Connecting First-Order ASP and the Logic FO(ID) Through Reducts

Miroslaw Truszczynski

Department of Computer Science, University of Kentucky, Lexington, KY 40506-0633, USA

In honor of Vladimir Lifschitz
on his 65th birthday!

Abstract. Recently, an answer-set programming (ASP) formalism of logic programing with the answer-set semantics has been extended to the full first-order setting. Earlier an extension of first-order logic with inductive definitions, the logic FO(ID), was proposed as a knowledge representation formalism and developed as an alternative ASP language. We present characterizations of these formalisms in terms of concepts of infinitary propositional logic. We use them to find a direct connection between the first-order ASP and the logic FO(ID) under some restrictions on the form of theories (programs) considered.

1 Introduction

Answer-set programming (*ASP*, for short) is a paradigm for modeling and solving search problems and their optimization variants. To model a search problem, one constructs a theory in some logic language so that once the theory is appended with an encoding of a problem instance, its models represent in some direct fashion solutions to that instance of the problem. The paradigm was first formulated in these terms by Marek and Truszczyński [14] and Niemelä [16] for the formalism of logic programming with the stable-model semantics by Gelfond and Lifschitz [10, 11], a nonmonotonic knowledge representation formalism with informal roots in epistemic interpretations of the negation as failure and closely connected to the default logic by Reiter [20]. Logic programming with the stable-model semantics and its extensions with the semantics of *answer sets* [11] are to this day the most broadly used dialects of ASP.

These formalisms were originally introduced with the syntax limited to *rules* and the semantics restricted to Herbrand interpretations. Recently researchers have sought to address these limitations. In particular, Pearce and Valverde [19] introduced the quantified equilibrium logic. That logic is a first-order extension of the equilibrium logic by Pearce [18], a propositional logic with the semantics of equilibrium models. For a class of theories that correspond to programs, equilibrium models and answer sets coincide and so, the quantified equilibrium logic is indeed a generalization of logic programming with the answer-set semantics to the full first-order setting. Ferraris, Lee and Lifschitz [8], proposed an alternative approach within the language of second-order logic. They introduced an operator SM that assigns a second-order sentence to a first-order one (incidentally, the operator SM closely resembles the one used to define circumscription). For a first-order sentence F, Ferraris et al. [8] proposed models of $SM[F]$ as *answer*

sets of F and showed that the concept generalizes that of an answer set of a propositional formula. They also proved that the semantics of sentences provided by answer sets coincides with that of the quantified equilibrium logic.

While papers on ASP often bring up the default negation operator in logic programming as the key element to its success as a modeling language, the fact is that in most applications the default negation *not* is used as if it were a classical one. Arguably, the main appeal of logic programming with the answer-set semantics as an ASP formalism comes from its capability to capture inductive definitions. The importance of inductive definitions for knowledge representation was argued by Denecker [1, 2], who proposed an extension of the first-order logic with inductive definitions, the logic FO(ID), as a knowledge representation formalism. A version of that language can be used according to the ASP paradigm and there are fast tools to support that use of the logic FO(ID).[1] The semantics of definitions in the logic FO(ID) is given by the well-founded semantics extended to the case of arbitrary interpretations. As the well-founded and answer-set semantics are related, the question arises about the relation between the first-order extensions of ASP that we discussed earlier and the logic FO(ID).

In this work we show a class of FO(ID) theories for which the relationship is quite direct. Under a simple translation of FO(ID) theories into programs, models of the FO(ID) theories in that class are precisely answer sets of the program resulting from the translation. In this way, we resolve in positive a conjecture by Vladimir Lifschitz[2]. We point out that a converse embedding is possible as well.

As the main technical device in our arguments we use characterizations of the first-order ASP and FO(ID) in the infinitary propositional logic. They are based on extensions to the infinitary case of the notion of reduct [7] on the one hand, and of the algebraic approach to stable and well-founded semantics [4] of general programs on the other. For the class of theories that we call programs, the concept of the reduct can be extended in two ways. We use one of them as a bridge to the first-order ASP as developed by Pearce and Valverde [19] and Ferraris et al. [8]. We use the other as the connection to the logic FO(ID), and note here in passing that it also provides an alternative characterization of a version of the first-order ASP proposed by Denecker et al. [3].

2 Stable models in infinitary propositional logic

Let A be a *propositional signature*, that is, a set of 0-ary relation symbols (or propositions). We assume the existence of a 0-ary predicate constant $\bot$, different from all symbols in A, and define sets $\mathcal{F}_0, \mathcal{F}_1, \ldots$ by induction as follows:

1. $\mathcal{F}_0 = A \cup \{\bot\}$
2. $\mathcal{F}_{i+1}$, where $i \geq 0$, 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 expressions $F \to G$, where $F, G \in \mathcal{F}_0 \cup \ldots \cup \mathcal{F}_i$.

We define $\mathcal{L}_A^{inf} = \bigcup_{i=0}^{\infty} \mathcal{F}_i$. We call elements of $\mathcal{L}_A^{inf}$ *infinitary formulas* (over A). Each formula $F \in \mathcal{L}_A^{inf}$ belongs to at least one set $\mathcal{F}_i$. We call the least index i of a set $\mathcal{F}_i$ containing F the *rank* of F.

[1] The IDP system, `http://dtai.cs.kuleuven.be/krr/software/idp3`.

[2] The conjecture was communicated to me by Yulia Lierler.

The primary connectives of the language are $\{\}^\vee$, $\{\}^\wedge$ and $\rightarrow$. We define the boolean connectives $\wedge$, $\vee$, $\leftrightarrow$ and $\neg$, as well as another 0-ary predicate constant $\top$, as shorthands: $F \vee G ::= \{F, G\}^\vee$; $F \wedge G ::= \{F, G\}^\wedge$; $F \leftrightarrow G ::= (F \rightarrow G) \wedge (G \rightarrow F)$; $\neg F ::= F \rightarrow \bot$; and $\top ::= \neg\bot$.

The standard semantics of propositional logic extends to the case of infinitary formulas. An *interpretation* of A is a subset of A (atoms in the subset are *true* and all the other ones are *false*). We denote the set of all interpretations of A by Int_A. For an interpretation I of A, and an infinitary formula F, we define the relation $I \models F$ by induction on the rank of a formula as follows:

1. $I \not\models \bot$
2. For every $p \in A$, $I \models p$ if $p \in I$
3. $I \models \mathcal{H}^\vee$ if there is a formula $F \in \mathcal{H}$ such that $I \models F$
4. $I \models \mathcal{H}^\wedge$ if for every formula $F \in \mathcal{H}$, $I \models F$
5. $I \models F \rightarrow G$ if $I \not\models F$ or $I \models G$.

The concept of the *reduct* proposed by Ferraris [7] for the standard propositional logic can be extended to sets of infinitary formulas. The inductive definition follows:

1. $\bot^I = \bot$
2. For $p \in A$, $p^I = \bot$ if $I \not\models p$; otherwise $p^I = p$
3. $(\mathcal{H}^\wedge)^I = \bot$ if $I \not\models \mathcal{H}^\wedge$; otherwise, $(\mathcal{H}^\wedge)^I = \{G^I \mid G \in \mathcal{H}\}^\wedge$
4. $(\mathcal{H}^\vee)^I = \bot$ if $I \not\models \mathcal{H}^\vee$; otherwise, $(\mathcal{H}^\vee)^I = \{G^I \mid G \in \mathcal{H}\}^\vee$
5. $(G \rightarrow H)^I = \bot$ if $I \not\models G \rightarrow H$; otherwise $(G \rightarrow H)^I = G^I \rightarrow H^I$.

If $\mathcal{F}$ is a set of infinitary formulas, we define the reduct $\mathcal{F}^I$ by $\mathcal{F}^I = \{F^I \mid F \in \mathcal{F}\}$.

Definition 1. *Let $\mathcal{F} \subseteq \mathcal{L}_A^{inf}$ be a set of infinitary formulas. An interpretation $I \in Int_A$ is a* stable model *of $\mathcal{F}$ if I is a minimal model of $\mathcal{F}^I$.*

The use of the term *model* in *stable model* is justified. That is, stable models are models. This is evident from the following more general property.[3]

Proposition 1. *For every set $\mathcal{F} \subseteq \mathcal{L}_A^{inf}$ and every interpretation $I \in Int_A$, $I \models \mathcal{F}$ if and only if $I \models \mathcal{F}^I$.*

If a set $\mathcal{F}$ of finitary formulas has no negative occurrences of variables, it has minimal models and each of these models is a stable model of the formula. This property does not hold in the case of infinitary formulas. Let $F_i = \{p_i, p_{i+1}, \ldots\}^\vee$, $i = 0, 1, \ldots$, and let $\mathcal{F} = \{F_0, F_1, \ldots\}$ (we assume here that $A = \{p_0, p_1, \ldots\}$). It is easy to see that there is no finite set $M \subseteq A$ such that $M \models \mathcal{F}$. On the other hand, every infinite set $M \subseteq A$ is a model of $\mathcal{F}$. It follows that $\mathcal{F}$ has no minimal models. It is also easy to show that $\mathcal{F}$ has no stable models.

Next, we extend the semantics of HT-interpretations to (sets of) infinitary formulas. Our goal is to extend the equilibrium logic [18] to the infinitary case and show that equilibrium models and stable models coincide.

[3] The proofs of this result and the next two closely follow those of the corresponding results in the finitary case [7] and we omit them.

Definition 2. *An* HT-interpretation *is a pair $\langle X, Y\rangle$, where $X, Y \in Int_A$ and $X \subseteq Y$. The satisfiability relation $\models_{ht}$ is specified by induction as follows:*

1. $\langle X, Y\rangle \not\models_{ht} \bot$
2. *For $p \in A$, $\langle X, Y\rangle \models_{ht} p$ if $p \in X$*
3. $\langle X, Y\rangle \models_{ht} \mathcal{H}^{\vee}$ *if there is $G \in \mathcal{H}$ such that $\langle X, Y\rangle \models_{ht} G$*
4. $\langle X, Y\rangle \models_{ht} \mathcal{H}^{\wedge}$ *if for every $G \in \mathcal{H}$, $\langle X, Y\rangle \models_{ht} G$*
5. $\langle X, Y\rangle \models_{ht} G \to H$ *if $Y \models G \to H$; and $\langle X, Y\rangle \not\models_{ht} G$ or $\langle X, Y\rangle \models_{ht} H$.*

If $\langle X, Y\rangle \models_{ht} F$, then we say that $\langle X, Y\rangle$ is an HT-model *of F. The concept of a model and the relation $\models_{ht}$ extend in a standard way to* sets *of infinitary formulas.*

The following result gathers important properties of the relation $\models_{ht}$. They extend the corresponding properties of that relation in the standard finitary setting [7, 9].

Theorem 1. *For every formula F and every interpretations $X, Y \in Int_A$ such that $X \subseteq Y$ we have:*

1. $\langle X, Y\rangle \models_{ht} F$ *implies* $Y \models F$
2. $\langle X, Y\rangle \models_{ht} \neg F$ *if and only if* $Y \not\models F$
3. $\langle Y, Y\rangle \models_{ht} F$ *if and only if* $Y \models F$.

The next result characterizes the relation $\models_{ht}$ in terms of the standard satisfiability relation and the reduct.

Theorem 2. *For every set $\mathcal{F} \subseteq \mathcal{L}_A^{inf}$ of infinitary formulas and for every interpretations $X, Y \in Int_A$ such that $X \subseteq Y$, $\langle X, Y\rangle \models_{ht} \mathcal{F}$ if and only if $X \models \mathcal{F}^Y$.*

Let $\mathcal{F} \subseteq \mathcal{L}_A^{inf}$ be a set of infinitary formulas. Directly extending the definition from the finitary case, we say that an HT-interpretation $\langle Y, Y\rangle$ is an *equilibrium* model of $\mathcal{F}$ if $\langle Y, Y\rangle \models_{ht} \mathcal{F}$ and there is no proper subset X of Y such that $\langle X, Y\rangle \models_{ht} \mathcal{F}$. The following result connects stable and equilibrium models and follows directly from Theorem 1(3) and Theorem 2.

Theorem 3. *An interpretation $Y \in Int_A$ is a stable model of a set $\mathcal{F}$ of infinitary formulas from $\mathcal{L}_A^{inf}$ if and only if $\langle Y, Y\rangle$ is an equilibrium model of $\mathcal{F}$.*

3 Stable models in first-order logic

Let σ be a signature of a language of first-order logic and let U be a set. By σ^U we denote the signature obtained by adding to σ distinct symbols u^* (*names*) for every $u \in U$.

Next, let I be an interpretation of σ, that is, a *structure* comprising a non-empty *domain*, written as $|I|$, and for each relation and function symbol in σ, a relation or function on $|I|$, of the same arity, to interpret it. We denote by $A_{\sigma,I}$ the set of all those atomic formulas of the first order language $\mathcal{L}_{\sigma^{|I|}}$ that are built of relation symbols in σ and the names of elements in $|I|$. We identify an interpretation I of σ with its extension I' to $\sigma^{|I|}$ defined by setting $I'(s) = I(s)$, for every $s \in \sigma$, and $I'(u^*) = u$ for every

$u \in |I|$. From now on, we use the same symbol for an interpretation I of σ and for its (unique) extension to $\sigma^{|I|}$ defined above. We represent an interpretation I of σ (and its extension to $\sigma^{|I|}$) by a pair $\langle I^f, I^r\rangle$, where I^f is an interpretation of the part of σ (or equivalently, of $\sigma^{|I|}$) that consists of constant and function symbols in σ, and I^r is a subset of $A_{\sigma,I}$ that describes in the obvious way the relations in I (the interpretation of new constants is determined and does not need to be explicitly represented). We write Int_σ for the set of all interpretations of σ (extended as described above).

Let F be a sentence in the language $\mathcal{L}_{\sigma^{|I|}}$. We define the relation $I \models F$ following the approach used, for instance, by Doets [6] (the definition of the value t^I of a term t in I is standard, we assume the reader is familiar with it):

1. $I \not\models \bot$
2. $I \models p(t_1, \ldots, t_k)$ if $p((t_1^I)^*, \ldots, (t_k^I)^*) \in I^r$
3. $I \models t_1 = t_2$ if $t_1^I = t_2^I$
4. $I \models F \vee G$ if $I \models F$ or $I \models G$ (the case of $\wedge$ is analogous)
5. $I \models F \to G$ if $I \not\models F$ or $I \models G$
6. $I \models \exists x F(x)$ if for some $u \in |I|$, $I \models F(u^*)$
7. $I \models \forall x F(x)$ if for every $u \in |I|$, $I \models F(u^*)$.

Let I be an interpretation of σ and F be a sentence in $\mathcal{L}_{\sigma^{|I|}}$. We define the *grounding of F with respect to I*, $gr_I(F)$, as follows:

1. $gr_I(\bot) = \bot$
2. $gr_I(p(t_1, \ldots, t_k)) = p((t_1^I)^*, \ldots, (t_k^I)^*)$
3. $gr_I(t_1 = t_2)) = \top$, if $t_1^I = t_2^I$, and $\bot$, otherwise
4. If $F = G \vee H$, $gr_I(F) = gr_I(G) \vee gr_I(H)$ (the case of $\wedge$ is analogous)
5. If $F = G \to H$, $gr_I(F) = gr_I(G) \to gr_I(H)$
6. If $F = \exists x G(x)$, $gr_I(F) = \{gr_I(G(u^*)) \mid u \in |I|\}^\vee$
7. If $F = \forall x G(x)$, $gr_I(F) = \{gr_I(G(u^*)) \mid u \in |I|\}^\wedge$.

In addition, if $\mathcal{F}$ is a set of sentences from the language $\mathcal{L}_{\sigma^{|I|}}$, we define $gr_I(\mathcal{F}) = \{gr_I(F) \mid F \in \mathcal{F}\}$. It is clear that for every sentence F in $\mathcal{L}_{\sigma^{|I|}}$, $gr_I(F)$ is an infinitary propositional formula in the signature $A_{\sigma,I}$. It is important to note that $gr_I(F)$ depends only on I^f and not on I^r.

There is a simple connection between the first-order satisfiability relation and the one we defined earlier for the infinitary propositional logic.

Proposition 2. *Let I be an interpretation of σ and F a sentence from $\mathcal{L}_{\sigma^{|I|}}$. Then $I \models F$ if and only if $I^r \models gr_I(F)$.*

Proof. The basis of the induction is evident. For example, let us assume that F has the form $t_1 = t_2$. By the definition, if $I \models t_1 = t_2$ then $t_1^I = t_2^I$. It follows that $gr_I(F) = \top$ and $I^r \models gr_I(F)$. Conversely, if $I^r \models gr_I(F)$, then $gr_I(F) \neq \bot$. Thus, $t_1^I = t_2^I$ and $I \models t_1 = t_2$.

The inductive step is simple, too. For example, let us assume that $F = \forall x G(x)$. By the definition, $I \models F$ if and only if $I \models G(u^*)$, for every $u \in |I|$. By the induction hypothesis (as F is a sentence, each $G(u^*)$ is a sentence, too), this condition is equivalent to $I^r \models gr_I(G(u^*))$, for all $u \in |I|$ which, in turn, is equivalent to $I^r \models \{gr_I(G(u^*)) \mid u \in |I|\}^\wedge$. Noting that $\{gr_I(G(u^*)) \mid u \in |I|\}^\wedge = gr_I(F)$ completes the argument. The other cases for F can be handled in a similar way. $\square$

With the notion of grounding in hand, we now define stable models of a set of first-order sentences.

Definition 3. *Let σ be a signature. An interpretation $I = \langle I^f, I^r \rangle$ of σ is a* stable model *of a set $\mathcal{F}$ of sentences from $\mathcal{L}_\sigma$ if I^r is a stable model of $gr_I(\mathcal{F})$.*

This concept of stability is well defined as formulas in $gr_I(\mathcal{F})$ are infinitary propositional formulas over the signature $A_{\sigma,I}$ and for every interpretation I of σ, $I^r \subseteq A_{\sigma,I}$.

By the definition of a stable model of a set of infinitary formulas over $A_{\sigma,I}$ we have the following direct characterization of stable models.

Proposition 3. *Let σ be a signature and let $\mathcal{F}$ be a set of sentences from $\mathcal{L}_\sigma$. An interpretation I of σ is a stable model of $\mathcal{F}$ if and only if I^r is a minimal model of the reduct $[gr_I(\mathcal{F})]^{I^r}$.*

Ferraris et al. [8] introduced an operator SM that assigns to each first-order sentence F (in signature σ) a second-order formula $SM[F]$. The details of the definition are immaterial to our subsequent discussion and we omit them. Ferraris et al. defined an interpretation $I \in Int_\sigma$ to be a *stable model* of F if I is a model of $SM[F]$. We show that our definition of stable models of a sentence is equivalent to the one based on the operator SM. We proceed in a roundabout way through a connection between our concept of stability and that based on the quantified logic *here-and-there* [19].

Following Ferraris et al. [8], we define an *HT-interpretation* of a first-order signature σ as a triple $I = \langle I^f, I^h, I^t \rangle$, where I^f is an interpretation of the part of σ that consists of constant and function symbols, and I^h and I^t are subsets of $A_{\sigma,I}$ such that $I^h \subseteq I^t$. Moreover, we define the relation $I \models_{ht} F$, where F is a sentence from $\mathcal{L}_{\sigma^{|I|}}$, follows:

1. $I \not\models_{ht} \bot$
2. $I \models_{ht} p(t_1, \ldots, t_k)$ if $p((t_1^I)^*, \ldots, p(t_k^I)^*) \in I^h$
3. $I \models_{ht} t_1 = t_2$ if $t_1^I = t_2^I$
4. $I \models_{ht} G \vee H$ if $I \models G$ or $I \models H$ (the case of $\wedge$ is analogous)
5. $I \models_{ht} G \to H$ if $\langle I^f, I^t \rangle \models G \to H$; and $I \not\models_{ht} G$ or $I \models_{ht} H$.
6. $I \models_{ht} \forall x G(x)$ if for every $u \in |I|$, $I \models_{ht} G(u^*)$
7. $I \models_{ht} \exists x G(x)$ if for some $u \in |I|$, $I \models_{ht} G(u^*)$.

An HT-interpretation $I = \langle I^f, I^h, I^t \rangle$ is an *equilibrium* model of a sentence F [19] if $I \models_{ht} F$, $I^h = I^t$, and for every *proper* subset X of I^h, $\langle I^f, X, I^t \rangle \not\models_{ht} F$. Ferraris et al. [8] proved the following result.

Theorem 4 (Ferraris et al. [8]). *An HT-interpretation $I = \langle I^f, I^h, I^h \rangle$ is an equilibrium model of F if and only if $\langle I^f, I^h \rangle$ is a stable model of F (a model of $SM[F]$).*

The key step in showing that these two notions of stability coincide with the one we introduced in Definition 3 consists of showing that the first-order relation $\models_{ht}$ can be expressed in terms of the relation $\models_{ht}$ of the infinitary propositional logic.

Proposition 4. *Let I be an HT-interpretation of σ and F a sentence from $\mathcal{L}_{\sigma^{|I|}}$. Then, $I \models_{ht} F$ if and only if $\langle I^h, I^t \rangle \models_{ht} gr_I(F)$.*

Proof. The basis of the induction is evident. For example, let us assume that $F = p(t_1, \ldots, t_k)$. Then $I \models_{ht} F$ if and only if $p((t_1^I)^*, \ldots, (t_k^I)^*) \in I^h$. Since $gr_I(F) = p((t_1^I)^*, \ldots, (t_k^I)^*)$, that condition is equivalent to $\langle I^h, I^t\rangle \models_{ht} gr_I(F)$.

The induction step is also simple. For instance, let $F = G \rightarrow H$ and let us assume that $I \models_{ht} G \rightarrow H$. It follows that $\langle I^f, I^t\rangle \models G \rightarrow H$ and, also, that $I \not\models_{ht} G$ or $I \models_{ht} H$. By Proposition 2 and by the induction hypothesis, we obtain $I^t \models gr_I(G \rightarrow H)$, and $\langle I^h, I^t\rangle \not\models_{ht} gr_I(G)$ or $\langle I^h, I^t\rangle \models_{ht} gr_I(H)$. Since $gr_I(G \rightarrow H) = gr_I(G) \rightarrow gr_I(H)$, it follows that $\langle I^h, I^t\rangle \models_{ht} gr_I(G) \rightarrow gr_I(H)$ and, consequently, that $\langle I^h, I^t\rangle \models_{ht} gr_I(G \rightarrow H)$. The converse implication and other cases for F can be reasoned in a similar way. □

We now state and prove the result showing the equivalence of the three definitions of stable models of a first-order sentence.

Theorem 5. *The following definitions of a stable model of a sentence F are equivalent:*

1. *the definition in terms of the operator SM*
2. *the definition in terms of equilibrium models*
3. *the definition in terms of ground programs (Definition 3).*

Proof. (1) and (2) are equivalent by Theorem 4. We will prove the equivalence of (2) and (3). Let I be an interpretation of σ such that $\langle I^f, I^r, I^r\rangle$ is an equilibrium model of F. It follows that $\langle I^f, I^r, I^r\rangle \models_{ht} F$ and for every proper subset X of I^r, $\langle I^f, X, I^r\rangle \not\models_{ht} F$. The first property and Proposition 4 imply that $\langle I^r, I^r\rangle \models_{ht} gr_I(F)$. The second property and Proposition 4 imply that for every proper subset X of I^r, $\langle X, I^r\rangle \not\models_{ht} gr_I(F)$. Thus, by Theorem 3, I is a stable model of $gr_I(F)$ and, consequently, a stable model of F according to Definition 3.

Conversely, let us assume that I is a classical interpretation of σ such that I^r is a stable model of $gr_I(F)$. It follows (Theorem 3) that $\langle I^r, I^r\rangle \models_{ht} gr_I(F)$ and there is no proper subset X of I^r such that $\langle X, I^r\rangle \models_{ht} gr_I(F)$. Using Proposition 4, we obtain that I is an equilibrium model of F. □

We note that the definitions of stable models of sentences given here and in terms of equilibrium models extend to infinite collections of sentences. The definition in terms of the operator SM does not lend itself in any obvious way to such an extension.

4 Programs

A *program* (in the language of the infinitary propositional logic) is a set of *program clauses*, that is, formulas $F \rightarrow p$, where $F \in \mathcal{L}_A^{inf}$ and $p \in A$. For consistency with the standard logic programming notation, we write a clause $F \rightarrow p$ as $p \leftarrow F$. We call p the *head* of the clause $p \leftarrow F$.

Let Π be a program in the signature A. We denote by A_Π^{out} the set of all atoms that appear in the heads of clauses in Π and define $A_\Pi^{in} = A \setminus A_\Pi^{out}$. If the program Π is clear from the context, we drop "Π" from the notation. We call atoms in A_Π^{in} and A_Π^{out} *input* and *output* atoms, respectively.[4]

[4] One can consider a slightly more general setting in which we define A_Π^{out} as a subset of A that contains the heads of all rules in Π (but, possibly, also some other atoms). That setting

For programs we can generalize the concept of a stable model by taking into account a given interpretation of its input atoms.

Definition 4. *Let Π be a program in a signature A. An interpretation I of A is an* input stable model *of Π if I is a stable model of $\Pi \cup (I \cap A^{in})$.*

The concept of an input stable model was introduced and studied by Lierler and Truszczynski [13] as the basis for the formalism SM(ASP). It is closely related to models of modular logic programming systems [12, 17, 3].

Input stable models have an elegant direct characterization in terms of stable models that extends the corresponding characterization for the finitary case given by Lierler and Truszczynski [13]. Let Π be a program. We define $\Pi^{in} = \Pi \cup \{a \leftarrow \neg\neg a \mid a \in A^{in}\}$. The proof of the characterization below is similar to the one for the finitary case and we omit it.

Proposition 5. *Let Π be a program in a signature A. An interpretation I of σ is an input stable model of Π if and only if I is a stable model of Π^{in}.*

For programs one can introduce an alternative notion of a reduct. Let $F \in \mathcal{L}_A^{inf}$ and $I \in Int_A$. We define F_I to be the formula obtained by replacing each atom p that occurs negatively in F with $\bot$, if $I \not\models p$, and with $\top$, otherwise.[5] For a program Π, we define $\Pi_I = \{p \leftarrow F_I \mid p \leftarrow F \in \Pi\}$.

We note that formulas F_I have only positive occurrences of atoms. Thus, for any two interpretations $J, J' \in Int_A$ such that $J \subseteq J'$, if $J \models F_I$ then $J' \models F_I$. In particular, it follows that for every program Π and every interpretation I, Π_I has a least model denoted by $LM(\Pi_I)$. This observation gives rise to the notion of an *ID-stable* model of a program. We introduce it below, together with the related notion of an *input* ID-stable model.

Definition 5. *Let Π be a program in a signature A. An interpretation I of A is an* ID-stable model *of a program Π if $I = LM(\Pi_I)$. An interpretation I of A is an* input ID-stable model *of Π if I is an ID-stable model of $\Pi \cup (I \cap A^{in})$.*

It is clear that $I \models \Pi$ if and only if $I \models \Pi_I$. Thus, ID-stable models of Π are models of Π and the use of the term "model" in "ID-stable model" (and so, also in "input ID-stable model") is justified.

In general, stable models and ID-stable models of programs do not coincide. For instance, let $\Pi = \{p \leftarrow \neg\neg p\}$. One can check that $I = \emptyset$ and $J = \{p\}$ are stable models of Π. On the other hand, since p occurs positively in Π, $\Pi_I = \Pi_J = \Pi$. The least model of Π is $\emptyset$. Thus, I is an ID-stable model of Π but J is not!

We will now show a class of programs for which the two concepts coincide. Let $\mathcal{N}$ be the set of all formulas F that satisfy the following property: every occurrence of the

---

reduces in a simple way to the one we consider. One just needs to extend Π with rules of the form $a \leftarrow \bot$, for every $a \in A_\Pi^{out}$ that is not the head of any rule in Π.

[5] The only connectives in the language are $\{\cdot\}^\vee$ (generalized $\vee$), $\{\cdot\}^\wedge$ (generalized $\wedge$), and $\rightarrow$. Thus, the notions of a positive and negative occurrence of a propositional atom in a formula are well defined.

implication in F has $\bot$ as the consequent and no occurrence of implication in its antecedent. The first requirement says that all occurrences of the implication operator can be replaced with the negation operator, the second requirement says that the negation operator cannot be nested.

Lemma 1. *Let Π be a program in signature A and I an interpretation of A such that $I \models \Pi$. Then $\Pi^I \equiv \{p \leftarrow F^I \mid p \leftarrow F \in \Pi\}$.*

Proof. Let $p \leftarrow F \in \Pi$. Since $I \models p \leftarrow F$,

$$(p \leftarrow F)^I = \begin{cases} p^I \leftarrow \bot & \text{if } I \not\models F \\ p \leftarrow F^I & \text{if } I \models F \text{ and } I \models p \end{cases}$$

while

$$p \leftarrow F^I = \begin{cases} p \leftarrow \bot & \text{if } I \not\models F \\ p \leftarrow F^I & \text{if } I \models F \text{ and } I \models p. \end{cases}$$

Thus the assertion follows. □

Lemma 2. *For every formula $F \in \mathcal{N}$, and every interpretation I, F^I and F_I have the same models contained in I.*

Proof. The proof is by induction on the rank of a formula. If $F = \bot$, then $F^I = \bot = F_I$ and the claim is evident. If $F = p$, and $I \models p$, then $p^I = p$ and $p_I = p$. If $I \not\models p$, then $p^I = \bot$ and $p_I = p$. In each case, the formulas p^I and p_I have the same models that are subsets of I.

For the inductive step there are several cases to consider. First, let us assume that $F = \mathcal{H}^\vee$. If $I \not\models F$, $F^I = \bot$ and it has no models contained in I. Moreover, for every $G \in \mathcal{H}$, $I \not\models G$. By the induction hypothesis, the formulas G^I and G_I have the same models contained in I. Since $G^I = \bot$, no G_I has models contained in I and, consequently, $F_I = \{G_I \mid G \in \mathcal{H}\}^\vee$ has no models contained in I. Thus, let us assume that $I \models F$. It follows that $(\mathcal{H}^\vee)^I = \{G^I \mid G \in \mathcal{H}\}^\vee$ and $(\mathcal{H}^\vee)_I = \{G_I \mid G \in \mathcal{H}\}^\vee$. By the induction hypothesis, for every $G \in H$, G^I and G_I have the same models contained in I. Thus, $(\mathcal{H}^\vee)^I$ and $(\mathcal{H}^\vee)_I$ have that property, too.

The argument for the case $F = \mathcal{H}^\wedge$ is similar. So, let us assume that $F = G \to \bot$. Since $F \in \mathcal{N}$, G has no occurrences of implication and, consequently, all occurrences of atoms in G are negative in F. It follows that if $I \not\models G$ then $F_I \equiv \bot \to \bot \equiv \top$ and, otherwise (if $I \models G$) $F_I \equiv \top \to \bot \equiv \bot$.

Moreover, if $I \not\models G$, then $I \models F$. Thus, $F^I = G^I \to \bot = \bot \to \bot \equiv \top \equiv F_I$. Similarly, if $I \models G$ then $I \not\models F$ and, consequently, $F^I = \bot$. Thus, also in this case $F_I \equiv F^I$ and the assertion follows. □

Theorem 6. *Let Π be a program such that for every clause $p \leftarrow G \in \Pi$, $G \in \mathcal{N}$. Then stable and ID-stable models of Π coincide, and input stable and input ID-stable models coincide.*

Proof. Reasoning in each direction we have that I is a model of Π. By Lemmas 1 and 2, Π^I and Π_I have the same models contained in I. Thus, if I is the least model of Π_I, I is the least model (and, in particular, a minimal model) of Π^I. Conversely, if I is a

minimal model of Π^I, then I is a minimal model of Π_I and so, the least model of Π_I. These two observations imply the first part of the assertion. The second part follows from the first part and from the definitions. □

The discussion above extends in a straightforward way to the first-order case. Let us consider a first-order signature σ. A *program clause* is a formula $\forall \boldsymbol{X}\,(G \to p(\boldsymbol{t}))$, where $\boldsymbol{t}$ is a tuple of terms of the arity equal to the arity of p, and $\boldsymbol{X}$ is a tuple of all free variables in $G \to p(\boldsymbol{t})$. A *program* is a collection of clauses. As before, we write a clause $\forall \boldsymbol{X}\,(G \to p(\boldsymbol{t}))$ as $\forall \boldsymbol{X}\,(p(\boldsymbol{t}) \leftarrow G)$ or even as $p(\boldsymbol{t}) \leftarrow G$.

First, we generalize to the first-order setting the results concerning input stable models.

Definition 6. *Let Π be a program in a first-order signature σ. An interpretation $I \in Int_\sigma$ is an input stable model of Π if I^r is an input stable model of $gr_I(\Pi)$.*

For a program Π in a first-order signature σ, we define σ^{out}_Π to be the set of all relation symbols occurring in the heads of clauses in Π and σ^{in}_Π to be the set of all other relation symbols in Π. Whenever there is no ambiguity, we drop Π from the notation. We set $\Pi^{in} = \Pi \cup \{p(\boldsymbol{X}) \leftarrow \neg\neg p(\boldsymbol{X}) \mid p \in \sigma^{in}\}$. We have the following result, which can be obtained by lifting through grounding the corresponding result from the infinitary propositional setting (Proposition 5).

Theorem 7. *Let Π be a program in a first-order signature σ. An interpretation $I \in Int_\sigma$ is an input stable model of Π if and only if I is a stable model of Π^{in}.*

The concepts of ID-stable and input ID-stable models can similarly be lifted to the first-order setting.

Definition 7. *If Π is a program in a first-order signature σ, an interpretation $I \in Int_\sigma$ is an (input)* ID-stable model *of Π if I^r is an (input) ID-stable model of $gr_I(\Pi)$.*

The definition of the class $\mathcal{N}$ extends literally to the first-order language. We define the class $\mathcal{N}^{fo}$ to consist of all first-order formulas F such that every occurrence of $\to$ in F has $\bot$ in its consequent and no occurrence of $\to$ in its antecedent (elements of $\mathcal{N}^{fo}$ may contain free variables). Since grounding formulas form $\mathcal{N}^{fo}$ results in infinitary propositional formulas from $\mathcal{N}$, Theorem 6 lifts to the first-order setting.

Theorem 8. *Let Π be a program over a first-order signature σ such that for every clause $p(\boldsymbol{t}) \leftarrow G \in \Pi$, $G \in \mathcal{N}^{fo}$. Then (input) stable and (input) ID-stable models of Π coincide.*

5 Logics PC(ID) and FO(ID)

We now consider the logic FO(ID) introduced by Denecker [2] and investigated in detail by Denecker and Ternovska [5]. Our goal is to relate that logic to the formalism of first-order programs under the stable model semantics. To this end, we first relate the logic FO(ID) with the logic of (first-order) programs under the semantics of ID-stable models. Next, we apply Theorem 8 to obtain the result we are interested in.

We start by extending the logic PC(ID), the propositional version of FO(ID), to the infinitary propositional case. Our approach consists of a straightforward extension to the infinitary setting of the algebraic approach developed by Denecker et al. [4].

Let A be a propositional signature and Π a program in $\mathcal{L}_A^{inf}$. The *Fitting* operator for Π, $\Phi_\Pi \colon Int_A \times Int_A \to Int_A$, is defined by

$$\Phi_\Pi(I, J) = \{p \mid p \leftarrow F \in \Pi \text{ and } I \models F_J\}.$$

The key monotonicity properties of the operator Φ_Π in the finitary setting extend in a direct way to the infinitary one.

Proposition 6. *For every program Π, the operator Φ_Π is monotone in the first argument and antimonotone in the second.*

We recall that for a program Π in the signature A we write A_Π^{out} for the set of all atoms that appear in the heads of clauses in Π and A_Π^{in} for $A \setminus A_\Pi^{out}$. We also omit Π from the notation whenever it is clear from the context.

For every interpretation $K \in Int_{A^{in}}$, we set $Int_{A,K} = \{I \in Int_A \mid I \cap A^{in} = K\}$. Given a program Π and an interpretation $K \in Int_{A^{in}}$, we define the *Fitting operator with input* K, $\Phi_{\Pi,K} \colon Int_{A,K} \times Int_{A,K} \to Int_{A,K}$, by setting

$$\Phi_{\Pi,K}(I, J) = K \cup \Phi_\Pi(I, J),$$

where $I, J \in Int_{A,K}$. The operator $\Phi_{\Pi,K}$ is well defined, that is, to every pair of interpretations from $Int_{A,K}$ it assigns an interpretation from $Int_{A,K}$. It is easy to see that for every $I, J \in Int_A$ $(= Int_{A,\emptyset})$, $\Phi_{\Pi,\emptyset}(I, J) = \Phi_\Pi(I, J)$. Moreover, by the previous result, $\Phi_{\Pi,K}$ is monotone in the first argument and antimonotone in the second.

Definition 8. *Let Π be a program in signature A and $K \in Int_{A^{in}}$ an interpretation. The* stable operator with input K, $St_{\Pi,K} \colon Int_{A,K} \to Int_{A,K}$, *is defined by*

$$St_{\Pi,K}(J) = lfp(\Phi_{\Pi,K}(\cdot, J)) \quad \textit{(where } J \in Int_{A,K}\textit{)}.$$

We note that input ID-stable models that we introduced in the previous section can be characterized in terms of the operator $St_{\Pi,K}$.

Proposition 7. *Let Π be a program in a signature A. An interpretation $I \in Int_A$ is an input ID-stable model of Π if and only if $St_{\Pi,I\cap A^{in}}(I) = I$.*

The operator $St_{\Pi,K}$ is a stepping stone to the fundamental concept of the logic PC(ID): the *well-founded model* based on an interpretation of input atoms. To introduce it, we define an operator $W_{\Pi,K}$ on $Int_{A,K} \times Int_{A,K}$ by setting:

$$W_{\Pi,K}(I, J) = (St_{\Pi,K}(J), St_{\Pi,K}(I)) \quad (\text{where } I, J \in Int_{A,K}).$$

By Proposition 6 and the definitions of the operators $\Phi_{\Pi,K}$ and $St_{\Pi,K}$, for every $K \in Int_{A^{in}}$ the operator $St_{\Pi,K}$ is antimonotone. It follows that the operator $W_{\Pi,K}$ is monotone with respect to the *precision order* $\leq_P$ defined as follows:

$$(I, J) \leq_P (I', J') \quad \text{if and only if} \quad I \subseteq I' \text{ and } J' \subseteq J.$$

The precision order is a partial order that imposes on $Int_{A,K} \times Int_{A,K}$ the structure of a complete lattice, with the pair (K, A) as its least element. By the Tarski-Knaster theorem, $W_{\Pi,K}$ has a least fixpoint, and this fixpoint is the limit of the transfinite sequence $\{W^{\alpha}_{\Pi,K}(K, A)\}_{\alpha}$, where iterations $W^{\alpha}_{\Pi,K}(K, A)$ of the operator $W_{\Pi,K}$ over (K, A) are defined in the standard way. If $I, J \in Int_{A,K}$, $I \subseteq J$, and $W_{\Pi,K}(I, J) = (I', J')$, then $I' \subseteq J'$. Thus, for every element (I, J) in the sequence $\{W^{\alpha}_{\Pi,K}(K, A)\}_{\alpha}$, $I \subseteq J$ and the limit (the least fixpoint of $W_{\Pi,K}$), satisfies the same property.

Let us assume that (I_0, J_0) is that limit. Then, we have $I_0 \subseteq J_0$ and

$$(I_0, J_0) = W_{\Pi,K}(I_0, J_0) = (St_{\Pi,K}(J_0), St_{\Pi,K}(I_0)).$$

It follows that $I_0 = St_{\Pi,K}(J_0)$ and $I_0 \subseteq St_{\Pi,K}(I_0)$. We call the pair $(I_0, St_{\Pi,K}(I_0))$ the *well-founded model of Π based on input K*.[6] Atoms in I_0 are *true* in the model and those not in $St_{\Pi,K}(I_0)$ are *false*. All the other atoms are *unknown*. If $I_0 = St_{\Pi,K}(I_0)$, then we call the well-founded model *total*, as no atoms are *unknown* in it.

Before we proceed, we note two results generalizing well-known properties of stable and well-founded models of standard finitary programs to the present setting. The first one states that the well-founded model of Π based on input K approximates all input ID-stable models of Π such that $I \cap A^{in} = K$. The second one relates total well-founded and stable models.

Proposition 8. *If (I_0, J_0) is the well-founded model of Π based on the input K, and I is an input ID-stable model of Π such that $I \cap A^{in} = K$, then $I_0 \subseteq I \subseteq J_0$.*

Proposition 9. *Let Π be a program and $K \in Int_{A^{in}}$. If the well-founded model of Π based on input K is total, then it is of the form (I, I), where $I \cap A^{in} = K$ and I is an input ID-stable model of Π.*

The well-founded models based on a specified input form the basis of the logic PC(ID). We introduce this logic here in a form adapted to the infinitary setting.

Definition 9. *Let A be a propositional signature. A* PC(ID) theory *is a pair (F, Π), where $F \subseteq \mathcal{L}^{inf}_{A}$ and Π is a program in A. An interpretation $I \in Int_A$ is an* ID-model *of (F, Π) if I is a model of F and (I, I) is the well-founded model of Π based on input $I \cap A^{in}$.*

One can define a PC(ID) theory as consisting of a propositional theory in $\mathcal{L}^{inf}_{A}$ and a set of programs. The concept of an ID-model extends directly to that more general setting. However, all salient features of the logic PC(ID) are captured by theories with a single program (cf. Denecker et al. [15]), which is the reason for the restriction we adopted.

A program Π in a signature A is *total* if for every $K \in Int_{A^{in}}$, the well-founded model of Π based on the input K is total. Total programs are also called *definitions*. A PC(ID) theory is *total* if its program is total. PC(ID) theories that arise naturally in knowledge representation applications are total. From now on we focus on total PC(ID)

[6] The concept of the well-founded model in the finitary propositional setting is due to Van Gelder et al. [21].

theories and study the connection between the concepts of ID-models and ID-stable models.

To this end, we extend the class of programs we consider. Namely, we allow programs to contain *constraints*, that is, clauses of the form $\bot \leftarrow \varphi$, where $\varphi \in \mathcal{L}_A^{inf}$.

Definition 10. *Let Π be an infinitary program in a signature A and let Π' and Π'' consist of all non-constraint and constraint clauses in Π, respectively. An interpretation I is an* input ID-stable model *of Π if I is an input ID-stable model of Π' and a model of Π''*

Remark. Constraints can also be handled directly in the language of programs as we introduced them earlier. Let Π be a program (without constraints). An atom f such that every clause in Π with f in the head is of the form $f \leftarrow \varphi \wedge \neg f$ is an *effective contradiction* for Π. Clauses in Π with heads that are effective contradictions for Π are called *effective constraints* in Π. Here, as in the finitary case, effective constraints work as constraints. Specifically, let Π be a program without constraints (but, possibly, with effective constraints), and let Π' be a program with constraints obtained from Π by replacing each effective constraint $f \leftarrow \varphi \wedge \neg f$ with the constraint $\bot \leftarrow \varphi$. One can show that an interpretation $I \in Int_A$ is an input ID-stable model of Π if and only if I is an input ID-stable model of Π' according to Definition 10. □

Let $F \subseteq \mathcal{L}_A^{inf}$. Formulas in F can be written as constraints. Namely, we define

$$F^{\leftarrow} = \{\bot \leftarrow \neg\varphi \mid \varphi \in F\}.$$

It is clear that F and $F^{\leftarrow}$ have the same models (are equivalent). We have the following connection between total PC(ID) theories and programs with constraints.

Theorem 9. *Let F be a set of formulas from $\mathcal{L}_A^{inf}$ and Π a definition in A (without constraints). An interpretation $I \in Int_A$ is an ID-model of (F, Π) if and only if I is an input ID-stable model of $\Pi \cup F^{\leftarrow}$.*

Proof. Let I be an ID-model of (F, Π). Then I is a model of F, and (I, I) is the well-founded model of Π based on the input $I \cap A^{in}$. By Proposition 9, I is an input ID-stable model of Π. By Definition 10, I is an input ID-stable model of $\Pi \cup F^{\leftarrow}$.

Conversely, let I be an input ID-stable model of $\Pi \cup F^{\leftarrow}$. By Definition 10, I is a model of F and an input ID-stable model of Π. Let (I_0, J_0) be the well-founded model of Π based on input $I \cap A^{in}$. By Proposition 8, $I_0 \subseteq I \subseteq J_0$. Since Π is total, $I_0 = J_0$ and so, (I, I) is the well-founded model of Π based on input $I \cap A^{in}$. Thus, I is an ID-model of (F, Π). □

Input stable and input ID-stable models coincide for the class of programs whose rules have formulas from $\mathcal{N}$ as their bodies (Theorem 6). Under the restriction to that class of programs, Theorem 9 implies the following connection between ID-models of PC(ID) theories and stable models of infinitary programs.

Theorem 10. *Let $F \subseteq \mathcal{L}_A^{inf}$ and Π be a definition (without constraints) in A such that for every clause $p \leftarrow G \in \Pi$, $G \in \mathcal{N}$. Then an interpretation I is an ID-model for (F, Π) if and only if I is a stable model of $\Pi^{in} \cup F^{\leftarrow}$.*

Proof. Let I be an ID-model for (F, Π). By Theorem 9, I is an input ID-stable model of $\Pi \cup F^{\leftarrow}$. By the definitions and Theorem 6, I is an input stable model of $\Pi \cup F^{\leftarrow}$ and so, by Proposition 5, a stable model of $(\Pi \cup F^{\leftarrow})^{in} = \Pi^{in} \cup F^{\leftarrow}$ (it is straightforward to see that Proposition 5 extends to the case of programs with constraints). All these implications can be reversed and so, the converse implication follows. □

We now move on to the first-order case. An *FO(ID) theory* in a first-order signature σ is a pair (F, Π), where F is a set of sentences in σ and Π is a program in σ. We define the semantics of FO(ID) theories by lifting the semantics of the infinitary PC(ID) theories. One can show that this definition is equivalent to the original one [5].

Definition 11. *Let (F, Π) be an FO(ID) theory in a signature σ. An interpretation $I = \langle I^f, I^r \rangle$ of σ is an* ID-model *of (F, Π) if I^r is an ID-model of the PC(ID) theory $(gr_I(F), gr_I(\Pi))$ (in the propositional signature $A_{\sigma,I}$).*

A definition Π is *total*, if for every interpretation $I = \langle I^f, I^r \rangle$, the well-founded model of $gr_I(\Pi)$ based on the input $I^r \cap A_{\sigma^{in},I}$ is total. In particular, it follows that if Π is total then $gr_I(\Pi)$ is total.

Theorem 11. *Let σ be a first-order signature and let (F, Π) be an FO(ID) theory such that Π is total. An interpretation I is an ID-model of (F, Π) if and only if I is an input ID-stable model of $\Pi \cup F^{\leftarrow}$.*

Proof. The following statements are equivalent:

1. I is an ID-model of (F, Π)
2. I^r is an ID-model of $(gr_I(F), gr_I(\Pi))$
3. I^r is an input ID-model of $gr_I(\Pi) \cup [gr_I(F)]^{\leftarrow}$
4. I^r is an input ID-model of $gr_I(\Pi) \cup gr_I(F^{\leftarrow}) = gr_I(\Pi \cup F^{\leftarrow})$
5. I is an input ID-stable model of $\Pi \cup F^{\leftarrow}$.

The equivalences (1) ≡ (2) and (4) ≡ (5) follow from the corresponding definitions. The equivalence (2) ≡ (3) follows from Theorem 9 and the equivalence (3) ≡ (4) from the identity $[gr_I(\varphi)]^{\leftarrow} = gr_I(\varphi^{\leftarrow})$, which is a direct consequence of the definition of the operators $gr_I(\cdot)$ and $\{\cdot\}^{\leftarrow}$. □

The next result connects ID-models of an FO(ID) theory whose program is a definition consisting of rules with bodies in $\mathcal{N}^{fo}$ and stable models of a certain program.

Theorem 12. *Let σ be a first-order signature and let (F, Π) be an FO(ID) theory such that Π is total and for every $\forall \boldsymbol{X}\, (p(\boldsymbol{X}) \leftarrow G) \in \Pi$, $G \in \mathcal{N}^{fo}$. An interpretation I is an ID-model of (F, Π) if and only if I is a stable model of $\Pi^{in} \cup F^{\leftarrow}$.*

Proof. The following statements are equivalent:

1. I is an ID-model of (F, Π)
2. I is an input ID-stable model of $\Pi \cup F^{\leftarrow}$
3. I is an input stable model of $\Pi \cup F^{\leftarrow}$
4. I is a stable model of $\Pi^{in} \cup F^{\leftarrow}$.

The equivalence (1) $\equiv$ (2) follows by Theorem 11. The equivalence (2) $\equiv$ (3) follows from the fact that Π has the same input stable and input ID-stable models (Theorem 8), and from the definitions of input stable and input ID-stable models of programs with constraints. Finally, the equivalence (3) $\equiv$ (4) follows from Theorem 7. □

If F and Π are finite, Theorem 12 can be restated in terms of the operator SM.

Corollary 1. *Let σ be a first-order signature and let (F, Π) be an FO(ID) theory such that Π is total and for every rule $\forall \boldsymbol{X}\,(p(\boldsymbol{X}) \leftarrow G) \in \Pi$, $G \in \mathcal{N}^{fo}$. An interpretation I is an ID-model of (F, Π) if and only if I is a model of $SM[\Pi^{in} \cup F^{\leftarrow}]$.*

A representation of programs in terms of FO(ID) theories is also possible. We will outline it below. We omit proofs as they are similar to other proofs we constructed in the paper. Let σ be a first-order signature. By σ^* we denote the extension of σ with relation symbol p^* for every relation symbol $p \in \sigma$ (p^* must be of the same arity as p). For a program Π, we define Π^* to be the program obtained by replacing each negative occurrence of an atom $p(\boldsymbol{t})$ in the body of a rule in Π by $p^*(\boldsymbol{t})$. It is easy to see that Π^* is total. Next, we define $F_\sigma = \{\forall \boldsymbol{X}\,(p(\boldsymbol{X}) \leftrightarrow p^*(\boldsymbol{X})) \mid p \text{ is a relational symbol in } \sigma\}$. Finally, for an interpretation $I \in Int_\sigma$, we define I^* to be an interpretation of σ^* that has the same domain as I, coincides with I on all symbols common to the two interpretations, and interprets every relation symbol p^* in the same way as p. One can prove the following results.

Theorem 13. *Let Π be a program in a first-order signature σ. An interpretation $I \in Int_\sigma$ is an ID-stable model of Π if and only if the interpretation I^* is an ID-model of (F_σ, Π^*).*

Corollary 2. *Let Π be a program in a first-order signature σ such that for every rule $\forall \boldsymbol{X}\,(p(\boldsymbol{X}) \leftarrow G) \in \Pi$, $G \in \mathcal{N}^{fo}$. An interpretation I of σ is a stable model of Π if and only if I^* is an ID-model of (F_σ, Π^*).*

6 Conclusions

We introduced characterizations of several first-order ASP logics in terms of reducts of infinitary propositional theories. We used these characterizations to relate these logics. Under some restrictions on program components in total FO(ID) theories, these theories can be encoded as programs so that their ID-models correspond precisely with answer-sets of the programs resulting from the translation. The restricted class of theories contains, in particular, theories that arise naturally in practical applications — their definitions are represented by standard stratified logic programs. A converse encoding (under similar syntactic restrictions) is possible, too. Thus, for theories arising in the ASP practice, there is no formal difference between the logic FO(ID) and the full first-order extension of logic programming with the answer-set semantics proposed in the literature.

Acknowledgments

I am grateful to Yulia Lierler for discussions and comments related to the topic of this work.

References

1. Denecker, M.: The well-founded semantics is the principle of inductive definition. In: Dix, J., del Cerro, L.F., Furbach, U. (eds.) Proceedings of the 6th European Workshop on Logics in Artificial Intelligencer, LNCS, vol. 1489, pp. 1–16. Springer (1998)
2. Denecker, M.: Extending classical logic with inductive definitions. In: Computational Logic - CL 2000. LNCS, vol. 1861, pp. 703–717. Springer (2000)
3. Denecker, M., Lierler, Y., Truszczynski, M., Vennekens, J.: A Tarskian semantics for Answer Set Programming (2012), a manuscript.
4. Denecker, M., Marek, V., Truszczyński, M.: Approximations, stable operators, well-founded fixpoints and applications in nonmonotonic reasoning. In: Minker, J. (ed.) Logic-Based Artificial Intelligence, pp. 127–144. Kluwer Academic Publishers (2000)
5. Denecker, M., Ternovska, E.: A logic for non-monotone inductive definitions. ACM Transactions on Computational Logic 9(2) (2008)
6. Doets, K.: From Logic to Logic Programming. Foundations of Computing Series, MIT Press, Cambridge, MA (1994)
7. Ferraris, P.: Answer sets for propositional theories. In: Logic Programming and Nonmonotonic Reasoning, 8th International Conference, LPNMR 2005. LNAI, vol. 3662, pp. 119–131. Springer (2005)
8. Ferraris, P., Lee, J., Lifschitz, V.: Stable models and circumscription. Artif. Intell. 175(1), 236–263 (2011)
9. Ferraris, P., Lifschitz, V.: Mathematical foundations of answer set programming. In: Artëmov, S., Barringer, H., d'Avila Garcez, A., Lamb, L.C., Woods, J. (eds.) We Will Show Them! Essays in Honour of Dov Gabbay. pp. 615–664. College Publications (2005)
10. Gelfond, M., Lifschitz, V.: The stable semantics for logic programs. In: Proceedings of the 5th International Conference on Logic Programming (ICLP 1988). pp. 1070–1080. MIT Press (1988)
11. Gelfond, M., Lifschitz, V.: Classical negation in logic programs and disjunctive databases. New Generation Computing 9, 365–385 (1991)
12. Gelfond, M.: Representing knowledge in A-Prolog. In: Kakas, A.C., Sadri, F. (eds.) Computational Logic: Logic Programming and Beyond. LNCS, vol. 2408, pp. 413–451. Springer (2002)
13. Lierler, Y., Truszczynski, M.: Transition systems for model generators - a unifying approach. Theory and Practice of Logic Programming 11(4-5), 629–646 (2011)
14. Marek, V., Truszczyński, M.: Stable models and an alternative logic programming paradigm. In: Apt, K., Marek, W., Truszczyński, M., Warren, D. (eds.) The Logic Programming Paradigm: a 25-Year Perspective, pp. 375–398. Springer, Berlin (1999)
15. Mariën, M., Wittocx, J., Denecker, M., Bruynooghe, M.: SAT(ID): Satisfiability of propositional logic extended with inductive definitions. In: Proceedings of the 11th International Conference on Theory and Applications of Satisfiability Testing, SAT 2008. LNCS, vol. 4996, pp. 211–224. Springer, Berlin (2008)
16. Niemelä, I.: Logic programming with stable model semantics as a constraint programming paradigm. Annals of Mathematics and Artificial Intelligence 25(3-4), 241–273 (1999)
17. Oikarinen, E., Janhunen, T.: Achieving compositionality of the stable model semantics for Smodels programs. Theory and Practice of Logic Programming 5–6, 717–761 (2008)
18. Pearce, D.: A new logical characterisation of stable models and answer sets. In: Dix, J., Pereira, L.M., Przymusinski, T.C. (eds.) Non-Monotonic Extensions of Logic Programming, NMELP '96. LNCS, vol. 1216, pp. 57–70. Springer (1997)
19. Pearce, D., Valverde, A.: Quantified equilibrium logic and foundations for answer set programs. In: Proceedings of the 24th International Conference on Logic Programming. LNCS, vol. 5366, pp. 546–560. Springer (2008)

20. Reiter, R.: A logic for default reasoning. Artificial Intelligence 13(1-2), 81–132 (1980)
21. Van Gelder, A., Ross, K., Schlipf, J.: The well-founded semantics for general logic programs. Journal of the ACM 38(3), 620–650 (1991)