hand,* $\mathrm{IF}[F; cd]$ *has no models.*

Example 12 *Let F_1 be $\neg(c = 1) \leftarrow \top$ and let F_2 be $\bot \leftarrow c = 1$. Under the functional stable model semantics, they are strongly equivalent to each other, and neither of them has a stable model. However, this is not the case with IF-programs. For instance, let I be an interpretation such that $|I| = \{1, 2\}$ and $I(c) = 2$. I satisfies* $\mathrm{IF}[F_2; c]$ *but not* $\mathrm{IF}[F_1; c]$.

While $\bot \leftarrow F$ is a constraint in our formalism, in view of Theorem 1, the last example illustrates that $\bot \leftarrow F$ is not considered a constraint in the semantics of IF-programs. This behavior deviates from the standard stable model semantics. Unlike the functional stable model semantics, in general, it is not obvious how various mathematical results established for the first-order stable model semantics, such as the theorem on strong equivalence [Lifschitz *et al.*, 2001], the theorem on completion [Ferraris *et al.*, 2011], and the splitting theorem [Ferraris *et al.*, 2009], can be extended to the above formalisms on intensional functions.

The following theorem gives a specific form of formulas on which the two semantics agree.

Theorem 12 *Let T be an IF-program whose rules have the form*

$$f(\mathbf{t}) = t_1 \leftarrow \neg\neg B \tag{24}$$

where f is an intensional function constant, $\mathbf{t}$ *and t_1 do not contain intensional function constants, and B is an arbitrary formula. We identify T with the corresponding first-order formula. Then we have $I \models \mathrm{SM}[T; \mathbf{f}]$ iff $I \models \mathrm{IF}[T; \mathbf{f}]$.*

8 Many-Sorted FSM

The following is the standard definition of many-sorted first-order logic. A signature σ is comprised of a set of function and predicate constants and a set of sorts. To every function and predicate constant of arity n, we assign argument sorts $s_1, \dots, s_n$ and to every function constant of arity n, we assign also its value sort s_{n+1}. We assume that there are infinitely many variables for each sort. Atomic formulas are built similar to the standard unsorted logic with the restriction that in a term $f(t_1, \dots, t_n)$ (an atom $p(t_1, \dots, t_n)$, respectively), the sort of t_i must be a subsort of the i-th argument of f (p, respectively). In addition $t_1 = t_2$ is an atomic formula if the sorts and t_1 and t_2 have a common supersort.

A many-sorted interpretation I has a non-empty universe $|I|^s$ for each sort s. When s_1 is a subsort of s_2, an interpretation must satisfy $|I|^{s_1} \subseteq |I|^{s_2}$. The notion of satisfaction is similar to the unsorted case with the restriction that an interpretation maps a term to an element in its associated sort.

The definition of many-sorted FSM is a straightforward extension of unsorted FSM. For any list $\mathbf{c}$ of constants in σ, an interpretation I is a *stable model* of F relative to $\mathbf{c}$ if I satisfies SM$[F; \mathbf{c}]$, where SM$[F; \mathbf{c}]$ is syntactically the same as in Section 3 but formulas are understood as in many-sorted logic.

8.1 Reducing Many-sorted FSM to unsorted FSM

We can turn many-sorted FSM into unsorted FSM as follows. Given a many-sorted signature σ, we define the signature σ^{ns} to contain every function and predicate constant from σ. In addition, for each sort $s \in \sigma$, we add a unary predicate $\mathtt{s}$ to σ^{ns}.

Given a formula F of many-sorted signature σ, we obtain the formula F^{ns} of the unsorted signature σ^{ns} as follows.

We replace every formula $\exists x F(x)$, where x is a variable of sort s, with the formula

$$\exists y(\mathtt{s}(y) \wedge F(y))$$

where y is an unsorted variable and $\mathtt{s}$ is a predicate constant in σ^{ns} corresponding to s in σ. Similarly, we replace every $\forall x\ F(x)$, where x is a variable of sort s, with the formula

$$\forall y(\mathtt{s}(y) \rightarrow F(y)).$$

By SF_σ we denote the conjunction of

- the formulas $\forall y(\mathtt{s}_i(y) \rightarrow \mathtt{s}_j(y))$ for every two sorts s_i and s_j in σ such that s_i is a subsort of s_j ($s_i \neq s_j$),
- the formulas $\exists y\ \mathtt{s}(y)$ for every sort s in σ
- the formulas

$$\forall y_1 \dots y_k(\mathtt{args}_1(y_1) \wedge \dots \wedge \mathtt{args}_k(y_k) \rightarrow \mathtt{vals}(f(y_1, \dots, y_k)))$$

  for each function constant f in σ, where the arity of f is k, and the i-th argument sort of f is $args_i$ and the value sort of f is $vals$.
- the formulas

$$\forall y_1 \dots y_{k+1}(\neg\mathtt{args}_1(y_1) \vee \dots \vee \neg\mathtt{args}_k(y_k) \rightarrow \{f(y_1, \dots, y_k) = y_{k+1}\}^{\mathrm{ch}})$$

for each function constant f in σ, where the arity of f is k and the i-th argument sort of f is $args_i$.

- the formulas

$$\forall y_1 \dots y_k(\neg \mathtt{args}_1(y_1) \vee \dots \vee \neg \mathtt{args}_k(y_k) \rightarrow \{p(y_1, \dots, y_k)\}^{\mathrm{ch}})$$

  for each predicate constant p in σ, where the arity of p is k, and the i-th argument sort of p is $args_i$.

Note that only the first three items are necessary for classical logic but we need to add the fourth and fifth items for the FSM semantics so that the witness J to dispute the stability of I can only disagree with I on the atomic formulas that actually correspond to atomic formulas in the many-sorted setting (which has arguments adhering to the argument sorts). Also note that the formulas in item 3 are not **c**-plain, which illustrates the usefulness of non-**c**-plain formulas.

We map an interpretation I of a many-sorted signature σ to an interpretation I^{ns} of an unsorted signature σ^{ns} as follows. First, the universe $|I^{ns}|$ of σ^{ns} is $\bigcup_{s \text{ is a sort in } \sigma} |I|^s$. We specify that the sort predicates and sorts correspond by defining the extent of sort predicate $\mathtt{s}$ for every sort $s \in \sigma$ as

$$\mathtt{s}^{I^{ns}} = |I|^s.$$

For every function constant f in σ and every tuple $\boldsymbol{\xi}$ comprised of elements from $|I^{ns}|$, we take

$$f^{I^{ns}}(\boldsymbol{\xi}) = \begin{cases} f^I(\boldsymbol{\xi}) & \text{if each } \xi_i \in |I|^{args_i} \text{ where } args_i \text{ is the } i\text{-th argument sort of } f \\ |I^{ns}|_0 & \text{otherwise} \end{cases}$$

where $|I^{ns}|_0$ is an arbitrarily chosen element in the universe $|I^{ns}|$ (we use the same element for every situation this case holds).

For every predicate constant p in σ and every $\boldsymbol{\xi}$, we take

$$p^{I^{ns}}(\boldsymbol{\xi}) = \begin{cases} p^I(\boldsymbol{\xi}) & \text{if each } \xi_i \in |I|^{args_i} \text{ where } args_i \text{ is the } i\text{-th argument sort of } p \\ \text{FALSE} & \text{otherwise.} \end{cases}$$

Note that FALSE was arbitrarily chosen.

The choice of I^{ns} mapping a function whose arguments are not of the intended sort to the value $|I^{ns}|_0$ is arbitrary and so there are many unsorted interpretations that correspond to the many-sorted interpretation. To characterize this one-to-many relationship, we say two unsorted interpretations I and J are *related* with relation R, denoted $R(I, J)$, if for every predicate or function constant c, we have

$c^I(\xi_1, \ldots, \xi_k) = c^J(\xi_1, \ldots, \xi_k)$ whenever each $\xi_i \in args_i$ where $args_i$ is the i-th argument sort of c.

Theorem 13 *Let F be a formula of a many-sorted signature σ, and let $\mathbf{c}$ be a set of function and predicate constants.*

(a) If an interpretation I of signature σ is a model of $\mathrm{SM}[F; \mathbf{c}]$, then I^{ns} is a model of $\mathrm{SM}[F^{ns} \wedge SF_\sigma; \mathbf{c}]$.
(b) If an interpretation L of signature σ^{ns} is a model of $\mathrm{SM}[F^{ns} \wedge SF_\sigma; \mathbf{c}]$ then there is some interpretation I of signature σ such that I is a model of $\mathrm{SM}[F; \mathbf{c}]$ and $R(L, I^{ns})$.

Example 13 *Consider $\sigma = \{s_1, s_2, f/1, 1, 2\}$ where both the argument and the value sort of function constant f are s_1. Take F to be $f(1) = 1 \wedge f(2) = 2$. The many-sorted interpretation I such that $|I|^{s_1} = \{1, 2\}$, $|I|^{s_2} = \{3, 4\}$, $n^I = f^I(n) = n$ for $n \in \{1, 2\}$ is clearly a stable model of F. However, if we drop the last two items of SF_σ, formula $F^{ns} \wedge SF_\sigma$ is*

$$f(1) = 1 \wedge f(2) = 2 \wedge$$
$$\exists y\, \mathtt{s}_1(y) \wedge \exists y\, \mathtt{s}_2(y) \wedge$$
$$\forall y_1(\mathtt{s}_1(y_1) \rightarrow \mathtt{s}_1(f(y_1)))$$

and K is an unsorted interpretation such that $|K| = \{1, 2, 3, 4\}$, $(\mathtt{s}_1)^K = \{1, 2\}$, $(\mathtt{s}_2)^K = \{3, 4\}$, $n^K = n$ for $n \in \{1, 2, 3, 4\}$, $f^K(n) = n$ for $n \in \{1, 2, 3, 4\}$, which is not a stable model of F^{ns} since we can take J that is different from K only on $f(4)$, i.e., $f^J(4) = 3$, to dispute the stability of K.

8.2 Relation to Multi-Valued Propositional Formulas Under the Stable Model Semantics

Multi-valued propositional formulas [Giunchiglia *et al.*, 2004] are an extension of the standard propositional formulas where atomic parts of a formula are equalities of the kind found in constraint satisfaction problems. Action languages such as $\mathcal{C}$+ [Giunchiglia *et al.*, 2004] and $\mathcal{BC}$ [Lee *et al.*, 2013] are defined based on multi-valued propositional formulas. In particular, the latter two languages are defined as shorthand for multi-valued propositional formulas under the stable model semantics, which is a special case of the functional stable model semantics as we show in this section.

A *multi-valued propositional signature* is a set σ of symbols called *multi-valued propositional constants (mvp-constants)*, along with a nonempty finite set *Dom*(c) of symbols, disjoint from σ, assigned to each mvp-constant c. We call *Dom*(c) the *domain* of c. A *multi-valued propositional atom (mvp-atom)* of a signature σ is an expression of the form $c\!=\!v$ ("the value of c is v") where $c \in \sigma$ and $v \in \mathit{Dom}(c)$. A *multi-valued propositional formula (mvp-formula)* of σ is a propositional combination of mvp-atoms.

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

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

Multi-valued propositional formulas can be viewed as a special class of ground first-order formulas of many-sorted signatures. We identify a multi-valued propositional signature with a many-sorted signature that consists of mvp-constants and their values understood as object constants. Each mvp-constant c is identified with an intensional object constant whose sort is $\mathit{Dom}(c)$. Each value in $\mathit{Dom}(c)$ is identified with a non-intensional object constant of the same sort $\mathit{Dom}(c)$, except that if the same value v belongs to multiple domains, the sort of v is the union of the domains.[14] For instance, if $\mathit{Dom}(c_1) = \{1, 2\}$ and $\mathit{Dom}(c_2) = \{2, 3\}$, then the sort of 2 is $\mathit{Dom}(c_1) \cup \mathit{Dom}(c_2)$, while the sort of 1 is $\mathit{Dom}(c_1)$ and the sort of 3 is $\mathit{Dom}(c_2)$. An mvp-atom $c\!=\!v$ is identified with an equality between an intensional object constant c and a non-intensional object constant v.

We identify an mvp-interpretation with the many-sorted interpretation in which each non-intensional object constant is mapped to itself, and is identified with an element in $\mathit{Dom}(c)$ for some intensional object constant c.

It is easy to check that an mvp-interpretation I is a stable model of F in the sense of multi-valued propositional formulas iff I is a stable model of F in the sense of the functional stable model semantics. Under this view, every mvp-formula is identified with a $\mathbf{c}$-plain formula, where $\mathbf{c}$ is the set of all mvp-constants. The elimination of intensional functions in favor of intensional predicates in Section 6.1 essentially turns mvp-formulas into the usual propositional formulas.

[14] This is because in many-sorted logic with ordered sorts, the equality is defined when both terms have the same common supersort.

9 Answer Set Programming Modulo Theories

Sections 5 and 6 show that intensional predicate constants and intensional function constants are interchangeable in many cases. On the other hand, this section shows that considering intensional functions has the computational advantage of making use of efficient computation methods available in the work on satisfiability modulo theories.

We define ASPMT as a special case of many-sorted FSM by restricting attention to interpretations that conform to the background theory.

9.1 ASPMT as a Special Case of the Functional Stable Model Semantics

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].

Let $\sigma^{\mathcal{T}}$ be the many-sorted signature of the background theory $\mathcal{T}$. An interpretation of $\sigma^{\mathcal{T}}$ is called the *background interpretation* if it satisfies the background theory. For instance, in the theory of reals, we assume that $\sigma^{\mathcal{T}}$ 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. A background interpretation interprets these symbols in the standard way.

Let σ be a signature that contains $\sigma^{\mathcal{T}}$. An interpretation of σ is called a $\mathcal{T}$*-interpretation* if it agrees with the fixed background interpretation of $\sigma^{\mathcal{T}}$ on the symbols in $\sigma^{\mathcal{T}}$.

A $\mathcal{T}$-interpretation is a $\mathcal{T}$*-model* of F if it satisfies F.

For any list $\mathbf{c}$ of constants in $\sigma \setminus \sigma^{\mathcal{T}}$, a $\mathcal{T}$-interpretation I is a $\mathcal{T}$*-stable model* of F relative to $\mathbf{c}$ if I satisfies $\mathrm{SM}[F; \mathbf{c}]$.

9.2 Describing Actions in ASPMT

The following example demonstrates how ASPMT can be applied to solve an instance of planning problem with the continuous time that requires real number computation. The encoding extends the standard ASP representation for transition systems [Lifschitz and Turner, 1999].

Example 14 *Consider the following running example from a Texas Action Group* *discussion posted by Vladimir Lifschitz.* [15]

> *A car is on a road of length* L*. If the accelerator is activated, the car will speed up with constant acceleration* A *until the accelerator is released or the car reaches its maximum speed* MS*, whichever comes first. If the brake is activated, the car will slow down with acceleration* $-\mathrm{A}$ *until the brake is released or the car stops, whichever comes first. Otherwise, the speed of the car remains constant. Give a formal representation of this domain, and write a program that uses your representation to generate a plan satisfying the following conditions: at duration 0, the car is at rest at one end of the road; at duration* T*, it should be at rest at the other end.*

This example can be represented in ASPMT as follows. Below s *ranges over time steps,* b *is a Boolean variable,* x, y, a, c, d *are variables over nonnegative reals, and* A *and* MS *are some specific real numbers.*

We represent that the actions Accel and Decel are exogenous and the duration of each time step is to be arbitrarily selected as

$$\{\mathit{Accel}(s) = b\}^{\rm ch},$$
$$\{\mathit{Decel}(s) = b\}^{\rm ch},$$
$$\{\mathit{Duration}(s) = x\}^{\rm ch}.$$

Both Accel and Decel cannot be performed at the same time:

$$\bot \leftarrow \mathit{Accel}(s) = \textsc{true} \wedge \mathit{Decel}(s) = \textsc{true}.$$

The effects of Accel and Decel on Speed are described as

$$\begin{aligned}
\mathit{Speed}(s+1) = y \leftarrow\ & \mathit{Accel}(s){=}\textsc{true}\ \wedge\ \mathit{Speed}(s){=}x\ \wedge\ \mathit{Duration}(s){=}d\\
& \wedge (y = x + \mathrm{A} \times d),\\
\mathit{Speed}(s+1) = y \leftarrow\ & \mathit{Decel}(s){=}\textsc{true}\ \wedge\ \mathit{Speed}(s){=}x\ \wedge\ \mathit{Duration}(s){=}d\\
& \wedge (y = x - \mathrm{A} \times d).
\end{aligned}$$

The preconditions of Accel and Decel are described as

$$\begin{aligned}
\bot \leftarrow\ & \mathit{Accel}(s){=}\textsc{true}\ \wedge\ \mathit{Speed}(s){=}x\ \wedge\ \mathit{Duration}(s){=}d\\
& \wedge (y = x + \mathrm{A} \times d)\ \wedge\ (y > \mathrm{MS}),\\
\bot \leftarrow\ & \mathit{Decel}(s){=}\textsc{true}\ \wedge\ \mathit{Speed}(s){=}x\ \wedge\ \mathit{Duration}(s){=}d\\
& \wedge (y = x - \mathrm{A} \times d)\ \wedge\ (y < 0).
\end{aligned}$$

[15] http://www.cs.utexas.edu/users/vl/tag/continuous_problem

Speed is inertial:

$$\{Speed(s+1)=x\}^{\rm ch} \leftarrow Speed(s)=x.$$

Speed at any moment does not exceed the maximum speed MS*:*

$$\bot \leftarrow Speed(s) > \text{MS}.$$

Location is defined in terms of Speed and Duration as

$$\begin{aligned} Location(s+1)=y \leftarrow\ & Location(s)=x \wedge Speed(s)=a \wedge Speed(s+1)=c \\ & \wedge Duration(s)=d\ \wedge\ y=x+((a+c)/2)\times d. \end{aligned}$$

Theorem 4 tells us that a tight ASPMT theory in Clark normal form can be turned into an SMT instance.

Example 14 Continued *Since the formalization above can be written in Clark Normal Form that is tight, its stable models coincide with the models of the completion formulas. For instance, to form the completion of* $Speed(1)$*, consider the rules that have* $Speed(1)$ *in the head:*

$$\begin{aligned} Speed(1)=y\ \leftarrow\ & Accel(0)=\text{TRUE}\ \wedge\ Speed(0)=x\ \wedge\ Duration(0)=d \\ & \wedge (y=x+\text{A}\times d) \wedge\ (y\le \text{MS}), \\ Speed(1)=y\ \leftarrow\ & Decel(0)=\text{TRUE}\ \wedge\ Speed(0)=x\ \wedge\ Duration(0)=d \\ & \wedge (y=x-\text{A}\times d) \wedge\ (y\ge 0), \\ Speed(1)=y\ \leftarrow\ & Speed(0)=y\ \wedge\ \neg\neg(Speed(1)=y) \end{aligned}$$

($\{c=v\}^{\rm ch} \leftarrow G$ *is strongly equivalent to* $c=v \leftarrow G \wedge \neg\neg(c=v)$*). The completion turns them into the following equivalence:*

$$\begin{aligned} & Speed(1)=y\ \leftrightarrow \\ & \exists xd(\ \ (Accel(0)=\text{TRUE}\ \wedge\ Speed(0)=x\ \wedge\ Duration(0)=d \\ & \qquad\qquad\qquad\qquad \wedge (y=x+\text{A}\times d) \wedge (y\le\text{MS})) \\ & \quad \vee (Decel(0)=\text{TRUE}\ \wedge\ Speed(0)=x\ \wedge\ Duration(0)=d \\ & \qquad\qquad\qquad\qquad \wedge (y=x-\text{A}\times d) \wedge (y\ge 0)) \\ & \quad \vee Speed(0)=y\ \ ). \end{aligned} \tag{25}$$

It is worth noting that most action descriptions can be represented by tight ASPMT theories due to the associated time stamps. In [Lee and Meng, 2013], ASPMT was used as the basis of extending action language $\mathcal{C}$+ [Giunchiglia *et al.*, 2004] to represent the durative action model of PDDL 2.1 [Fox and Long, 2003] and the start-process-stop model of representing continuous changes in PDDL+ [Fox and Long, 2006]. In [Lee *et al.*, 2017], language $\mathcal{C}$+ was further extended to allow ordinary differential equations (ODE), the concept borrowed from SAT modulo ODE. As our action language is based on ASPMT, which in turn is founded on the basis of ASP and SMT, it enjoys the development in SMT solving techniques as well as the expressivity of ASP language.

9.3 Implementations of ASPMT

A few implementations of ASPMT emerged based on the idea that reduces tight ASPMT theories to the input language of SMT solvers. System ASPMT2SMT [Bartholomew and Lee, 2014] is a proof-of-concept implementation of ASPMT by reducing ASPMT programs into the input language of SMT solver Z3, and is shown to effectively handle real number computation for reasoning about continuous changes. The system allows a fragment of ASPMT in the input language, whose syntax resembles ASP rules and which can be effectively translated into the input language of SMT solvers. In particular, the language imposes a syntactic condition that quantified variables can be eliminated by equivalent rewriting.

Wałega *et al.* [2015] extended the system ASPMT2SMT to handle nonmonotonic spatial reasoning that uses both qualitative and quantitative information, where spatial relations are encoded in theory of nonlinear real arithmetic.

In [Lee *et al.*, 2017], based on the recent development in SMT called "Satisfiability Modulo Ordinary Differential Equations (ODE)" [Gao *et al.*, 2013a] and its implementation DREAL [Gao *et al.*, 2013b], the system CPLUS2ASPMT was built on top of ASPMT2SMT. The paper showed that a general class of hybrid automata with non-linear flow conditions and non-convex invariants can be turned into first-order action language $\mathcal{C}$+, and CPLUS2ASPMT can be used to compute the action language modulo ODE by translating $\mathcal{C}$+ into ASPMT. For example, the effect of *Accel* in Example 14 can be represented using ODE as

$$\mathit{Speed}(s{+}1) = x + y \leftarrow \mathit{Accel}(s){=}\mathrm{TRUE} \wedge \mathit{Speed}(s){=}x \wedge \mathit{Duration}(s){=}\delta \wedge \\ y{=}\int_0^{\delta} \mathrm{A}\ dt \ \wedge\ y{\le}\mathrm{MS}.$$

The theory of reals is decidable as shown by Tarski, and some SMT solvers do not always approximate reals with floating point numbers. Even for undecidable theories, such as formulas with trigonometric functions and differential equations, SMT solving techniques ensure certain error-bounds: A δ-complete decision procedure [Gao *et al.*, 2013a] for such an SMT formula F returns false if F is unsatisfiable, and returns true if its syntactic “numerical perturbation” of F by bound δ is satisfiable, where $\delta > 0$ is number provided by the user to bound on numerical errors. This is practically useful since it is not possible to sample exact values of physical parameters in reality. ASPMT is able to take the advantage of the SMT solving techniques whereas it is shown that the ASPMT description of action domains is much more compact than the SMT counterpart.

In [Asuncion *et al.*, 2015], the authors presented the “ordered completion,” that compiles logic programs with convex aggregates into the input language of SMT solvers. The focus there was to compute the standard ASP language using SMT solvers. So unlike the other systems mentioned above, neither intensional functions nor various background theories in SMT were considered there. On the other hand, the input programs are not restricted to tight programs.

10 Comparing ASPMT with Other Approaches to Combining ASP with CSP/SMT

We compare ASPMT with other approaches to combining ASP with CSP/SMT. These approaches can be related to a special case of ASPMT in which all functions are non-intensional.

10.1 Relation to Clingcon Programs

A *constraint satisfaction problem* (CSP) is a tuple (V, D, C), where V is a set of *constraint variables* with their respective *domains* in D, and C is a set of *constraints* that specify some legal assignments of values in the domains to the constraint variables.

A *clingcon program* Π [Gebser *et al.*, 2009] with a constraint satisfaction problem (V, D, C) is a set of rules of the form

$$a \leftarrow B, N, \textit{Cn}, \tag{26}$$

where a is a propositional atom or $\bot$, B is a set of positive propositional literals, N is a set of negative propositional literals, and *Cn* is a set of constraints from C, possibly preceded by *not*.

Clingcon programs can be viewed as ASPMT instances. Below is a reformulation of the semantics using the terminologies in ASPMT. We assume that constraints are expressed by ASPMT sentences of signature $V \cup \sigma^{\mathcal{T}}$, where V is a set of object constants, which is identified with the set of constraint variables V in (V, D, C), whose value sorts are identified with the domains in D; we assume that $\sigma^{\mathcal{T}}$ is disjoint from V and contains all values in D as object constants, and other symbols to represent constraints, such as $+$, $\times$, and $\geq$. In other words, we represent a constraint as a formula $F(v_1, \dots, v_n)$ over $V \cup \sigma^{\mathcal{T}}$ where $F(x_1, \dots, x_n)$ is a formula of the signature $\sigma^{\mathcal{T}}$ and $F(v_1, \dots, v_n)$ is obtained from $F(x_1, \dots, x_n)$ by substituting the object constants $(v_1, \dots, v_n)$ in V for $(x_1, \dots, x_n)$. We say this background theory $\mathcal{T}$ *conforms* to (V, D, C).

For any signature σ that consists of object constants and propositional constants, we identify an interpretation I of σ as the tuple $\langle I^f, X \rangle$, where I^f is the restriction of I onto the object constants in σ, and X is a set of propositional constants in σ that are true under I.

Given a clingcon program Π with (V, D, C), and a $\mathcal{T}$-interpretation $I = \langle I^f, X \rangle$, we define the *constraint reduct of* Π *relative to* X *and* I^f (denoted by $\Pi^X_{I^f}$) as the set of rules $a \leftarrow B$ for each rule (26) in Π such that $I^f \models Cn$, and $X \models N$. We say that a set X of propositional atoms is a *constraint answer set* of Π relative to I^f if X is a minimal model of $\Pi^X_{I^f}$.

Example 1 continued *The rules*

$$Amt_1 =^{\$} Amt_0{+}1 \leftarrow not\ Flush,$$
$$Amt_1 =^{\$} 0 \leftarrow Flush$$

are identified with

$$\bot \leftarrow not\ Flush, not(Amt_1 =^{\$} Amt_0{+}1)$$
$$\bot \leftarrow Flush, not(Amt_1 =^{\$} 0)$$

under the semantics of clingcon programs with the theory of integers as the background theory; Amt_0, Amt_1 are object constants and Flush is a propositional constant. Consider I_1 in Example 1, which can be represented as $\langle (I_1)^f, X \rangle$ where $(I_1)^f$ maps Amt_0 to 5, and Amt_1 to 6, and $X = \emptyset$. The set X is the constraint answer set relative to $(I_1)^f$ because X is the minimal model of the constraint reduct relative to X and $(I_1)^f$, which is the empty set.

Similar to the way that rules are identified as a special case of formulas [Ferraris *et al.*, 2011], we identify a clingcon program Π with the conjunction of implications $B \wedge N \wedge Cn \rightarrow a$ for all rules (26) in Π. The following theorem tells us that clingcon programs are a special case of ASPMT in which the background theory $\mathcal{T}$ conforms to (V, D, C), and intensional constants are limited to propositional constants only, and do not allow function constants, so the language cannot express the default assignment of values to a function.

Theorem 14 *Let Π be a clingcon program with CSP (V, D, C), let $\mathbf{p}$ be the set of all propositional constants occurring in Π, let $\mathcal{T}$ be the background theory con-*

forming to (V, D, C)*, and let* $\langle I^f, X\rangle$ *be a* $\mathcal{T}$*-interpretation. Set* X *is a constraint answer set of* Π *relative to* I^f *iff* $\langle I^f, X\rangle$ *is a* $\mathcal{T}$*-stable model of* Π *relative to* $\mathbf{p}$.

Note that a clingcon program does not allow an atom that consists of elements from both V and $\mathbf{p}$. Thus the truth value of an atom is determined by either I^f or X, but not by involving both of them.

In [Lierler and Susman, 2016], the authors compared Constraint ASP and SMT by relating the different terminologies and concepts used in each of them. This is related to the relationship shown in Theorem 14 since $\mathcal{T}$-stable models of an ASPMT program Π relative to $\emptyset$ are precisely SMT models of Π with background theory $\mathcal{T}$. One main difference between the two comparisons is that an *answer set* in [Lierler and Susman, 2016] is a set containing ordinary atoms and theory/constraint atoms, while a *stable model* in this paper is a classical model.

10.2 Relation to ASP(LC) Programs

Liu *et al.* [2012] consider logic programs with linear constraints, or *ASP(LC)* programs, comprised of rules of the form

$$a \leftarrow B, N, LC \tag{27}$$

where a is a propositional atom or $\bot$, B is a set of positive propositional literals, and N is a set of negative propositional literals, and LC is a set of *theory atoms*—linear constraints of the form $\sum_{i=1}^{n}(c_i \times x_i) \bowtie k$ where $\bowtie \in \{\leq, \geq, =\}$, each x_i is an object constant whose value sort is integers (or reals), and each c_i, k is an integer (or real).

An ASP(LC) program Π can be viewed as an ASPMT formula whose background theory $\mathcal{T}$ is the theory of integers or the theory of reals. We identify an ASP(LC) program Π with the conjunction of ASPMT formulas $B \wedge N \wedge LC \rightarrow a$ for all rules (27) in Π.

An *LJN-intepretation* is a pair (X, T) where X is a set of propositional atoms and T is a subset of theory atoms occurring in Π such that there is some $\mathcal{T}$-interpretation I that satisfies $T \cup \overline{T}$, where $\overline{T}$ is the set of negations of each theory atom occurring in Π but not in T. An LJN-interpretation (X, T) satisfies an atom b if $b \in X$, the negation of an atom $not\ c$ if $c \notin X$, and a theory atom t if $t \in T$. The notion of satisfaction is extended to other propositional connectives as usual.

The *LJN-reduct* of a program Π with respect to an LJN-interpretation (X, T), denoted by $\Pi^{(X,T)}$, consists of rules $a \leftarrow B$ for each rule (27) such that (X, T) satisfies $N \wedge LC$. (X, T) is an *LJN-answer set* of Π if (X, T) satisfies Π, and X is the smallest set of atoms satisfying $\Pi^{(X,T)}$.

The following theorem tells us that there is a one-to-many relationship between LJN-answer sets and the stable models in the sense of ASPMT. Essentially, the set of theory atoms in an LJN-answer set encodes all valid mappings for functions in the stable model semantics.

Theorem 15 *Let Π be an ASP(LC) program of signature $\langle\sigma^p, \sigma^f\rangle$ where σ^p is a set of propositional constants, and let σ^f be a set of object constants, and let I^f be an interpretation of σ^f.*

(a) If (X, T) is an LJN-answer set of Π, then for any $\mathcal{T}$-interpretation I such that $I^f \models T \cup \overline{T}$, we have $\langle I^f, X\rangle \models \mathrm{SM}[\Pi; \sigma^p]$.
(b) For any $\mathcal{T}$-interpretation $I = \langle I^f, X\rangle$, if $\langle I^f, X\rangle \models \mathrm{SM}[\Pi; \sigma^p]$, then an LJN-interpretation (X, T) where

$$T = \{t \mid t \textit{ is a theory atom in } \Pi \textit{ such that } I^f \models t\}$$

is an LJN-answer set of Π.

Example 15 *Let F be*

$$\begin{array}{ll} a \leftarrow x-z>0. & b \leftarrow x-y\leq 0. \\ c \leftarrow b,\ y-z\leq 0. & \leftarrow \textit{not } a. \\ b \leftarrow c. & \end{array}$$

The LJN-interpretation $L = \langle\{a\}, \{x-z>0\}\rangle$ is an answer set of F since $\{(x-z>0, \neg(x-y\leq 0), \neg(y-z\leq 0)\}$ is satisfiable (e.g., take $x^I = 2, y^I = 1, z^I = 0$) and the set $\{a\}$ is the minimal model satisfying the reduct F^L, which is equivalent to $(\top \rightarrow a) \wedge (c \rightarrow b)$. In accordance with Theorem 15, the interpretation I such that $x^I = 2, y^I = 1, z^I = 0, a^I = \mathrm{TRUE}, b^I = \mathrm{FALSE}, c^I = \mathrm{FALSE}$ satisfies $I \models \mathrm{SM}[F; abc]$.

As with clingcon programs, ASP(LC) programs do not allow intensional functions.

10.3 Relation to Lin-Wang Programs

Lin and Wang (2008) extended answer set semantics with functions by extending the definition of a reduct, and also provided loop formulas for such programs. We can provide an alternative account of their results by considering the notions there as special cases of the definitions presented in this paper. Essentially, they restricted attention to a special case of non-Herbrand interpretations such that object constants form the universe, and ground terms other than object constants are mapped to the object constants. More precisely, according to [Lin and Wang, 2008], an *LW-program* P consists of *type definitions* and a set of rules of the form

$$A \leftarrow B_1, \ldots, B_m, not\ C_1, \ldots, not\ C_n \tag{28}$$

where A is $\bot$ or an atom, and B_i ($1 \le i \le m$) and C_j ($1 \le j \le n$) are atomic formulas possibly containing equality. Type definitions are essentially a special case of many-sorted signature declarations, where each sort is a set of object constants. For such many-sorted signature, we say that a many-sorted interpretation I is a *P-interpretation* if it evaluates each object constant to itself, and each ground term other than object constants to an object constant conforming to the type definitions of P. The *functional reduct* of P under I is a normal logic program without functions obtained from P by

(1) replacing each functional term $f(t_1, \ldots, t_n)$ with c where $f^I(t_1, \ldots, t_n) = c$;
(2) removing any rule containing $c \neq c$ or $c = d$ where c, d are distinct constants;
(3) removing any remaining equalities from the remaining rules;
(4) removing any rule containing *not* A in the body of the rule where $A^I = \text{TRUE}$;
(5) removing any remaining *not* A from the bodies of the remaining rules.

A P-interpretation is an answer set of P in the sense of [Lin and Wang, 2008] if I is the minimal model of P^I.

The following theorem tells us that LW programs are a special case of FSM formulas whose function constants are non-intensional.

Theorem 16 *Let P be an LW-program and let F be the FOL-representation of the set of rules in P. The following conditions are equivalent to each other:*

(a) I is an answer set of P in the sense of [Lin and Wang, 2008];
(b) I is a P-interpretation that satisfies $\text{SM}[F; \mathbf{p}]$ *where* $\mathbf{p}$ *is the list of all predicate constants occurring in F.*

In other words, like clingcon programs, Lin-Wang programs can be identified with a special case of the first-order stable model semantics from [Ferraris *et al.*, 2011], which do not allow intensional functions.

11 Conclusion

In this paper, we presented the functional stable model semantics, which properly extends the first-order stable model semantics to distinguish between intensional and non-intensional functions. We observe that many properties known for the first-order stable model semantics naturally extend to the functional stable model semantics.

The presented semantics turns out to be useful for overcoming the limitations of the stable model semantics originating from the propositional setting, and enables us to combine with other related formalisms where general functions play a central role in efficient computation. ASPMT benefits from the expressiveness of ASP modeling language while leveraging efficient constraint/theory solving methods originating from SMT. For instance, it provides a viable approach to nonmonotonic reasoning about hybrid transitions where discrete and continuous changes co-exist.

The relationship between ASPMT and SMT is similar to the relationship between ASP and SAT. We expect that, in addition to completion and the results shown in this paper, many other results known between ASP and SAT can be carried over to the relationship between ASPMT and SMT, thereby contributing to efficient first-order reasoning in answer set programming. A future work is to lift the limitation of the current ASPMT implementation limited to tight programs by designing and implementing a native computation algorithm which borrows the techniques from SMT, similar to the way that ASP solvers adapted SAT solving computation.

Acknowledgements We are grateful to Yi Wang and Nikhil Loney for many useful discussions and to the anonymous referees for their constructive comments. This work was partially supported by the National Science Foundation under Grants IIS-1319794, IIS-1526301, and IIS-1815337.

References

[Asuncion *et al.*, 2015] Vernon Asuncion, Yin Chen, Yan Zhang, and Yi Zhou. Ordered completion for logic programs with aggregates. *Artificial Intelligence*, 224:72–102, 2015.

[Babb and Lee, 2013] Joseph Babb and Joohyung Lee. Cplus2ASP: Computing action language $\mathcal{C}$+ in answer set programming. In *Proceedings of International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR)*, pages 122–134, 2013.

[Balduccini, 2009] 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.

[Balduccini, 2012] 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*, pages 24–39, 2012.

[Barrett *et al.*, 2009] 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.

[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, 2013a] Michael Bartholomew and Joohyung Lee. Functional stable model semantics and answer set programming modulo theories. In *Proceedings of International Joint Conference on Artificial Intelligence (IJCAI)*, 2013.

[Bartholomew and Lee, 2013b] Michael Bartholomew and Joohyung Lee. A functional view of strong negation. In *Working Notes of the 5th Workshop on Answer Set Programming and Other Computing Paradigms (ASPOCP)*, 2013.

[Bartholomew and Lee, 2013c] Michael Bartholomew and Joohyung Lee. On the stable model semantics for intensional functions. *Theory and Practice of Logic Programming*, 13(4-5):863–876, 2013.

[Bartholomew and Lee, 2014] Michael Bartholomew and Joohyung Lee. System ASPMT2SMT: Computing aspmt theories by smt solvers. In *Proceedings of European Conference on Logics in Artificial Intelligence (JELIA)*, pages 529–542, 2014.

[Brewka *et al.*, 2011] Gerhard Brewka, Ilkka Niemelä, and Miroslaw Truszczynski. Answer set programming at a glance. *Communications of the ACM*, 54(12):92–103, 2011.

[Cabalar, 2011] Pedro Cabalar. Functional answer set programming. *Theory and Practice of Logic Programming*, 11(2-3):203–233, 2011.

[Clark, 1978] Keith Clark. Negation as failure. In Herve Gallaire and Jack Minker, editors, *Logic and Data Bases*, pages 293–322. Plenum Press, New York, 1978.

[Ferraris *et al.*, 2007] Paolo Ferraris, Joohyung Lee, and Vladimir Lifschitz. A new perspective on stable models. In *Proceedings of International Joint Conference on Artificial Intelligence (IJCAI)*, pages 372–379, 2007.

[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.

[Fox and Long, 2003] Maria Fox and Derek Long. PDDL2.1: An extension to PDDL for expressing temporal planning domains. *J. Artif. Intell. Res. (JAIR)*, 20:61–124, 2003.

[Fox and Long, 2006] Maria Fox and Derek Long. Modelling mixed discrete-continuous domains for planning. *J. Artif. Intell. Res. (JAIR)*, 27:235–297, 2006.

[Gao *et al.*, 2013a] Sicun Gao, Soonho Kong, and Edmund Clarke. Satisfiability modulo ODEs. *arXiv preprint arXiv:1310.8278*, 2013.

[Gao *et al.*, 2013b] Sicun Gao, Soonho Kong, and Edmund M Clarke. dReal: An SMT solver for nonlinear theories over the reals. In *International Conference on Automated Deduction*, pages 208–214. Springer Berlin Heidelberg, 2013.

[Gebser *et al.*, 2009] M. Gebser, M. Ostrowski, and T. Schaub. Constraint answer set solving. In *Proceedings of International Conference on Logic Programming (ICLP)*, pages 235–249, 2009.

[Gelfond and Kahl, 2014] Michael Gelfond and Yulia Kahl. *Knowledge Representation, Reasoning, and the Design of Intelligent Agents*. Cambridge University Press, 2014.

[Gelfond and Lifschitz, 1988] Michael Gelfond and Vladimir Lifschitz. The stable model semantics for logic programming. In Robert Kowalski and Kenneth Bowen, editors, *Proceedings of International Logic Programming Conference and Symposium*, pages 1070–1080. MIT Press, 1988.

[Giunchiglia *et al.*, 2004] Enrico Giunchiglia, Joohyung Lee, Vladimir Lifschitz, Norman McCain, and Hudson Turner. Nonmonotonic causal theories. *Artificial Intelligence*, 153(1–2):49–104, 2004.

[Janhunen *et al.*, 2011] Tomi Janhunen, 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 and Meng, 2013] Joohyung Lee and Yunsong Meng. Answer set programming modulo theories and reasoning about continuous changes. In *Proceedings of International Joint Conference on Artificial Intelligence (IJCAI)*, 2013.

[Lee *et al.*, 2013] Joohyung Lee, Vladimir Lifschitz, and Fangkai Yang. Action language $\mathcal{BC}$: Preliminary report. In *Proceedings of International Joint Conference on Artificial Intelligence (IJCAI)*, 2013.

[Lee *et al.*, 2017] Joohyung Lee, Nikhil Loney, and Yunsong Meng. Representing hybrid automata by action language modulo theories. *Theory and Practice of Logic Programming*, 2017.

[Lierler and Susman, 2016] Yuliya Lierler and Benjamin Susman. Constraint answer set programming versus satisfiability modulo theories. In *IJCAI*, pages 1181–1187, 2016.

[Lifschitz and Turner, 1999] Vladimir Lifschitz and Hudson Turner. Representing transition systems by logic programs. In *Proceedings of International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR)*, pages 92–106, 1999.

[Lifschitz and Yang, 2011] Vladimir Lifschitz and Fangkai Yang. Eliminating function symbols from a nonmonotonic causal theory. In Gerhard Lakemeyer and Sheila A. McIlraith, editors, *Knowing, Reasoning, and Acting: Essays in Honour of Hector J. Levesque*. College Publications, 2011.

[Lifschitz and Yang, 2013] Vladimir Lifschitz and Fangkai Yang. Functional completion. *Journal of Applied Non-Classical Logics*, 23(1-2):121–130, 2013.

[Lifschitz *et al.*, 2001] Vladimir Lifschitz, David Pearce, and Agustin Valverde. Strongly equivalent logic programs. *ACM Transactions on Computational Logic*, 2:526–541, 2001.

[Lifschitz, 1988] Vladimir Lifschitz. On the declarative semantics of logic programs with negation. In Jack Minker, editor, *Foundations of Deductive Databases and Logic Programming*, pages 177–192. Morgan Kaufmann, San Mateo, CA, 1988.

[Lifschitz, 1994] Vladimir Lifschitz. Circumscription. In D.M. Gabbay, C.J. Hogger, and J.A. Robinson, editors, *Handbook of Logic in AI and Logic Programming*, volume 3, pages 298–352. Oxford University Press, 1994.

[Lifschitz, 1997] Vladimir Lifschitz. On the logic of causal explanation. *Artificial Intelligence*, 96:451–465, 1997.

[Lifschitz, 2008] Vladimir Lifschitz. What is answer set programming? In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 1594–1597. MIT Press, 2008.

[Lifschitz, 2011] Vladimir Lifschitz. Datalog programs and their stable models. In O. de Moor, G. Gottlob, T. Furche, and A. Sellers, editors, *Datalog Reloaded: First International Workshop, Datalog 2010, Oxford, UK, March 16-19, 2010. Revised Selected Papers*. Springer, 2011.

[Lifschitz, 2012] Vladimir Lifschitz. Logic programs with intensional functions. In *Proceedings of International Conference on Principles of Knowledge Representation and Reasoning (KR)*, pages 24–31, 2012.

[Lin and Wang, 2008] Fangzhen Lin and Yisong Wang. Answer set programming with functions. In *Proceedings of International Conference on Principles of Knowledge Representation and Reasoning (KR)*, pages 454–465, 2008.

[Lin, 1995] Fangzhen Lin. Embracing causality in specifying the indirect effects of actions. In *Proceedings of International Joint Conference on Artificial Intelligence (IJCAI)*, pages 1985–1991, 1995.

[Liu *et al.*, 2012] Guohua Liu, 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)*, pages 32–42, 2012.

[McCarthy, 1980] John McCarthy. Circumscription—a form of non-monotonic reasoning. *Artificial Intelligence*, 13:27–39,171–172, 1980.

[Mellarkod *et al.*, 2008] Veena S Mellarkod, Michael Gelfond, and Yuanlin Zhang. Integrating answer set programming and constraint logic programming. *Annals of Mathematics and Artificial Intelligence*, 53(1-4):251–287, 2008.

[Wałega *et al.*, 2015] Przemysław Andrzej Wałega, Mehul Bhatt, and Carl Schultz. ASPMT(QS): non-monotonic spatial reasoning with answer set programming modulo theories. In *Logic Programming and Nonmonotonic Reasoning*, pages 488–501. Springer, 2015.

A Review of Reduct-Based Definition of Stable Models

Some of the proofs below use the definition of functional stable models based on the notions of an infinitary ground formula and a reduct from [Bartholomew and Lee, 2013c]. We review the semantics below.

A.1 *Infinitary Ground Formulas*

We assume that a signature and an interpretation are defined the same as in the standard first-order logic. For each element ξ in the universe $|I|$ of I, we introduce a new symbol $\xi^\diamond$, called an *object name*. By σ^I we denote the signature obtained from σ by adding all object names $\xi^\diamond$ as additional object constants. We will identify an interpretation I of signature σ with its extension to σ^I defined by $I(\xi^\diamond) = \xi$.

We assume the primary connectives of infinitary ground formulas to be $\bot$, $\{\}^\wedge$, $\{\}^\vee$, and $\rightarrow$. The usual propositional connectives $\wedge, \vee$ are considered as shorthands: $F \wedge G$ as $\{F, G\}^\wedge$, and $F \vee G$ as $\{F, G\}^\vee$.

Let A be the set of all ground atomic formulas of signature σ^I. The sets $\mathcal{F}_0, \mathcal{F}_1, \dots$ are defined recursively as follows:

- $\mathcal{F}_0 = A \cup \{\bot\}$;
- $\mathcal{F}_{i+1} (i \geq 0)$ consists of expressions $\mathcal{H}^\vee$ and $\mathcal{H}^\wedge$, for all subsets $\mathcal{H}$ of $\mathcal{F}_0 \cup \ldots \cup \mathcal{F}_i$, and of the expressions $F \rightarrow G$, where F and G belong to $\mathcal{F}_0 \cup \cdots \cup \mathcal{F}_i$.

We define $\mathcal{L}_A^{inf} = \bigcup_{i=0}^{\infty} \mathcal{F}_i$, and call elements of $\mathcal{L}_A^{inf}$ *infinitary ground formulas* of σ w.r.t. I.

For any interpretation I of σ and any infinitary ground formula F w.r.t. I, the definition of satisfaction, $I \models F$, is as follows:

- For atomic formulas, the definition of satisfaction is the same as in the standard first-order logic;
- $I \models \mathcal{H}^\vee$ if there is a formula $G \in \mathcal{H}$ such that $I \models G$;
- $I \models \mathcal{H}^\wedge$ if, for every formula $G \in \mathcal{H}$, $I \models G$;
- $I \models G \rightarrow H$ if $I \not\models G$ or $I \models H$.

Given an interpretation, we identify any first-order sentence with an infinitary ground formula via the process of grounding relative to that interpretation. Let F be any first-order sentence of a signature σ, and let I be an interpretation of σ. By $gr_I[F]$ we denote the infinitary ground formula w.r.t. I that is obtained from F by the following process:

- If F is an atomic formula, $gr_I[F]$ is F;
- $gr_I[G \odot H] = gr_I[G] \odot gr_I[H] \quad (\odot \in \{\wedge, \vee, \rightarrow\})$;
- $gr_I[\exists x G(x)] = \{gr_I[G(\xi^\diamond)] \mid \xi \in |I|\}^\vee$;
- $gr_I[\forall x G(x)] = \{gr_I[G(\xi^\diamond)] \mid \xi \in |I|\}^\wedge$.

A.2 Stable Models in terms of Grounding and Reduct

For any two interpretations I, J of the same signature and any list $\mathbf{c}$ of distinct predicate and function constants, we write $J <^{\mathbf{c}} I$ if

- J and I have the same universe and agree on all constants not in $\mathbf{c}$,
- $p^J \subseteq p^I$ for all predicate constants p in $\mathbf{c}$,[16] and
- J and I do not agree on $\mathbf{c}$.

The *reduct* $F^{\underline{I}}$ of an infinitary ground formula F relative to an interpretation I is defined as follows:

- For any atomic formula F, $F^{\underline{I}} = \begin{cases} \bot & \text{if } I \not\models F \\ F & \text{otherwise.} \end{cases}$
- $(\mathcal{H}^\wedge)^{\underline{I}} = \{G^{\underline{I}} \mid G \in \mathcal{H}\}^\wedge$
- $(\mathcal{H}^\vee)^{\underline{I}} = \{G^{\underline{I}} \mid G \in \mathcal{H}\}^\vee$
- $(G \rightarrow H)^{\underline{I}} = \begin{cases} \bot & \text{if } I \not\models G \rightarrow H \\ G^{\underline{I}} \rightarrow H^{\underline{I}} & \text{otherwise.} \end{cases}$

The following theorem presents an alternative definition of a stable model that is equivalent to the one in the previous section.

Theorem 17 (Theorem 1 from [Bartholomew and Lee, 2013c]) *Let F be a sentence and let $\mathbf{c}$ be a list of intensional constants. An interpretation I satisfies* $\mathrm{SM}[F; \mathbf{c}]$ *iff*

- *I satisfies F, and*
- *no interpretation J such that $J <^{\mathbf{c}} I$ satisfies $(gr_I[F])^{\underline{I}}$.*

[16] For any symbol c in a signature, c^I denotes the evaluation of I on c.

B Proofs

B.1 Proof of Theorem 1

Theorem 1 *For any first-order formulas F and G, if G is negative on $\mathbf{c}$, $\mathrm{SM}[F \wedge G; \mathbf{c}]$ is equivalent to $\mathrm{SM}[F; \mathbf{c}] \wedge G$.*

Proof. By Lemma 2,

$$\begin{array}{rl}\mathrm{SM}[F \wedge \neg G;\ \mathbf{c}] & = F \wedge \neg G \wedge \neg\exists\widehat{\mathbf{c}}((\widehat{\mathbf{c}} < \mathbf{c}) \wedge (F \wedge \neg G)^*(\widehat{\mathbf{c}})) \\ & \Leftrightarrow F \wedge \neg G \wedge \neg\exists\widehat{\mathbf{c}}((\widehat{\mathbf{c}} < \mathbf{c}) \wedge F^*(\widehat{\mathbf{c}}) \wedge \neg G) \\ & \Leftrightarrow F \wedge \neg\exists\widehat{\mathbf{c}}((\widehat{\mathbf{c}} < \mathbf{c}) \wedge F^*(\widehat{\mathbf{c}})) \wedge \neg G \\ & = \mathrm{SM}[F;\ \mathbf{c}] \wedge \neg G.\end{array}$$

∎

B.2 Proof of Theorem 2

Lemma 4 *Choice*$(\mathbf{c})^*(\widehat{\mathbf{c}})$ *is equivalent to*

$$(\mathbf{c}^{pred} \le \widehat{\mathbf{c}}^{pred}) \wedge (\mathbf{c}^{func} = \widehat{\mathbf{c}}^{func}).$$

Proof. *Choice*$(\mathbf{c})$ is the conjunction for each predicate p in $\mathbf{c}^{pred}$ of $\forall\mathbf{x}(p(\mathbf{x}) \vee \neg p(\mathbf{x}))$ and for each function f in $\mathbf{c}^{func}$ of $\forall\mathbf{x}y(f(\mathbf{x}) = y \vee \neg f(\mathbf{x}) = y)$.

First,

$$[\forall\mathbf{x}(p(\mathbf{x}) \vee \neg p(\mathbf{x}))]^*(\widehat{\mathbf{c}})$$

is equivalent to

$$\forall\mathbf{x}(\widehat{p}(\mathbf{x}) \vee (\neg\widehat{p}(\mathbf{x}) \wedge \neg p(\mathbf{x}))),$$

which is further equivalent to

$$\forall\mathbf{x}(p(\mathbf{x}) \rightarrow \widehat{p}(\mathbf{x})),$$

or simply $p \le \widehat{p}$.

Next,

$$[\forall\mathbf{x}y(f(\mathbf{x}) = y \vee \neg(f(\mathbf{x}) = y))]^*(\widehat{\mathbf{c}})$$

is equivalent to

$$\forall\mathbf{x}y((\widehat{f}(\mathbf{x}) = y \wedge f(\mathbf{x}) = y) \vee (\neg(\widehat{f}(\mathbf{x}) = y) \wedge \neg(f(\mathbf{x}) = y))),$$

which is further equivalent to

$$\forall \mathbf{x} y(f(\mathbf{x}) = y \leftrightarrow \widehat{f}(\mathbf{x}) = y),$$

or simply $f = \widehat{f}$.

Thus, $\mathit{Choice}(\mathbf{c})^*(\widehat{\mathbf{c}})$ is the conjunction for each predicate p in $\mathbf{c}^{pred}$ of $p \leq \widehat{p}$ and for each function f in $\mathbf{c}^{func}$ of $f = \widehat{f}$, or simply $\mathit{Choice}(\mathbf{c})^*(\widehat{\mathbf{c}})$ is

$$(\mathbf{c}^{pred} \leq \widehat{\mathbf{c}}^{pred}) \wedge (\mathbf{c}^{func} = \widehat{\mathbf{c}}^{func}). \quad \blacksquare$$

Theorem 2 *For any first-order formula F and any disjoint lists $\mathbf{c}$, $\mathbf{d}$ of distinct constants, the following formulas are logically valid:*

$$(i) \quad \text{SM}[F; \mathbf{cd}] \rightarrow \text{SM}[F; \mathbf{c}]$$

$$(ii) \quad \text{SM}[F \wedge \mathit{Choice}(\mathbf{d}); \mathbf{cd}] \leftrightarrow \text{SM}[F; \mathbf{c}].$$

Proof. The proof is not long, but there is a notational difficulty that we need to overcome before we can present it. The notation $F^*(\widehat{\mathbf{c}})$ does not take into account the fact that the construction of this formula depends on the choice of the list $\mathbf{c}$ of intensional constants. Since the dependence on $\mathbf{c}$ is essential in the proof of Theorem 2, we use here the more elaborate notation $F^{*[\mathbf{c}]}(\widehat{\mathbf{c}})$. For instance, if F is $p(x) \wedge q(x)$ then

$$F^{*[p]}(\widehat{p}) \text{ is } \widehat{p}(x) \wedge q(x),$$

$$F^{*[pq]}(\widehat{p}, \widehat{q}) \text{ is } \widehat{p}(x) \wedge \widehat{q}(x).$$

It is easy to verify by induction on F that for any disjoint lists $\mathbf{c}$, $\mathbf{d}$ of distinct predicate constants,

$$F^{*[\mathbf{c}]}(\widehat{\mathbf{c}}) \; = \; F^{*[\mathbf{cd}]}(\widehat{\mathbf{c}}, \mathbf{d}). \tag{B.1}$$

(i) In the notation introduced above, $\text{SM}[F; \mathbf{c}]$ is

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

By (B.1), this formula can be written also as

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

which is equivalent to

$$F \wedge \neg \exists \widehat{\mathbf{c}}(((\widehat{\mathbf{c}}, \mathbf{d}) < (\mathbf{c}, \mathbf{d})) \wedge F^{*[\mathbf{cd}]}(\widehat{\mathbf{c}}, \mathbf{d})). \tag{B.2}$$

On the other hand, SM$[F; \mathbf{cd}]$ is

$$F \wedge \neg \exists \widehat{\mathbf{c}}\widehat{\mathbf{d}}(((\widehat{\mathbf{c}}, \widehat{\mathbf{d}}) < (\mathbf{c}, \mathbf{d})) \wedge F^{*[\mathbf{cd}]}(\widehat{\mathbf{c}}, \widehat{\mathbf{d}})). \tag{B.3}$$

It is clear that (B.3) entails (B.2).

(ii) Note that, by (B.1) and Lemma 4, the formula

$$\exists \widehat{\mathbf{c}}\widehat{\mathbf{d}}(((\widehat{\mathbf{c}}, \widehat{\mathbf{d}}) < (\mathbf{c}, \mathbf{d})) \wedge F^{*[\mathbf{cd}]}(\widehat{\mathbf{c}}, \widehat{\mathbf{d}}) \wedge \mathit{Choice}(\mathbf{d})^{*[\mathbf{cd}]}(\widehat{\mathbf{c}}, \widehat{\mathbf{d}}))$$

is equivalent to

$$\exists \widehat{\mathbf{c}}\widehat{\mathbf{d}}(((\widehat{\mathbf{c}}, \widehat{\mathbf{d}}) < (\mathbf{c}, \mathbf{d})) \wedge F^{*[\mathbf{cd}]}(\widehat{\mathbf{c}}, \widehat{\mathbf{d}}) \wedge (\mathbf{d} = \widehat{\mathbf{d}})).$$

It follows that it can be also equivalently rewritten as

$$\exists \widehat{\mathbf{c}}((\widehat{\mathbf{c}} < \mathbf{c}) \wedge F^{*[\mathbf{cd}]}(\widehat{\mathbf{c}}, \mathbf{d})).$$

By (B.1), the last formula can be represented as

$$\exists \widehat{\mathbf{c}}((\widehat{\mathbf{c}} < \mathbf{c}) \wedge F^{*[\mathbf{c}]}(\widehat{\mathbf{c}})).$$

∎

B.3 Proof of Theorem 3

Recall that about first-order formulas F and G we say that F is *strongly equivalent* to G if, for any formula H, any occurrence of F in H, and any list $\mathbf{c}$ of distinct predicate and function constants, SM$[H; \mathbf{c}]$ is equivalent to SM$[H'; \mathbf{c}]$, where H' is obtained from H by replacing the occurrence of F by G.

Lemma 5 *Formula*

$$(F \leftrightarrow G) \wedge ((F^*(\widehat{\mathbf{c}}) \leftrightarrow G^*(\widehat{\mathbf{c}})) \rightarrow (H^*(\widehat{\mathbf{c}}) \leftrightarrow (H')^*(\widehat{\mathbf{c}})))$$

is logically valid.

Proof. By induction on the structure of H. ∎

The following lemma is equivalent to the "only if" part of Theorem 3.

Lemma 6 *If the formula (9) is logically valid, then F is strongly equivalent to G.*

Proof. Assume that (9) is logically valid. We need to show that

$$H \wedge \neg \exists \widehat{\mathbf{c}}((\widehat{\mathbf{c}} < \mathbf{c}) \wedge H^*(\widehat{\mathbf{c}})) \tag{B.4}$$

is equivalent to

$$H' \wedge \neg\exists\widehat{\mathbf{c}}((\widehat{\mathbf{c}} < \mathbf{c}) \wedge (H')^*(\widehat{\mathbf{c}})). \tag{B.5}$$

Since (9) is logically valid, the first conjunctive term of (B.4) is equivalent to the first conjunctive term of (B.5). By Lemma 5, it also follows that the same relationship holds between the two second conjunctive terms of the same formulas. ∎

The following lemma is equivalent to the "if" part of Theorem 3.

Lemma 7 *If F is strongly equivalent to G, then (9) is logically valid.*

Proof. Let C be the formula *Choice*$(\mathbf{c})$. Let E stand for $F \leftrightarrow G$, and E' be $F \leftrightarrow F$. Since F is strongly equivalent to G, the formula SM$[E \leftrightarrow C]$ is equivalent to SM$[E' \leftrightarrow C]$.

Recall that by Lemma 4, *Choice*$(\mathbf{c})^*(\widehat{\mathbf{c}})$, which we abbreviate as C^*, is equivalent to

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

On the other hand, $\widehat{\mathbf{c}} < \mathbf{c}$ can be equivalently rewritten as

$$(\widehat{\mathbf{c}}^{pred} < \mathbf{c}^{pred}) \vee ((\widehat{\mathbf{c}}^{pred} = \mathbf{c}^{pred}) \wedge (\widehat{\mathbf{c}}^{func} \neq \mathbf{c}^{func})).$$

It follows that

$$\widehat{\mathbf{c}} < \mathbf{c} \rightarrow (C^* \leftrightarrow \bot)$$

is logically valid.

It is easy to see that $(E \leftrightarrow C)^*$ can be rewritten as

$$E \wedge (E^*(\widehat{\mathbf{c}}) \leftrightarrow C^*),$$

and that $E^*(\widehat{\mathbf{c}})$ is equivalent to

$$E \wedge (F^*(\widehat{\mathbf{c}}) \leftrightarrow G^*(\widehat{\mathbf{c}})).$$

Using these two facts and Lemma 1, we can simplify SM$[E \leftrightarrow C]$ as follows:

$$\begin{aligned}
\text{SM}[E \leftrightarrow C] &\Leftrightarrow (E \leftrightarrow C) \wedge \neg\exists\widehat{\mathbf{c}}((\widehat{\mathbf{c}} < \mathbf{c}) \wedge E \wedge (E^*(\widehat{\mathbf{c}}) \leftrightarrow C^*)) \\
&\Leftrightarrow E \wedge \neg\exists\widehat{\mathbf{c}}((\widehat{\mathbf{c}} < \mathbf{c}) \wedge (E^*(\widehat{\mathbf{c}}) \leftrightarrow \bot)) \\
&\Leftrightarrow E \wedge \neg\exists\widehat{\mathbf{c}}((\widehat{\mathbf{c}} < \mathbf{c}) \wedge \neg E^*(\widehat{\mathbf{c}})) \\
&\Leftrightarrow E \wedge \neg\exists\widehat{\mathbf{c}}((\widehat{\mathbf{c}} < \mathbf{c}) \wedge \neg(F^*(\widehat{\mathbf{c}}) \leftrightarrow G^*(\widehat{\mathbf{c}}))) \\
&= (F \leftrightarrow G) \wedge \forall\widehat{\mathbf{c}}((\widehat{\mathbf{c}} < \mathbf{c}) \rightarrow (F^*(\widehat{\mathbf{c}}) \leftrightarrow G^*(\widehat{\mathbf{c}}))).
\end{aligned}$$

X U Y' |= (F*(c) <-> G*(c)) & (F <->G)

Similarly, SM$[E' \leftrightarrow C]$ is equivalent to

$$(F \leftrightarrow F) \wedge \forall\widehat{\mathbf{c}}((\widehat{\mathbf{c}} < \mathbf{c}) \rightarrow (F^*(\widehat{\mathbf{c}}) \leftrightarrow F^*(\widehat{\mathbf{c}}))),$$

which is logically valid. Consequently, (9) is logically valid also. ∎

Theorem 3 *Let F and G be first-order formulas, let $\mathbf{c}$ be the list of all constants occurring in F or G, and let $\widehat{\mathbf{c}}$ be a list of distinct predicate and function variables corresponding to $\mathbf{c}$. The following conditions are equivalent to each other.*

- *F and G are strongly equivalent to each other;*
- *Formula*
$$(F \leftrightarrow G) \wedge (\widehat{\mathbf{c}} < \mathbf{c} \rightarrow (F^*(\widehat{\mathbf{c}}) \leftrightarrow G^*(\widehat{\mathbf{c}})))$$
is logically valid.

Proof. Immediate from Lemma 6 and Lemma 7. ∎

B.4 Proof of Theorem 4

Lemma 8 *For any first-order sentence F, any list $\mathbf{c}$ of constants, and any interpretations I and J such that $J <^{\mathbf{c}} I$, if $I \models gr_I(F)^{\underline{I}}$ and $J \not\models gr_I(F)^{\underline{I}}$, then there is some constant d occurring strictly positively in F such that $d^I \neq d^J$.*

Proof. By induction on the structure of F. ∎

Lemma 9 *If a ground formula F is negative on a list $\mathbf{c}$ of predicate and function constants, then for every $J <^{\mathbf{c}} I$,*

$$J \models F^{\underline{I}} \textit{ iff } I \models F.$$

Proof. By induction on the structure of F. ∎

Theorem 4 *For any formula F in Clark normal form relative to $\mathbf{c}$ that is tight on $\mathbf{c}$, an interpretation I that satisfies $\exists xy(x \neq y)$ is a model of* SM$[F; \mathbf{c}]$ *iff I is a model of* COMP$[F; \mathbf{c}]$.

Proof. In this proof, we use Theorem 17 and refer to the reduct-based characterization of a stable model.

($\Leftarrow$) Take an interpretation I that is a model of COMP$[F; \mathbf{c}]$. I is clearly a model of F. We wish to show that, for any interpretation J such that $J <^{\mathbf{c}} I$, we have $J \not\models gr_I[F]^{\underline{I}}$. Let S be a subset of $\mathbf{c}$ consisting of constants c on which I and J disagree, that is, $c^I \neq c^J$. Let s_0 be a constant from S such that there is no edge in the dependency graph from s_0 to any constant in S. Such an s_0 is guaranteed to exist since F is tight on $\mathbf{c}$.

If s_0 is a predicate, then for some ξ, we have $s_0(\xi)^I = \text{TRUE}$ and $s_0(\xi)^J = \text{FALSE}$ by definition of $J <^{\mathbf{c}} I$. If s_0 is a function, then for some ξ, we have $s_0(\xi)^I = v$ and $s_0(\xi)^J \neq v$.

Since F is in Clark normal form, there must be a rule in $gr_I[F]$ of the form $B \rightarrow s_0(\xi^\diamond)$ if s_0 is a predicate ($B \rightarrow s_0(\xi^\diamond) = v$ if s_0 is a function) where B may be $\top$. Further it must be that $I \models B$ since if not, I would not be a model of COMP$[F; \mathbf{c}]$. Thus, the corresponding rule in $gr_I[F]^{\underline{I}}$ is $B^{\underline{I}} \rightarrow s_0(\xi^\diamond)$ ($B^{\underline{I}} \rightarrow s_0(\xi^\diamond) = v$ if s_0 is a function).

Now there are two cases to consider:

- Case 1: $J \models B^{\underline{I}}$. In this case, $J \not\models B^{\underline{I}} \rightarrow s_0(\xi^\diamond)$ (or $J \not\models B^{\underline{I}} \rightarrow s_0(\xi^\diamond) = v$ if s_0 is a function) and so $J \not\models gr_I[F]^{\underline{I}}$.
- Case 2: $J \not\models B^{\underline{I}}$. By Lemma 8, there is a constant d occurring strictly positively in B that I and J disagree on. However, this means there is an edge from s_0 to d and since I and J disagree on d, d belongs to S which contradicts the fact that s_0 was chosen so that it had no edge to any element in S. Thus this case cannot arise.

($\Rightarrow$) Assume $I \models \text{SM}[F; \mathbf{c}]$. F can be viewed as the conjunction of $\forall \mathbf{x}(H(\mathbf{x}) \leftarrow G(\mathbf{x}))$, where each H is an atomic formula containing each intensional constant c_i. It is sufficient to prove that $I \models \forall \mathbf{x}(H(\mathbf{x}) \rightarrow G(\mathbf{x}))$ for each such formula. Assume for the sake of contradiction that for some formula $\forall \mathbf{x}(H(\mathbf{x}) \rightarrow G(\mathbf{x}))$ whose H contains an intensional constant c, $I \models H(\xi)$ and $I \not\models G(\xi)$ for some list ξ of object names.

Consider an interpretation J that differs from I only in that $J \not\models H(\xi)$. ($I \models \exists xy(x \neq y)$ means there are at least two elements in the universe so this is possible when c is a function constant.)

- Clearly, $J \models (H(\xi) \leftarrow G(\xi))^{\underline{I}}$ because $G(\xi)^{\underline{I}} = \bot$.
- For other rules $H(\xi') \leftarrow G(\xi')$ where ξ' is a list of object names different from ξ, clearly, $J \models H(\xi')$ iff $I \models H(\xi')$. Since G is negative on $\mathbf{c}$ and $J <^{\mathbf{c}} I$, by Lemma 9 we have $I \models G(\xi')$ iff $J \models G(\xi')^{\underline{I}}$. Since $I \models H(\xi') \leftarrow G(\xi')$, it follows that $J \models (H(\xi') \leftarrow G(\xi'))^{\underline{I}}$.
- For all other rules $H'(\xi) \leftarrow G'(\xi)$ whose H' has an intensional constant different from c, we have $I \models H'(\xi) \leftarrow G'(\xi)$. Since $H'(\xi) \leftarrow G'(\xi)$ is negative on $\mathbf{c}$ and $J <^{\mathbf{c}} I$, by Lemma 9, we have $J \models (H'(\xi) \leftarrow G'(\xi))^{\underline{I}}$.

The presence of J contradicts that $I \models \text{SM}[F; \mathbf{c}]$. ∎

Theorem 5 *The set of formulas consisting of*

$$\forall \mathbf{x}(f(\mathbf{x}) = 1 \leftrightarrow p(\mathbf{x})), \tag{B.6}$$

and FC_f *entails*

$$\mathrm{SM}[F; p\mathbf{c}] \leftrightarrow \mathrm{SM}[F^p_f \wedge DF_f; f\mathbf{c}].$$

Proof. For any interpretation I of signature $\sigma \supseteq \{f, p, \mathbf{c}\}$ satisfying (B.6), it is clear that $I \models F$ iff $I \models F^p_f \wedge DF_f$ since DF_f is a tautology and F^p_f is equivalent to F under (B.6). Thus it only remains to be shown that

$$I \models \exists \widehat{p}\widehat{\mathbf{c}}((\widehat{p}\widehat{\mathbf{c}} < p\mathbf{c}) \wedge F^*(\widehat{p}\widehat{\mathbf{c}}))$$

iff

$$I \models \exists \widehat{f}\widehat{\mathbf{c}}((\widehat{f}\widehat{\mathbf{c}} < f\mathbf{c}) \wedge (F^p_f)^*(\widehat{f}\widehat{\mathbf{c}}) \wedge DF^*_f(\widehat{f}\widehat{\mathbf{c}})).$$

Let $\sigma' = \sigma \cup \{g, q, \mathbf{d}\}$ be an extended signature such that $g, q, \mathbf{d}$ are similar to $f, p, \mathbf{c}$ respectively, and do not belong to σ.

($\Rightarrow$) Assume $I \models \exists \widehat{p}\widehat{\mathbf{c}}((\widehat{p}\widehat{\mathbf{c}} < p\mathbf{c}) \wedge F^*(\widehat{p}\widehat{\mathbf{c}}))$. This is equivalent to saying that there is an interpretation J of σ that agrees with I on all constants other than p and $\mathbf{c}$ such that $\mathcal{I} = J^{p\mathbf{c}}_{q\mathbf{d}} \cup I$ of signature σ' satisfies $(q\mathbf{d} < p\mathbf{c} \wedge F^*(q\mathbf{d}))$.

It is sufficient to show that there is an interpretation K of σ that agrees with J on all constants other than f such that $\mathcal{I}' = K^{f\mathbf{c}}_{g\mathbf{d}} \cup I$ of signature σ' satisfies $(g\mathbf{d} < f\mathbf{c} \wedge (F^p_f)^*(g\mathbf{d}) \wedge DF^*_f(g\mathbf{d}))$. We define the interpretation of K on f as follows:

$$f^K(\vec{\xi}) = \begin{cases} 1 & \text{if } p^J(\vec{\xi}) = \text{TRUE} \\ 0 & \text{otherwise.} \end{cases}$$

We now show $\mathcal{I}' \models g\mathbf{d} < f\mathbf{c}$:

- Case 1: $\mathcal{I} \models (q = p)$. Since $\mathcal{I} \models q\mathbf{d} < p\mathbf{c}$, by definition $\mathcal{I} \models \mathbf{d}^{pred} \leq \mathbf{c}^{pred}$ and $\mathcal{I} \models \neg(q\mathbf{d} = p\mathbf{c})$ and since in this case, $\mathcal{I} \models (q = p)$, it must be that $\mathcal{I} \models \neg(\mathbf{d} = \mathbf{c})$. From this, we conclude $\mathcal{I}' \models \neg(g\mathbf{d} = f\mathbf{c})$. Further, since $\mathcal{I}' \models \mathbf{d}^{pred} \leq \mathbf{c}^{pred}$, we conclude $\mathcal{I}' \models g\mathbf{d} < f\mathbf{c}$.
- Case 2: $\mathcal{I} \models \neg(q = p)$. Since $\mathcal{I} \models q\mathbf{d} < p\mathbf{c}$, by definition, $\mathcal{I} \models \mathbf{d}^{pred} \leq \mathbf{c}^{pred}$ and $\mathcal{I} \models (q \leq p)$. Thus, since in this case $\mathcal{I} \models \neg(q = p)$, it must be that $\mathcal{I} \models \exists \mathbf{x}(p(\mathbf{x}) \wedge \neg q(\mathbf{x}))$. From the definition of f^K and from (B.6), this is equivalent to $\mathcal{I}' \models \exists \mathbf{x}(f(\mathbf{x}) = 1 \wedge g(\mathbf{x}) = 0)$. Thus, we conclude $\mathcal{I}' \models \neg(f = g)$ and since $\mathcal{I}' \models \mathbf{d}^{pred} \leq \mathbf{c}^{pred}$, we further conclude that $\mathcal{I}' \models g\mathbf{d} < f\mathbf{c}$.

We now show $\mathcal{I}' \models \mathit{DF}^*_f(g\mathbf{d})$:
Since $\mathcal{I} \models q\mathbf{d} < p\mathbf{c}$, by definition, $\mathcal{I} \models (q \leq p)$, or equivalently $\mathcal{I} \models \forall \mathbf{x}(q(\mathbf{x}) \rightarrow p(\mathbf{x}))$ and by contraposition, $\mathcal{I} \models \forall \mathbf{x}(\neg p(\mathbf{x}) \rightarrow \neg q(\mathbf{x}))$. Finally, by (B.6),$FC_f$, and the definition of f^K, $\mathcal{I}' \models \forall \mathbf{x}(f(\mathbf{x}) = 0 \rightarrow g(\mathbf{x}) = 0)$ or simply $\mathcal{I}' \models \mathit{DF}^*_f(g\mathbf{d})$.

We now show $\mathcal{I}' \models (F^p_f)^*(g\mathbf{d})$ by proving the following:

Claim: $\mathcal{I} \models F^*(q\mathbf{d})$ iff $\mathcal{I}' \models (F^p_f)^*(g\mathbf{d})$.

The proof of the claim is by induction on the structure of F.

- Case 1: F is an atomic formula not containing p. F^p_f is exactly F thus $F^*(q\mathbf{d})$ is exactly $(F^p_f)^*(g\mathbf{d})$ so certainly the claim holds.
- Case 2: F is $p(\mathbf{t})$ where $\mathbf{t}$ contains an intensional function constant from $\mathbf{c}$. $F^*(q\mathbf{d})$ is $p(\mathbf{t}) \wedge q(\mathbf{t}')$ where $\mathbf{t}'$ is the result of replacing all intensional functions from $\mathbf{c}$ occurring in $\mathbf{t}$ with the corresponding function from $\mathbf{d}$. Since F^p_f is $f(\mathbf{t}) = 1$, formula $(F^p_f)^*(g\mathbf{d})$ is $f(\mathbf{t}) = 1 \wedge g(\mathbf{t}') = 1$. The claim follows from (B.6) and the definition of f^K.
- Case 3: F is $p(\mathbf{t})$ where $\mathbf{t}$ does not contain any intensional function constant from $\mathbf{c}$. $F^*(q\mathbf{d})$ is $q(\mathbf{t})$. Since F^p_f is $f(\mathbf{t}) = 1$, formula $(F^p_f)^*(g\mathbf{d})$ is $f(\mathbf{t}) = 1 \wedge g(\mathbf{t}) = 1$. Since $\mathcal{I} \models (q \leq p)$, if $\mathcal{I} \models q(\mathbf{t})$, then $\mathcal{I} \models p(\mathbf{t})$. The claim follows from (B.6) and the definition of f^K.
- The other cases are straightforward from I.H.

($\Leftarrow$) Assume $I \models \exists \widehat{f}\widehat{\mathbf{c}}((\widehat{f}\widehat{\mathbf{c}} < f\mathbf{c}) \wedge (F^p_f)^*(\widehat{f}\widehat{\mathbf{c}}) \wedge \mathit{DF}^*_f(\widehat{f}\widehat{\mathbf{c}}))$. This is equivalent to saying that there is an interpretation J of σ that agrees with I on all constants other than f and $\mathbf{c}$ such that $\mathcal{I} = J^{f\mathbf{c}}_{g\mathbf{d}} \cup I$ of signature σ' satisfies $(g\mathbf{d} < f\mathbf{c}) \wedge (F^p_f)^*(f\mathbf{c}) \wedge \mathit{DF}^*_f(f\mathbf{c})$.

It is sufficient to show that there is an interpretation K of σ that agrees with J on all constants other than p such that $\mathcal{I}' = K^{p\mathbf{c}}_{q\mathbf{d}} \cup I$ of signature σ' satisfies $(q\mathbf{d} < p\mathbf{c} \wedge F^*(q\mathbf{d})$. We define the interpretation of K on p as follows:

$$p^K(\vec{\xi}) = \begin{cases} \text{TRUE} & \text{if } f^J(\vec{\xi}) = 1 \\ \text{FALSE} & \text{otherwise.} \end{cases}$$

We now show $\mathcal{I}' \models q\mathbf{d} < p\mathbf{c}$:

- Case 1: $\mathcal{I} \models (g = f)$. By definition of p^K and by (B.6), in this case, $\mathcal{I} \models q = p$ and in particular, $\mathcal{I} \models q \leq p$. Since $\mathcal{I} \models g\mathbf{d} < f\mathbf{c}$, by definition $\mathcal{I} \models \mathbf{d}^{pred} \leq \mathbf{c}^{pred}$ and $\mathcal{I} \models \neg(g\mathbf{d} = f\mathbf{c})$ and since in this case, $\mathcal{I} \models (g = f)$, it must be that $\mathcal{I} \models \neg(\mathbf{d} = \mathbf{c})$. From this, we conclude $\mathcal{I}' \models \neg(q\mathbf{d} = p\mathbf{c})$. Further, since $\mathcal{I}' \models \mathbf{d}^{pred} \leq \mathbf{c}^{pred}$, we conclude $\mathcal{I}' \models q\mathbf{d} < p\mathbf{c}$.
- Case 2: $\mathcal{I} \models \neg(g = f)$. Since $\mathcal{I} \models \mathit{DF}^*_f(g\mathbf{d})$, it must be that $\mathcal{I} \models \forall \mathbf{x}(f(\mathbf{x}) = 0 \rightarrow g(\mathbf{x}) = 0)$. From this, we conclude by definition of p^K, FC_f (note that

$0 \neq 1$ is essential here) and (B.6) that $\mathcal{I}' \models \forall \mathbf{x}(\neg p(\mathbf{x}) \rightarrow \neg q(\mathbf{x}))$. Equivalently, this is $\mathcal{I}' \models \forall \mathbf{x}(q(\mathbf{x}) \rightarrow p(\mathbf{x}))$ or simply $\mathcal{I}' \models q \leq p$.

Now, since $\mathcal{I} \models FC_f$, then $\mathcal{I} \models \forall \mathbf{x}(f(\mathbf{x}) = 0 \vee f(\mathbf{x}) = 1)$. Thus, for the assumption in this case that $\mathcal{I} \models \neg(g = f)$ to hold, it must be that $\mathcal{I} \models \exists \mathbf{x}(f(\mathbf{x}) = 1 \wedge \neg(g(\mathbf{x}) = 1))$. By defintion of p^K and (B.6), it follows that $\mathcal{I}' \models \exists \mathbf{x}(p(\mathbf{x}) \wedge \neg q(\mathbf{x}))$. Thus, since $\mathcal{I}' \models \neg(q = p)$, then $\mathcal{I}' \models \neg(q\mathbf{d} = p\mathbf{c})$. Also, since $\mathcal{I} \models g\mathbf{d} < f\mathbf{c}$, by definition $\mathcal{I}' \models \mathbf{d}^{pred} \leq \mathbf{c}^{pred}$, and thus we conclude that $\mathcal{I}' \models q\mathbf{d} < p\mathbf{c}$.

The proof of $\mathcal{I}' \models F^*(q\mathbf{d})$ is by induction similar to the proof of the claim above. ∎

B.6 Proof of Corollary 6

For two interpretations I of signature σ_1 and J of signature σ_2, by $I \cup J$ we denote the interpretation of signature $\sigma_1 \cup \sigma_2$ and universe $|I| \cup |J|$ that interprets all symbols occurring only in σ_1 in the same way I does and similarly for σ_2 and J. For symbols appearing in both σ_1 and σ_2, I must interpret these the same as J does, in which case $I \cup J$ also interprets the symbol in this way.

Corollary 6

(a) An interpretation I of the signature of F is a model of $\mathrm{SM}[F; p\mathbf{c}]$ *iff I^p_f is a model of* $\mathrm{SM}[F^p_f \wedge \mathit{DF}_f \wedge \mathit{FC}_f; f\mathbf{c}]$.

(b) An interpretation J of the signature of F^p_f is a model of $\mathrm{SM}[F^p_f \wedge \mathit{DF}_f \wedge \mathit{FC}_f; f\mathbf{c}]$ *iff $J = I^p_f$ for some model I of* $\mathrm{SM}[F; p\mathbf{c}]$.

Proof.

(a⇒) Assume I of the signature of F is a model of $\mathrm{SM}[F; p\mathbf{c}]$. By definition of I^p_f, $I \cup I^p_f \models \forall \mathbf{x}(f(\mathbf{x}) = 1 \leftrightarrow p(\mathbf{x})) \wedge \mathit{FC}_f$. Since $I \models \mathrm{SM}[F; p\mathbf{c}]$, it must be that $I \cup I^p_f \models \mathrm{SM}[F; p\mathbf{c}]$ and further by Theorem 5, $I \cup I^p_f \models \mathrm{SM}[F^p_f \wedge \mathit{DF}_f; f\mathbf{c}]$. By Theorem 1, we have $I \cup I^p_f \models \mathrm{SM}[F^p_f \wedge \mathit{DF}_f \wedge \mathit{FC}_f; f\mathbf{c}]$. Finally, since the signature of I does not contain f, we conclude $I^p_f \models \mathrm{SM}[F^p_f \wedge \mathit{DF}_f \wedge \mathit{FC}_f; f\mathbf{c}]$.

(a⇐) Assume I^p_f is a model of $\mathrm{SM}[F^p_f \wedge \mathit{DF}_f \wedge \mathit{FC}_f; f\mathbf{c}]$. By Theorem 1, I^p_f is a model of $\mathrm{SM}[F^p_f \wedge \mathit{DF}_f; f\mathbf{c}]$. By definition of I^p_f, $I \cup I^p_f \models \forall \mathbf{x}(f(\mathbf{x}) = 1 \leftrightarrow p(\mathbf{x})) \wedge \mathit{FC}_f$. Since $I^p_f \models \mathrm{SM}[F^p_f \wedge \mathit{DF}_f; f\mathbf{c}]$, it must be that $I \cup I^p_f \models \mathrm{SM}[F^p_f \wedge \mathit{DF}_f; f\mathbf{c}]$ and further by Theorem 5, $I \cup I^p_f \models \mathrm{SM}[F; p\mathbf{c}]$. Finally, since the signature of I^p_f does not contain p, we conclude $I \models \mathrm{SM}[F; p\mathbf{c}]$.

(b⇒) Assume an interpretation J of the signature of F^p_f is a model of $\mathrm{SM}[F^p_f \wedge \mathit{DF}_f \wedge \mathit{FC}_f; f\mathbf{c}]$. Let $I = J^f_p$, where J^f_p denotes the interpretation of the signature F obtained from J by replacing f^J with the set p^I that consists of the tuples $\langle\xi_1,\dots,\xi_n\rangle$ for all $\xi_1,\dots,\xi_n$ from the universe of J such that $f^J(\xi_1,\dots,\xi_n)=1$. By definition of I, $I\cup J\models\forall\mathbf{x}(f(\mathbf{x})=1\leftrightarrow p(\mathbf{x}))$. Since $J\models\mathrm{SM}[F_f^p\wedge FC_f\wedge DF_f;f\mathbf{c}]$, it must be that $I\cup J\models\mathrm{SM}[F_f^p\wedge DF_f\wedge FC_f;f\mathbf{c}]$. Since FC_f is comprised of constraints, by Theorem 1, $I\cup J\models\mathrm{SM}[F_f^p\wedge DF_f;f\mathbf{c}]\wedge FC_f$. In particular, $I\cup J\models\mathrm{SM}[F_f^p\wedge DF_f;f\mathbf{c}]$ and further by Theorem 5, $I\cup J\models\mathrm{SM}[F;p\mathbf{c}]$. Finally, since the signature of J does not contain p, we conclude $I\models\mathrm{SM}[F;p\mathbf{c}]$.

(b$\Leftarrow$) Take any I such that $J=I_f^p$ and $I\models\mathrm{SM}[F;p\mathbf{c}]$. By definition of I_f^p, $I\cup J\models\forall\mathbf{x}(f(\mathbf{x})=1\leftrightarrow p(\mathbf{x}))\wedge FC_f$. Since $I\models\mathrm{SM}[F;p\mathbf{c}]$, it must be that $I\cup J\models\mathrm{SM}[F;p\mathbf{c}]$ and further by Theorem 5, $I\cup J\models\mathrm{SM}[F_f^p\wedge DF_f;f\mathbf{c}]$. Since the signature of I does not contain f, we conclude $J\models\mathrm{SM}[F_f^p\wedge DF_f;f\mathbf{c}]$. Finally, since by definition of I_f^p, $J\models FC_f$, and since FC_f is comprised of constraints, by Theorem 1 we conclude $J\models\mathrm{SM}[F_f^p\wedge DF_f\wedge FC_f;f\mathbf{c}]$. ∎

B.7 Proof of Theorem 7

Theorem 7 *For any f-plain formula F, the set of formulas consisting of*

$$\forall\mathbf{x}y(p(\mathbf{x},y)\leftrightarrow f(\mathbf{x})=y) \tag{B.7}$$

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

$$\mathrm{SM}[F;f\mathbf{c}]\leftrightarrow\mathrm{SM}[F_p^f;p\mathbf{c}].$$

Proof. For any interpretation I of signature $\sigma\supseteq\{f,p,\mathbf{c}\}$ satisfying (B.7), it is clear that $I\models F$ iff $I\models F_p^f$ since F_p^f is simply the result of replacing all $f(\mathbf{x})=y$ with $p(\mathbf{x},y)$. Thus it only remains to be shown that

$$I\models\exists\widehat{f}\widehat{\mathbf{c}}((\widehat{f}\widehat{\mathbf{c}}<f\mathbf{c})\wedge F^*(\widehat{f}\widehat{\mathbf{c}}))$$

iff

$$I\models\exists\widehat{p}\widehat{\mathbf{c}}((\widehat{p}\widehat{\mathbf{c}}<p\mathbf{c})\wedge(F_p^f)^*(\widehat{p}\widehat{\mathbf{c}})).$$

Let $\sigma'=\sigma\cup\{g,q,\mathbf{d}\}$ be an extended signature such that $g,q,\mathbf{d}$ are similar to $f,p,\mathbf{c}$ respectively, and do not belong to σ.

($\Rightarrow$) Assume $I\models\exists\widehat{f}\widehat{\mathbf{c}}((\widehat{f}\widehat{\mathbf{c}}<f\mathbf{c})\wedge F^*(\widehat{f},\widehat{\mathbf{c}}))$. This is equivalent to saying that there is an interpretation J of σ that agrees with I on all constants other than f and $\mathbf{c}$ such that $\mathcal{I}=J_{g\mathbf{d}}^{f\mathbf{c}}\cup I$ of signature σ' satisfies $(g\mathbf{d}<f\mathbf{c})\wedge F^*(g\mathbf{d})$.

It is sufficient to show that there is an interpretation K of σ that agrees with J on all constants other than p such that $\mathcal{I}'=K_{q\mathbf{d}}^{p\mathbf{c}}\cup I$ of signature σ' satisfies $(q\mathbf{d}<$

$p\mathbf{c}) \wedge (F^f_p)^*(q\mathbf{d})$. We define the interpretation of K on p as follows:

$$p^K(\vec{\xi}, \xi') = \begin{cases} \text{TRUE} & \text{if } \mathcal{I} \models f(\vec{\xi}) = \xi' \wedge g(\vec{\xi}) = \xi' \\ \text{FALSE} & \text{otherwise.} \end{cases}$$

We first show that if $\mathcal{I} \models (g\mathbf{d} < f\mathbf{c})$ then $\mathcal{I}' \models (q\mathbf{d} < p\mathbf{c})$:
Observe that from the definition of p^K, it follows that $\mathcal{I} \models \forall \mathbf{x}y(q(\mathbf{x}, y) \rightarrow f(\mathbf{x}) = y)$ and from (B.7), this is equivalent to $\forall \mathbf{x}y(q(\mathbf{x}, y) \rightarrow p(\mathbf{x}, y))$ or simply $q \leq p$. Thus, since $\mathcal{I}' \models \mathbf{d}^{pred} \leq \mathbf{c}^{pred}$, we have $\mathcal{I}' \models q\mathbf{d}^{pred} \leq p\mathbf{c}^{pred}$.

- Case 1: $\mathcal{I} \models \forall \mathbf{x}y(f(\mathbf{x}) = y \leftrightarrow g(\mathbf{x}) = y)$.
  In this case it then must be the case that $\mathcal{I} \models \mathbf{d} \neq \mathbf{c}$. Thus it follows that $\mathcal{I}' \models q\mathbf{d} \neq p\mathbf{c}$. Consequently, we conclude that

  $$\mathcal{I}' \models (q\mathbf{d}^{pred} \leq p\mathbf{c}^{pred}) \wedge q\mathbf{d} \neq p\mathbf{c}$$

  or simply, $\mathcal{I}' \models (q\mathbf{d} < p\mathbf{c})$.
- Case 2: $\mathcal{I} \models \neg\forall \mathbf{x}y(f(\mathbf{x}) = y \leftrightarrow g(\mathbf{x}) = y)$.
  In this case it then must be the case that for some $\mathbf{t}$ and c that $\mathcal{I} \models f(\mathbf{t}) = c \wedge g(\mathbf{t}) \neq c$. By the definition of p^K, this means that $q(\mathbf{t}, c)^{\mathcal{I}'} = \text{FALSE}$ but by (B.7), $p(\mathbf{t}, c)^{\mathcal{I}'} = \text{TRUE}$. Therefore, $\mathcal{I}' \models p \neq q$ and thus $\mathcal{I}' \models q\mathbf{d} \neq p\mathbf{c}$. Consequently, we conclude

  $$\mathcal{I}' \models (q\mathbf{d}^{pred} \leq p\mathbf{c}^{pred}) \wedge q\mathbf{d} \neq p\mathbf{c}$$

  or simply, $\mathcal{I}' \models (q\mathbf{d} < p\mathbf{c})$.

We now show that $\mathcal{I} \models (F^f_p)^*(q\mathbf{d})$ by proving the following:

Claim: $\mathcal{I} \models F^*(g\mathbf{d})$ iff $\mathcal{I}' \models (F^f_p)^*(q\mathbf{d})$

The proof of the claim is by induction on the structure of F.

- Case 1: F is an atomic formula not containing f. F^f_p is exactly F thus $F^*(g\mathbf{d})$ is exactly $(F^f_p)^*(q\mathbf{d})$ so certainly the claim holds.
- Case 2: F is $f(\mathbf{t}) = t_1$. $F^*(g\mathbf{d})$ is $f(\mathbf{t}) = t_1 \wedge g(\mathbf{t}) = t_1$. F^f_p is $p(\mathbf{t}, t_1)$ and $(F^f_p)^*(q\mathbf{d})$ is $q(\mathbf{t}, t_1)$. By the definition of p^K, it is clear that $\mathcal{I} \models f(\mathbf{t}) = t_1 \wedge g(\mathbf{t}) = t_1$ iff $\mathcal{I}' \models q(\mathbf{t}, t_1)$, so certainly the claim holds.
- The other cases are straightforward from I.H.

($\Leftarrow$) Assume $\mathcal{I} \models \exists\widehat{p}\widehat{\mathbf{c}}((\widehat{p}\widehat{\mathbf{c}} < p\mathbf{c}) \wedge (F^f_p)^*(\widehat{p}\widehat{\mathbf{c}}))$. This is equivalent to saying that there is an interpretation J of σ that agrees with I on all constants other than p and $\mathbf{c}$ such that $\mathcal{I} = J^{p\mathbf{c}}_{q\mathbf{d}} \cup I$ of signature σ' satisfies $(q\mathbf{d} < p\mathbf{c}) \wedge (F^f_p)^*(q\mathbf{d})$.

It is sufficient to show that there is an interpretation K of σ that agrees with J on all constants other than f such that $\mathcal{I}' = K^{f\mathbf{c}}_{g\mathbf{d}} \cup I$ of signature σ' satisfies

$(g\mathbf{d} < f\mathbf{c}) \wedge F^*(g\mathbf{d})$. We define the interpretation of K on f as follows:

$$f^K(\vec{\xi}) = \begin{cases} \xi' & \text{if } \mathcal{I} \models p(\vec{\xi}, \xi') \wedge q(\vec{\xi}, \xi') \\ \xi'' & \text{if } \mathcal{I} \models p(\vec{\xi}, \xi') \wedge \neg q(\vec{\xi}, \xi') \text{ where } \xi' \neq \xi''. \end{cases}$$

Note that the assumption that there are at least two elements in the universe is essential to this definition. This definition is sound due to $(B.7)$ entailing $\forall\vec{\xi}\exists\xi'(p(\vec{\xi}, \xi'))$.

We first show if $\mathcal{I} \models (q\mathbf{d} < p\mathbf{c})$ then $\mathcal{I}' \models (g\mathbf{d} < f\mathbf{c})$:
Observe that $\mathcal{I} \models (q\mathbf{d} < p\mathbf{c})$ by definition entails $\mathcal{I} \models (q\mathbf{d}^{pred} \leq p\mathbf{c}^{pred})$ and further by definition, $\mathcal{I} \models (\mathbf{d}^{pred} \leq \mathbf{c}^{pred})$ and then since f and g are not predicates, $\mathcal{I}' \models ((g\mathbf{d})^{pred} \leq (f\mathbf{c})^{pred})$.

- Case 1: $\mathcal{I} \models \forall\mathbf{x}y(p(\mathbf{x}, y) \leftrightarrow q(\mathbf{x}, y))$. In this case, $\mathcal{I} \models (p = q)$ so for it to be the case that $\mathcal{I} \models (q\mathbf{d} < p\mathbf{c})$, it must be that $\mathcal{I} \models \neg(\mathbf{c} = \mathbf{d})$. It then follows that $\mathcal{I}' \models \neg(f\mathbf{c} = g\mathbf{d})$. Consequently, in this case, $\mathcal{I}' \models ((g\mathbf{d})^{pred} \leq (f\mathbf{c})^{pred}) \wedge \neg(f\mathbf{c} = g\mathbf{d})$ or simply $\mathcal{I}' \models (g\mathbf{d} < f\mathbf{c})$.
- Case 2: $\mathcal{I} \models \neg\forall\mathbf{x}y(p(\mathbf{x}, y) \leftrightarrow q(\mathbf{x}, y))$. In this case, since $\mathcal{I} \models (q \leq p)$, then it follows that $\exists\mathbf{x}y(p(\mathbf{x}, y) \wedge \neg q(\mathbf{x}, y))$. It follows from the definition of p^K that $\mathcal{I}' \models \exists\mathbf{x}yz((p(\mathbf{x}, y) \leftrightarrow g(\mathbf{x}) = z) \wedge y \neq z)$ and then from (B.7), it follows that $\mathcal{I}' \models \exists\mathbf{x}yz((f(\mathbf{x}) = y \leftrightarrow g(\mathbf{x}) = z) \wedge y \neq z)$ or simply $\mathcal{I}' \models f \neq g$. It then follows that $\mathcal{I}' \models \neg(f\mathbf{c} = g\mathbf{d})$. Consequently, in this case $\mathcal{I}' \models ((g\mathbf{d})^{pred} \leq (f\mathbf{c})^{pred}) \wedge \neg(f\mathbf{c} = g\mathbf{d})$ or simply $\mathcal{I}' \models (g\mathbf{d} < f\mathbf{c})$.

Next, the proof of $\mathcal{I}' \models F^*(g\mathbf{d})$ is by induction similar to the proof of the claim above.

B.8 Proof of Corollary 8

Corollary 8 *Let F be an f-plain sentence.*

(a) 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_p^f is a model of $\mathrm{SM}[F_p^f \wedge \mathit{UEC}_p;\ p\mathbf{c}]$.
(b) An interpretation J of the signature of F_p^f that satisfies $\exists xy(x \neq y)$ is a model of $\mathrm{SM}[F_p^f \wedge \mathit{UEC}_p;\ p\mathbf{c}]$ iff $J = I_p^f$ for some model I of $\mathrm{SM}[F; f\mathbf{c}]$.

Proof.

(a$\Rightarrow$) Assume $I \models \mathrm{SM}[F; f\mathbf{c}] \wedge \exists xy(x \neq y)$. Since $I \models \exists xy(x \neq y)$, $I \cup I_p^f \models \exists xy(x \neq y)$ since by definition of I_p^f, I and I_p^f share the same universe.

By definition of I_p^f, $I \cup I_p^f \models (B.7)$. Since $I \models \mathrm{SM}[F; f\mathbf{c}]$, we have $I \cup I_p^f \models$

SM$[F; f\mathbf{c}]$ and by Theorem 7, we have $I \cup I_p^f \models$ SM$[F_p^f; p\mathbf{c}]$. It's clear that $I \models$ UEC_p, so by Theorem 1, we have $I \cup I_p^f \models$ SM$[F_p^f \wedge UEC_p; p\mathbf{c}]$. Since the signature of I does not contain f, we conclude $I_p^f \models$ SM$[F_p^f \wedge UEC_p; p\mathbf{c}]$.

(a$\Leftarrow$) Assume $I \models \exists xy(x \neq y)$ and $I_p^f \models$ SM$[F_p^f \wedge UEC_p; p\mathbf{c}]$. By Theorem 1, $I_p^f \models$ SM$[F_p^f; p\mathbf{c}]$. Since $I \models \exists xy(x \neq y)$, we have $I \cup I_p^f \models \exists xy(x \neq y)$ since by definition of I_p^f, I and I_p^f share the same universe.

By definition of I_p^f, $I \cup I_p^f \models (B.7)$. Since $I_p^f \models$ SM$[F_p^f; p\mathbf{c}]$, we have $I \cup I_p^f \models$ SM$[F_p^f; p\mathbf{c}]$ and by Theorem 7, we have $I \cup I_p^f \models$ SM$[F; f\mathbf{c}]$. Since the signature of I_p^f does contain f, we conclude $I \models$ SM$[F; f\mathbf{c}]$.

(b$\Rightarrow$) Assume $J \models \exists xy(x \neq y)$ and $J \models$ SM$[F_p^f \wedge UEC_p; p\mathbf{c}]$. Let $I = J_f^p$ where J_f^p denotes the interpretation of the signature of F obtained from J by replacing the set p^J with the function f^I such that $f^I(\xi_1, \dots, \xi_k) = \xi_{k+1}$ for all tuples $\langle \xi_1, \dots, \xi_k, \xi_{k+1} \rangle$ in p^J. This is a valid definition of a function since we assume $J \models$ SM$[F_p^f \wedge UEC_p; p\mathbf{c}]$, from which we obtain by Theorem 1 that $J \models$ SM$[F_p^f; p\mathbf{c}] \wedge UEC_p$ and specifically, $J \models UEC_p$. Clearly, $J = I_p^f$ so it only remains to be shown that $I \models$ SM$[F; f\mathbf{c}]$.

Since I and J have the same universe and $J \models \exists xy(x \neq y)$, it follows that $I \cup J \models \exists xy(x \neq y)$. Also by the definition of J_f^p, we have $I \cup J \models (B.7)$. Thus by Theorem 7, $I \cup J \models$ SM$[F; f\mathbf{c}] \leftrightarrow$ SM$[F_p^f; p\mathbf{c}]$.

Since we assume $J \models$ SM$[F_p^f; p\mathbf{c}]$, it is the case that $I \cup J \models$ SM$[F_p^f; p\mathbf{c}]$ and thus it must be the case that $I \cup J \models$ SM$[F; f\mathbf{c}]$. Now since the signature of J does not contain f, we conclude $I \models$ SM$[F; f\mathbf{c}]$.

(b$\Leftarrow$)Take any I such that $J = I_p^f$ and $I \models$ SM$[F; f\mathbf{c}]$. Since $J \models \exists xy(x \neq y)$ and I and J share the same universe, $I \cup J \models \exists xy(x \neq y)$. By definition of $J = I_p^f$, $I \cup J \models (B.7)$. Thus by Theorem 7, $I \cup J \models$ SM$[F; f\mathbf{c}] \leftrightarrow$ SM$[F_p^f; p\mathbf{c}]$.

Since we assume $I \models$ SM$[F; f\mathbf{c}]$, it is the case that $I \cup J \models$ SM$[F; f\mathbf{c}]$ and thus it must be the case that $I \cup J \models$ SM$[F_p^f; p\mathbf{c}]$. Further, due to the nature of functions, (B.7) entails UEC_p so $I \cup J \models UEC_p$. However since the signature of I does not contain p, we conclude $J \models$ SM$[F_p^f; p\mathbf{c}] \wedge UEC_p$ and since UEC_p is comprised of constraints only, by Theorem 1 $J \models$ SM$[F_p^f \wedge UEC_p; p\mathbf{c}]$. ∎

B.9 Proof of Theorem 9

Theorem 9 *For any head-$\mathbf{c}$-plain sentence F that is tight on $\mathbf{c}$ and any interpretation I satisfying $\exists xy(x \neq y)$, we have $I \models$ SM$[F; \mathbf{c}]$ iff $I \models$ SM$[UF_{\mathbf{c}}(F); \mathbf{c}]$.*

Proof. It is easy to check that the completion of $UF_{\mathbf{c}}(F)$ relative to $\mathbf{c}$ is equivalent to the completion of F relative to $\mathbf{c}$. By Theorem 4, we conclude that $\mathrm{SM}[UF_{\mathbf{c}}(F); \mathbf{c}]$ is equivalent to $\mathrm{SM}[F; \mathbf{c}]$. ∎

B.10 Proof of Theorem 10

For any formula F containing object constants f and g, we call it *fg-indistinguishable* if every occurrence of f and g in F is in a subformula of the form $(f = t) \wedge (g = t)$ that is fg-plain. For any interpretations I and J of F, we say I and J satisfy the relation $R(I, J)$ if

- $|I| = |J|$,
- $I(f) \neq I(g)$,
- $J(f) \neq J(g)$, and
- for all symbols c other than f and g, $I(c) = J(c)$.

Lemma 10 *If a formula F is fg-indistinguishable, then for any interpretations I and J such that $R(I, J)$, $F^I = F^J$.*

Proof. Notice that any fg-indistinguishable formula is built on atomic formulas not containing f and g, and formula of the form $(f = t) \wedge (g = t)$, using propositional connectives and quantifiers. The proof is by induction on such formulas.

Theorem 10 *For any set $\mathbf{c}$ of constants, there is no strongly equivalent transformation that turns an arbitrary sentence into a $\mathbf{c}$-plain sentence.*

Proof. The proof follows from the claim.

Claim: There is no f-plain formula that is strongly equivalent to $p(f) \wedge p(1) \wedge p(2) \wedge \neg p(3)$.

Let F be $p(f) \wedge p(1) \wedge p(2) \wedge \neg p(3)$. Then $F^*(g)$ is $p(f) \wedge p(g) \wedge p(1) \wedge p(2) \wedge \neg p(3)$. Let $I = \{p(1), p(2), f = 1, g = 2\}$ and $J = \{p(1), p(2), f = 1, g = 3\}$ (numbers are interpreted as themselves). It is easy to check that $I \models F^*(g)$ and $J \not\models F^*(g)$.

Assume for the sake of contradiction that there is a f-plain formula G that is strongly equivalent to F. Since G is f-plain, $G^*(g)$ is fg-indistinguishable. Since $R(I, J)$ holds, by Lemma 10, $I \models G^*(g)$ iff $J \models G^*(g)$, but this contradicts Theorem 3. ∎

B.11 Proof of Theorem 11

Theorem 11 *For any definite causal theory T, $I \models \mathrm{CM}[T;\mathbf{f}]$ iff $I \models \mathrm{SM}[Tr(T);\mathbf{f}]$.*

Proof. Assume that, without loss of generality, the rules (21)–(22) have no free variables. It is sufficient to prove that under the assumption that I satisfies T, for every rule (21), $J^{\mathbf{f}}_{\mathbf{g}} \cup I$ satisfies

$$B \rightarrow g(\mathbf{t}) = t_1$$

iff $J^{\mathbf{f}}_{\mathbf{g}} \cup I$ satisfies

$$(\neg\neg B)^*(\mathbf{g}) \rightarrow g(\mathbf{t}) = t_1 \wedge f(\mathbf{t}) = t_1.$$

Indeed, this is true since B is equivalent to $(\neg\neg B)^*(\mathbf{g})$ (Lemma 2), and I satisfies T. ∎

B.12 Proof of Theorem 12

Theorem 12 $I \models \mathrm{SM}[T;\mathbf{f}]$ *iff* $I \models \mathrm{IF}[T;\mathbf{f}]$.

Proof. We wish to show that $I \models T \wedge \neg\exists\widehat{\mathbf{f}}(\widehat{\mathbf{f}} < \mathbf{f} \wedge F^*(\widehat{\mathbf{f}}))$ iff $I \models T \wedge \neg\exists\widehat{\mathbf{f}}(\widehat{\mathbf{f}} \neq \mathbf{f} \wedge F^{\diamond}(\widehat{\mathbf{f}}))$. The first conjunctive terms are identical and if $I \not\models T$ then the claim holds.

Let us assume then, that $I \models T$. By definition, $\widehat{\mathbf{f}} < \mathbf{f}$ is equivalent to $\widehat{\mathbf{f}} \neq \mathbf{f}$. What remains to be shown is the correspondence between $F^*(\widehat{\mathbf{f}})$ and $F^{\diamond}(\widehat{\mathbf{f}})$.

Consider any list of functions $\mathbf{g}$ of the same length as $\mathbf{f}$. Let $\mathcal{I} = J^{\mathbf{f}}_{\mathbf{g}} \cup I$ be an interpretation of an extended signature $\sigma' = \sigma \cup \mathbf{g}$ where J is an interpretation of σ and J and I agree on functions not belonging to $\mathbf{f}$.

Consider any rule $f(\mathbf{t}) = t_1 \leftarrow \neg\neg B$ from T. The corresponding rule in $F^*(\mathbf{g})$ is equivalent to

$$f(\mathbf{t}) = t_1 \wedge g(\mathbf{t}) = t_1 \leftarrow B.$$

The corresponding rule in $F^{\diamond}(\mathbf{g})$ is equivalent to

$$g(\mathbf{t}) = t_1 \leftarrow B.$$

Now we consider cases

- $I \not\models B$. Clearly, both versions of the rule are vacuously satisfied by $\mathcal{I}$.

- $I \models B$. Then, since $I \models T$ it must be that $I \models f(\mathbf{t}) = t_1$ and so the corresponding rule in $F^*(\mathbf{g})$ is further equivalent to

$$g(\mathbf{t}) = t_1 \leftarrow B$$

which is equivalent to the corresponding rule in $F^\diamond(\mathbf{g})$ and so certainly $\mathcal{I}$ satisfies both corresponding rules or neither.

Thus, $\mathcal{I} \models F^*(\mathbf{g})$ iff $\mathcal{I} \models F^\diamond(\mathbf{g})$ and so the claim holds. ∎

B.13 Proof of Theorem 13

Lemma 11 *Given a formula F of many-sorted signature σ and an interpretation I of σ, $I \models gr_I[F]$ iff $I^{ns} \models gr_{I^{ns}}[F^{ns}]$.*

Proof. By induction on the structure of F. ∎

Lemma 12 *Given a formula F of many-sorted signature σ, interpretations I and J of σ and an interpretation K of σ^{ns} such that*

- *for every sort s in σ, $|I|^s = |J|^s = s^K$,*
- *for every predicate and function constant c and for every tuple $\boldsymbol{\xi}$ composed of elements from $|I^{ns}|$ such that $\xi_i \in |I|^{args_i}$ for every $\xi_i \in \boldsymbol{\xi}$, where $args_i$ is the i-th argument sort of c, we have $c(\boldsymbol{\xi})^K = c(\boldsymbol{\xi})^J$,*
- *for every predicate and function constant c and for every tuple $\boldsymbol{\xi}$ composed of elements from $|I^{ns}|$ such that $\xi_i \notin |I|^{args_i}$ for some $\xi_i \in |I|^{args_i}$, where $args_i$ is the i-th argument sort of c, we have $c(\boldsymbol{\xi})^K = c(\boldsymbol{\xi})^{I^{ns}}$,*

J is a model of $gr_I[F]^{\underline{I}}$ iff K is a model of $gr_{I^{ns}}[F^{ns}]^{\underline{I^{ns}}}$.

Proof. By induction on the structure of F. ∎

Lemma 13 *Given a formula F of many-sorted signature σ and two interpretations L and L_1 of σ^{ns} such that $R(L, L_1)$, if $L \models F^{ns} \wedge SF_\sigma$, then $L_1 \models F^{ns} \wedge SF_\sigma$.*

Proof. Assume that $L \models F^{ns} \wedge SF_\sigma$. We first show that $L_1 \models SF_\sigma$. Since $R(L, L_1)$, L and L_1 agree on all sort predicates s corresponding to sorts $s \in \sigma$. Thus, L_1 clearly satisfies the first two items of SF_σ. We now consider the third item of SF_σ. For tuples $\xi_1, \ldots, \xi_k$ such that each $\xi_i \in args_i$ where $args_i$ is the i-th argument sort of f, since $R(L, L_1)$, L and L_1 agree on $f(\xi_1, \ldots, \xi_k)$ so L_1 satisfies the implication. For all other tuples, the implication is vacuously satisfied. Finally, the fourth and fifth items of SF_σ are tautologies in classical logic so we conclude that $L_1 \models SF_\sigma$.

Next, $L_1 \models F^{ns}$ can be shown by induction on the structure of F^{ns}. ∎

Lemma 14 *Given a formula F of many-sorted signature σ, a set of function and predicate constants $\mathbf{c}$ from σ and two interpretations L and L_1 of σ^{ns} such that $R(L, L_1)$, if L is a stable model of $F^{ns} \wedge SF_\sigma$ w.r.t. $\mathbf{c}$, then L_1 is a stable model of $F^{ns} \wedge SF_\sigma$ w.r.t. $\mathbf{c}$.*

Proof. Omitted. The proof is long but not complicated. ∎

Theorem 13 *Let F be a formula of a many-sorted signature σ, and let $\mathbf{c}$ be a set of function and predicate constants.*

(a) If an interpretation I of signature σ is a model of $\mathrm{SM}[F; \mathbf{c}]$, then I^{ns} is a model of $\mathrm{SM}[F^{ns} \wedge SF_\sigma; \mathbf{c}]$.
(b) If an interpretation L of signature σ^{ns} is a model of $\mathrm{SM}[F^{ns} \wedge SF_\sigma; \mathbf{c}]$ then there is some interpretation I of signature σ such that I is a model of $\mathrm{SM}[F; \mathbf{c}]$ and $R(L, I^{ns})$.

Proof.

(a) Consider an interpretation I (of many-sorted signature σ) that is a stable model of F w.r.t. $\mathbf{c}$. This means that $I \models F$ and there is no interpretation J such that $J <^{\mathbf{c}} I$ and $J \models gr_I[F]^{\underline{I}}$. We wish to show that $I^{ns} \models F^{ns} \wedge SF_\sigma$ and there is no (unsorted) interpretation K such that $K <^{\mathbf{c}} I^{ns}$ and $K \models gr_{I^{ns}}[F^{ns} \wedge SF_\sigma]^{\underline{I^{ns}}}$. From Lemma 11, $I \models F$ iff $I^{ns} \models F^{ns}$. It follows from the definition of I^{ns} that $I^{ns} \models SF_\sigma$ so we conclude that $I \models F$ iff $I^{ns} \models F^{ns} \wedge SF_\sigma$. For the second item, we will prove the contrapositive: if there is an (unsorted) interpretation K such that $K <^{\mathbf{c}} I^{ns}$ and $K \models gr_{I^{ns}}[F^{ns} \wedge SF_\sigma]^{\underline{I^{ns}}}$, then there is a (many-sorted) interpretation J such that $J <^{\mathbf{c}} I$ and $J \models gr_I[F]^{\underline{I}}$.

Assume there is an interpretation K such that $K <^{\mathbf{c}} I^{ns}$ and $K \models gr_{I^{ns}}[F^{ns} \wedge SF_\sigma]^{\underline{I^{ns}}}$. We obtain the interpretation J as follows. For every sort s in σ, $|J|^s = |I|^s$. For every predicate and function constant c in σ and every tuple $\boldsymbol{\xi}$ such that each element $\xi_i \in |I|^{s_i}$ where s_i is the sort of the i-th argument of c, we let $c^J(\boldsymbol{\xi}) = c^K(\boldsymbol{\xi})$. For predicate constants, it is not hard to see that this is a valid assignment as atoms are either true or false regardless of considering many-sorted or unsorted logic.

We argue that this assignment is also valid for function constants. That is, K does not map a function f to a value outside of $|I|^s$ where s is the value sort of f. This follows from the fact that $I^{ns} \models SF_\sigma$ and in particular, the third item of SF_σ. Thus, since $K \models gr_{I^{ns}}[F^{ns} \wedge SF_\sigma]^{\underline{I^{ns}}}$, it follows that K too maps functions to elements of the appropriate sort.

We now show that $J <^{\mathbf{c}} I$. Since $K \models gr_{I^{ns}}[SF_\sigma]^{\underline{I^{ns}}}$, the fourth and fifth rules in SF_σ are choice formulas that force K to agree with I^{ns} on every predicate and function constant c for every tuple that has at least one element outside of the corresponding sort. For every predicate and function constant c and all tuples that have all elements in the appropriate sort, K and J agree. Further, since I and I^{ns} agree on these as well, it follows immediately since $K <^{\mathbf{c}} I^{ns}$, that $J <^{\mathbf{c}} I$.

To apply Lemma 12, we verify the conditions of the lemma. It is clear that the second condition is true. The first condition follows from the definition of $K <^{\mathbf{c}} I^{ns}$: since the sort predicates are not in $\mathbf{c}$, K and I^{ns} agree on these predicates. The third condition follows from the fact that since $K \models gr_{I^{ns}}[F^{ns} \wedge SF_\sigma]^{I^{ns}}_{\underline{}}$ it follows that $K \models gr_{I^{ns}}[SF_\sigma]^{I^{ns}}_{\underline{}}$; the fourth and fifth rules in SF_σ are choice formulas that force K to agree with I^{ns} for every tuple that has at least one element outside of the corresponding sort. Thus, by Lemma 12, since $K \models gr_{I^{ns}}[F^{ns} \wedge SF_\sigma]^{I^{ns}}_{\underline{}}$ and thus, $K \models gr_{I^{ns}}[F^{ns}]^{I^{ns}}_{\underline{}}$, it follows that $J \models gr_I[F]^{I}_{\underline{}}$.

(b) Given an interpretation L that is a stable model of $F^{ns} \wedge SF_\sigma$ w.r.t. $\mathbf{c}$, we first obtain the interpretation L_1 of σ^{ns} as follows.

- $|L_1| = |L|$;
- $\mathtt{s}^{L_1} = \mathtt{s}^L$ for every $\mathtt{s}$ corresponding to a sort s from σ;
- $c(\xi_1, \dots, \xi_k)^{L_1} = c(\xi_1, \dots, \xi_k)^L$ for every tuple $\xi_1, \dots, \xi_k$ such that $\xi_i \in s_i$ where s_i is the i-th argument sort of c;
- $c(\xi_1, \dots, \xi_k)^{L_1} = |L_1|_0$ for every tuple $\xi_1, \dots, \xi_k$ such that $\xi_i \notin s_i$ for some i where s_i is the i-th argument sort of c.

It is easy to see that $R(L, L_1)$. By Lemma 14, L_1 is a stable model of $F^{ns} \wedge SF_\sigma$ w.r.t. $\mathbf{c}$. We then obtain the interpretation I of signature σ as follows.

For every sort s in σ, $|I|^s = s^{L_1}$. For every predicate and function constant c in σ and every tuple $\boldsymbol{\xi}$ such that $\xi_i \in |L|^{s_i}$ where s_i is the sort of the i-th argument of c, we have $c(\boldsymbol{\xi})^I = c(\boldsymbol{\xi})^{L_1}$. For predicate constants, it is not hard to see that this is a valid assignment as atoms are either true or false regardless of considering many-sorted or unsorted logic.

We argue that this assignment is also valid for function constants. That is, I does not map a function f to a value outside of $|I|^s$ where s is the value sort of f. This follows from the fact that $L_1 \models SF_\sigma$ (by Lemma 13) and in particular, the third item of SF_σ. Thus, it follows that I too maps functions to elements of the appropriate sort.

Now it is clear that $L_1 = I^{ns}$ and so we have $R(L, I^{ns})$. We now show that I is a stable model of F.

We have an interpretation I (of many-sorted signature σ) such that I^{ns} is a stable model of $F^{ns} \wedge SF_\sigma$ w.r.t. $\mathbf{c}$. This means that $I^{ns} \models F^{ns} \wedge SF_\sigma$ and there is no interpretation K such that $K <^{\mathbf{c}} I^{ns}$ and $K \models gr_{I^{ns}}[F^{ns} \wedge SF_\sigma]^{I^{ns}}_{\underline{}}$. We wish to show that $I \models F$ and there is no interpretation J such that $J <^{\mathbf{c}} I$ and $J \models gr_I[F]^{\underline{I}}$. From Lemma 11, $I \models F$ iff $I^{ns} \models F^{ns}$ so we conclude that $I \models F$. For the second item, we will prove the contrapositive; if there is a (many-sorted) interpretation J such that $J <^{\mathbf{c}} I$ and $J \models gr_I[F]^{\underline{I}}$, then there is an (unsorted) interpretation K such that $K <^{\mathbf{c}} I^{ns}$ and $K \models gr_{I^{ns}}[F^{ns} \wedge SF_\sigma]^{\underline{I}^{ns}}$.

Assume there is an interpretation J such that $J <^{\mathbf{c}} I$ and $J \models gr_I[F]^{\underline{I}}$. We obtain the interpretation K be J^{ns}.

We now show that $K <^{\mathbf{c}} I^{ns}$. For every predicate and function constant c for every tuple that has at least one element outside of the corresponding sort, by definition of $K = J^{ns}$, $c^K = c^{I^{ns}} = |I^{ns}|_0$ if c is a function constant and $c^K = c^{I^{ns}} = \text{FALSE}$ if c is a predicate constant. That is, for every predicate and function constant c for every tuple that has at least one element outside of the corresponding sort, K and I^{ns} agree. For every predicate and function constant c and all tuples of elements in the appropriate sort, K and J agree. Further, since I and I^{ns} agree on these as well, $K <^{\mathbf{c}} I^{ns}$ follows immediately from $J <^{\mathbf{c}} I$.

To apply Lemma 12, we must verify the conditions of the lemma. It is clear that the second condition is true. The first condition follows from the definition of $K = J^{ns}$. The third condition follows from the observation above: by definition of $K = J^{ns}$, $c^K = c^{I^{ns}} = |I^{ns}|_0$ if c is a function constant and $c^K = c^{I^{ns}} = \text{FALSE}$ if c is a predicate constant. Thus, by Lemma 12, since $J \models gr_I[F]^{\underline{I}}$, it follows that $K \models gr_{I^{ns}}[F^{ns}]^{\underline{I}^{ns}}$.

Then, it is easy to see that by definition of I^{ns}, $I^{ns} \models SF_\sigma$. Then, by definition of $K = J^{ns}$, it is clear that $K \models SF_\sigma$. We will show that $K \models (SF_\sigma)^{\underline{I}^{ns}}$.

Since K and I^{ns} agree on all sort predicates, it is clear that K satisfies the formulas in the first two items of $(SF_\sigma)^{\underline{I}^{ns}}$.

Since K and I^{ns} agree on all function constants f for tuples $\xi_i, \ldots, \xi_k$ such that each ξ_i is in $|I|^{s_i}$ where s_i is the i-th argument sort of f, it is clear that K satisfies the third item of $(SF_\sigma)^{\underline{I}^{ns}}$.

The last two items of $(SF_\sigma)^{\underline{I}^{ns}}$ are only satisfied if K agrees with I^{ns} on all predicate (function) constants c and all tuples $\xi_1, \ldots, \xi_k$ such that some ξ_i is not in $|I|^{s_i}$ where s_i is the i-th argument sort of c. However, by definition of $K = J^{ns}$ and I^{ns}, both K and I^{ns} map this to $|I^n s|_0$ if c is a function constant or FALSE if c is a predicate constant so K satisfies these items. So we conclude that $K \models gr_{I^{ns}}[F^{ns} \wedge SF_\sigma]^{\underline{I}^{ns}}$. ∎

Lemma 15 *Let Π be a clingcon program with CSP (V, D, C), let $\mathcal{T}$ be the background theory conforming to (V, D, C), let $\mathbf{p}$ be the set of all propositional constants occurring in Π, let I be a $\mathcal{T}$-interpretation $\langle I^f, X\rangle$ and let J be an interpretation $\langle I^f, Y\rangle$ such that $Y \subset X$. If $I \models \Pi$, then $Y \models \Pi^X_{I^f}$ iff $J \models \Pi^{\underline{I}}$.*

Proof. Assume $I \models \Pi$.

($\Rightarrow$) Assume $Y \models \Pi^X_{I^f}$. This means that Y satisfies every rule in the reduct $\Pi^X_{I_f}$. For each rule r of the form (26) in Π, there are two cases:

- Case 1: $X \models B$ and $I^f \models \mathit{Cn}$. In this case, $r^X_{I_f}$ is

$$a \leftarrow B, \tag{B.8}$$

  and $r^{\underline{I}}$ is equivalent to

$$a^{\underline{I}} \leftarrow B^{\underline{I}} \tag{B.9}$$

  under the assumption $I \models \Pi$.
  - Subcase 1: $I \models B$. Since $I \models \Pi$, it must be that $I \models a$. Consequently, (B.9) is the same as (B.8), so it follows that $J \models r^{\underline{I}}$.
  - Subcase 2: $I \not\models B$. Since $B^{\underline{I}} = \bot$, clearly, $J \models r^{\underline{I}}$.
- Case 2: $X \not\models B$ or $I^f \not\models \mathit{Cn}$. Clearly, $r^{\underline{I}}$ is equivalent to $\top$, so $J \models r^{\underline{I}}$.

($\Leftarrow$) Assume $J \models \Pi^{\underline{I}}$. For each rule r of the form (26) in Π, there are two cases:

- Case 1: $I \not\models N \wedge \mathit{Cn}$. In this case, the reduct $r^X_{I_f}$ is empty. Clearly, $Y \models r^X_{I_f}$.
- Case 2: $I \models N \wedge \mathit{Cn}$. The reduct $r^X_{I_f}$ is $a \leftarrow B$.
  - Subcase 1: $I \models B$. $r^{\underline{I}}$ is equivalent to $a^{\underline{I}} \leftarrow (B \wedge N \wedge \mathit{Cn})^{\underline{I}}$. Since $J \models r^{\underline{I}}$, it must be that $a^{\underline{I}} = a$ and $J \models a$. Consequently, $Y \models a$, so $Y \models r^X_{I_f}$.
  - Subcase 2: $I \not\models B$ (i.e., $X \not\models B$). Since $Y \subset X$, we have $Y \not\models B$ so $Y \models r^X_{I_f}$.

∎

Theorem 14 *Let Π be a clingcon program with CSP (V, D, C), let $\mathbf{p}$ be the set of all propositional constants occurring in Π, let $\mathcal{T}$ be the background theory conforming to (V, D, C), and let $\langle I^f, X\rangle$ be a $\mathcal{T}$-interpretation. Set X is a constraint answer set of Π relative to I^f iff $\langle I^f, X\rangle$ is a $\mathcal{T}$-stable model of Π relative to $\mathbf{p}$.*

Proof.

$$X \text{ is a constraint answer set of } \Pi \text{ relative to } I^f$$

iff

$$X \text{ satisfies } \Pi^X_{I_f}, \text{ and no proper subset } Y \text{ of } X \text{ satisfies } \Pi^X_{I_f}$$

iff (by Lemma 15)

$\langle I^f, X\rangle$ is a $\mathcal{T}$-model of Π, and no interpretation J such that $J <^{\mathbf{p}} \langle I^f, X\rangle$ satisfies $\Pi^{\underline{I}}$

iff

$$\langle I^f, X\rangle \text{ is a } \mathcal{T}\text{-stable model of } \Pi \text{ relative to } \mathbf{p}.$$

∎

B.15 Proof of Theorem 15

Lemma 16 *For any ASP(LC) program Π, any LJN interpretation (X, T), and any $\mathcal{T}$-interpretation $I = \langle I^f, Y\rangle$, the following conditions are equivalent:*

- $I \models T \cup \overline{T}$;
- *For every theory atom t occurring in Π, it holds that $(X, T) \models t$ iff $I \models t$.*

Proof.

(i) Assume $I \models T \cup \overline{T}$. Take any theory atom t occurring in Π.
($\Rightarrow$) Assume $(X, T) \models t$. It is immediate that $t \in T$ and so by the assumption on I, we have $I \models t$.
($\Leftarrow$) Assume $I \models t$. Since $I \models T$, it follows that $t \in T$ and so $(X, T) \models t$.

(ii) Assume that, for every theory atom t occurring in Π, it holds that $(X, T) \models t$ iff $I \models t$. By definition of $(X, T) \models t$, for every t occurring in Π, it follows that $t \in T$ iff $I \models t$. Thus $I \models T$ and $I \models \overline{T}$ so $I \models T \cup \overline{T}$.

∎

Lemma 17 *Given an ASP(LC) program Π, two LJN-interpretations (X, T) and (Y, T) such that $(X, T) \models \Pi$ and $Y \subseteq X$, and two $\mathcal{T}$-interpretations $I = \langle I^f, X\rangle$ and $J = \langle I^f, Y\rangle$ such that $I \models \Pi$, and $I^f \models T \cup \overline{T}$, It holds that $Y \models \Pi^{(X,T)}$ iff $J \models \Pi^{\underline{I}}$.*

Proof. ($\Rightarrow$) Assume $Y \models \Pi^{(X,T)}$. This means that Y satisfies every rule in the reduct $\Pi^{(X,T)}$. For each rule r of the form (27) in Π, there are two cases:

- Case 1: $(X, T) \models N \wedge LC$.
  In this case, the corresponding rule in the reduct $\Pi^{(X,T)}$ is

$$a \leftarrow B.$$

  On the other hand, $r^{\underline{I}}$ has two cases:
  · Subcase 1: $I \models B$.
  Since we assume $I \models \Pi$, it must be that $I \models a$. By Lemma 16, since $(X, T) \models$

t for all t in LC, so too does I and so $I \models LC$. In this case, $r^{\underline{I}}$ is

$$a \leftarrow B, \top, \ldots, \top, LC^{\underline{I}}.$$

Since I and J interpret object constants in the same way and $I \models LC^{\underline{I}}$, we have $J \models LC^{\underline{I}}$. Thus by definition of J, it follows that $J \models B$ iff $Y \models B$ and $J \models a$ iff $Y \models a$, so the claim holds.

· Subcase 2: $I \not\models B$. The reduct $r^{\underline{I}}$ is either $a \leftarrow \bot$ or $\bot \leftarrow \bot$ and in either case, $J \models r^{\underline{I}}$.

- Case 2: $(X, T) \not\models N \wedge LC$.
  By the condition of I and by Lemma 16, $I \not\models N \wedge LC$ so $r^{\underline{I}}$ is $a \leftarrow \bot$ or $\bot \leftarrow \bot$ depending on whether $I \models a$. Thus, J trivially satisfies $r^{\underline{I}}$.

($\Leftarrow$) Assume $J \models \Pi^{\underline{I}}$. This means that J satisfies every rule in $\Pi^{\underline{I}}$. For any rule r of the form (27) in Π, there are two cases.

- Case 1: $I \not\models N \wedge LC$.
  By the condition of I and by Lemma 16, $(X, T) \not\models N \wedge LC$. Thus the reduct $\Pi^{(X,T)}$ does not contain a corresponding rule so there is nothing for Y to satisfy.
- Case 2: $I \models N \wedge LC$.
  By the condition of I and by Lemma 16, $(X, T) \models N \wedge LC$ so the reduct $r^{(X,T)}$ is $a \leftarrow B$.
  · Subcase 1: $I \not\models B$.
  By the condition of I, $X \not\models B$ and since $Y \subseteq X$, $Y \not\models B$. Thus, $Y \models r^{(X,T)}$.
  · Subcase 2: $I \models B$.
  Since $I \models \Pi$, it must be that $I \models a$ so the reduct $r^{\underline{I}}$ is $a \leftarrow B \wedge LC^{\underline{I}}$. Now since J and I agree on every object constant and since $I \models LC^{\underline{I}}$, we have $J \models LC^{\underline{I}}$. Thus, $J \models r^{\underline{I}}$ iff $J \models a \leftarrow B$. Since we assume $J \models \Pi^{I}$, we conclude $J \models a \leftarrow B$. Now by definition of J, it follows that $Y \models r^{(X,T)}$.

∎

Theorem 15 *Let Π be an ASP(LC) program of signature $\langle \sigma^p, \sigma^f \rangle$ where σ^p is a set of propositional constants, and let σ^f be a set of object constants, and let I^f be an interpretation of σ^f.*

(a) If (X, T) is an LJN-answer set of Π, then for any $\mathcal{T}$-interpretation I such that $I^f \models T \cup \overline{T}$, we have $\langle I^f, X \rangle \models \mathrm{SM}[\Pi; \sigma^p]$.
(b) For any $\mathcal{T}$-interpretation $I = \langle I^f, X \rangle$, if $\langle I^f, X \rangle \models \mathrm{SM}[\Pi; \sigma^p]$, then an LJN-interpretation (X, T) where

$$T = \{t \mid t \text{ is a theory atom in } \Pi \text{ such that } I^f \models t\}$$

is an LJN-answer set of Π.

Proof. In this proof, we refer to the reduct-based characterization of a stable model from [Bartholomew and Lee, 2013c].

(a) Assume (X, T) is an LJN-answer set of Π. Take any $\mathcal{T}$-interpretation $I = \langle I^f, X\rangle$ such that $I^f \models_{bg} T \cup \overline{T}$.

Now for any atom p, by the condition of I, we have $I \models p$ iff $(X, T) \models p$. Similarly, for any theory atom t occurring in Π, by the condition of I and by Lemma 16, $I \models t$ iff $(X, T) \models t$. Thus, since $(X, T) \models \Pi$, $I \models \Pi$.

We must now show that there is no interpretation J such that $J <^{\sigma_p} I$ and $J \models \Pi^{\underline{I}}$. Take any $J <^{\sigma_p} I$. That is, $J = \langle I^f, Y\rangle$ such that $Y \subset X$. By Lemma 17, $J \models \Pi^{\underline{I}}$ iff $Y \models \Pi^{(X,T)}$ but since (X, T) is an LJN-answer set of Π, $Y \not\models \Pi^{(X,T)}$ and thus $J \not\models \Pi^{\underline{I}}$ so I is a stable model of Π.

(b) Assume $I = \langle I^f, X\rangle$ is a stable model of Π.

Now for any atom p, by definition of (X, T), $(X, T) \models p$ iff $I \models p$. Similarly, for any theory atom t occurring in Π, by the condition of I and Lemma 16, $(X, T) \models t$ iff $I \models t$. Thus, since $I \models \Pi$, $(X, T) \models \Pi$.

We must now show that there is no set of atoms Y such that $Y \subset X$ and $Y \models \Pi^{(X,T)}$. Take any $Y \subset X$. By Lemma 17, $Y \models \Pi^{(X,T)}$ iff $J \models \Pi^{\underline{I}}$ where $J = \langle I^f, Y\rangle$. Since $J <^{\sigma^p} I$ and I is a stable model of Π, $J \not\models \Pi^{\underline{I}}$. Thus $Y \not\models \Pi^{(X,T)}$ and so (X, T) is an LJN-answer set of Π. ∎

B.16 Proof of Theorem 16

The proof of the theorem is rather obvious once we view the type declarations of LW-program as a special case of the many-sorted signature declarations. So we omit the proof here.