An FLP-Style Answer-Set Semantics for Abstract-Constraint Programs with Disjunctions*

Johannes Oetsch, Jörg Pührer, and Hans Tompits

Technische Universität Wien,
Institut für Informationssysteme 184/3,
Favoritenstraße 9-11, A-1040 Vienna, Austria,
{oetsch,puehrer,tompits}@kr.tuwien.ac.at

Abstract

We introduce an answer-set semantics for abstract-constraint programs with disjunction in rule heads in the style of Faber, Leone, and Pfeifer (FLP). To this end, we extend the definition of an answer set for logic programs with aggregates in rule bodies using the usual FLP-reduct. Additionally, we also provide a characterisation of our semantics in terms of unfounded sets, likewise generalising the standard concept of an unfounded set. Our work is motivated by the desire to have simple and rule-based definitions of the semantics of an answer-set programming (ASP) language that is close to those implemented by the most prominent ASP solvers. The new definitions are intended as a theoretical device to allow for development methods and methodologies for ASP, e.g., debugging or testing techniques, that are general enough to work for different types of solvers. We use abstract constraints as an abstraction of literals whose truth values depend on subsets of an interpretation. This includes weight constraints, aggregates, and external atoms, which are frequently used in real-world answer-set programs. We compare the new semantics to previous semantics for abstract-constraint programs and show that they are equivalent to recent extensions of the FLP semantics to propositional and first-order theories when abstract-constraint programs are viewed as theories.

1998 ACM Subject Classification D.1.6 Logic Programming, D.3.1 Formal Definitions and Theory, I.2.4 Knowledge Representation Formalisms and Methods

Keywords and phrases answer-set programming, abstract constraints, aggregates, disjunction

Digital Object Identifier 10.4230/LIPIcs.xxx.yyy.p

1 Introduction

In order to reflect various programming needs, the basic answer-set programming (ASP) language, as originally defined by Gelfond and Lifschitz [12], has been extended in several ways to accommodate constructs like aggregates, weight constraints, and external atoms. *Abstract-constraint programs* [21, 23] are generalised logic programs providing abstractions of such commonly-used constructs and thus are perfectly suited to study different language extensions in a uniform manner. Hereby, abstract constraints are dedicated literals whose truth value depends on a set of propositional atoms.

In this paper, we consider abstract-constraint programs with disjunction in the heads and define an answer-set semantics for this kind of programs in the style of Faber, Leone, and Pfeifer ("FLP" for short), based on a simple reduct-based definition extending the original one defined for disjunctive logic program for aggregates in rule bodies [6]. The FLP semantics has been introduced to provide

* This work was partially supported by the Austrian Science Fund (FWF) under grant P21698.

Conference title on which this volume is based on.
Editors: Billy Editor, Bill Editors; pp. 1–12

Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

an intuitive handling of aggregates and is implemented in the solvers DLV [4, 16] and DLVHEX [3]. Recently, the FLP semantics has also been extended to propositional theories [30] and to first-order theories with aggregates [1]. However, in contrast to these extensions, the language we consider can be viewed as the smallest superset of the languages supported by current state-of-the-art ASP solvers.

Besides the basic reduct-based definition of answer sets, we also introduce a characterisation of our semantics in terms of *unfounded sets*, generalising the standard concept of an unfounded set [5].

Concerning the semantics for abstract-constraint programs in general, among the different proposals in the literature [23, 29, 27, 22, 20, 19, 26], to the best of our knowledge, only the work of Shen, You, and Yuan [27] deals with disjunctions in the head, i.e., they consider the same language as we do. Their semantics coincides with ours for the case of convex abstract-constraint programs, which is also the fragment that is currently implemented in common ASP solvers, but their approach depends on an involved program transformation that introduces fresh atoms—a potential advantage of our definition lies in its simplicity. Moreover, while we treat abstract-constraint atoms in the spirit of the FLP semantics, Shen, You, and Yuan [27] handle them the same way as Son, Pontelli, and Tu [29]. Relations to semantics for more restricted classes of abstract-constraint programs follow from known results.

Our main motivation for developing the characterisations discussed in this paper is of a rather practical nature. We want to have clear, declarative, and rule-based definitions that capture the languages of a majority of modern ASP solvers to a large extent. The new characterisations are intended as a theoretical device to facilitate uniform development methods and methodologies for ASP, like debugging or testing techniques [25, 13], that are general enough to work for different types of solvers. Indeed, since sufficiently efficient ASP solvers became available in the late 1990s, there has never been a standard for implemented ASP languages. Different ASP solvers support different language features, some of which are syntactic sugar, while others add expressiveness to the formalism. In particular, the languages of DLV, Clasp [10, 11], and of other solvers based on the grounder lparse and its de-facto successor Gringo, like smodels [24, 28], cmodels [18], and pbmodels [20], support different features. For instance, DLV allows for disjunction in rule heads which are not supported in Clasp and many related solvers. These, on the other hand, allow for weight constraints [28] in rule heads, whereas aggregates in DLV are restricted to appear in rule bodies only.[1]

The semantics characterised in this paper conservatively extends that of DLV, providing a theoretical basis for adding, e.g., choice rules to the language of DLV. In particular, our characterisation in terms of unfounded sets can be seen as a practical step towards an implementation in DLV as unfounded sets are central elements of the evaluation strategy of this solver. The introduced semantics also coincides with that of Simons, Niemelä, and Soininen [28] implemented in Clasp whenever no negative weights appear in weight constraints. Negative weights are rarely used and their semantics have been considered unintuitive by some authors [9, 8]. Thus, our characterisations lay a solid foundation for programming support methods operating on both solver dialects of Clasp and DLV. Besides that, they are of theoretical interest as they clarify the role of aggregate domains in rule heads in extensions of the FLP semantics. Moreover, the reduct-based definition identifies a single condition on the spoiling interpretation that is necessary for extending the original definition of the FLP semantics to programs with aggregates in rule heads.

Besides the relation of our semantics with the one by Shen, You, and Yuan [27], as pointed out above, we also discuss relations to other proposals of semantics for abstract-constraint programs. As mentioned, the FLP semantics has been extended to propositional theories by Truszczyński [30]

[1] Note that the ASP solver ClaspD [2] supports both disjunctions and aggregates (more precisely, weight constraints) in rule heads but not within the same rule.

for comparison with the semantics by Ferraris [7, 8]. Like the definition of Ferraris, the FLP semantics for propositional theories depends on a recursively defined reduct. Our results show that the semantics introduced in this paper is equivalent to that proposed by Truszczyński when abstract-constraint programs are translated into theories. In this sense, our definitions reflect the semantics by Truszczyński for programs with disjunctive rules.

This paper is organised as follows. In the next section, we give some background on abstract-constraint programs and discuss how special literals often used in real-world answer-set programs can be expressed as abstract-constraint atoms. In Section 3, we first recapitulate the FLP semantics for the fragment of abstract-constraint programs corresponding to the logic-programming language it was originally designed for and discuss shortcomings of a straightforward extension of the FLP semantics to full abstract-constraint programs. We then continue with our reduct-based semantics and the characterisation in terms of unfounded sets. Section 4 presents relations to other semantics of abstract-constraint programs. Moreover, we discuss the relation to recent extensions of the FLP semantics to theories. We conclude the paper in Section 5. For space reasons, most proofs are omitted.

2 Preliminaries

We assume a fixed propositional language based on a countable set $\mathcal{A}$ of (propositional) atoms. We use "not" as the symbol for *default negation*. An *abstract-constraint atom*, or *c-atom*, is a pair $A = \langle D, C\rangle$, where $D \subseteq \mathcal{A}$ is the *domain* of A, denoted by D_A, and $C \subseteq 2^D$ is a collection of sets of atoms, called the *satisfiers* of A, denoted by C_A. The domain of a default negated c-atom $not\ A$ is given by $D_{not\,A} = D_A$. For an atom a, we identify the c-atom $\langle\{a\}, \{\{a\}\}\rangle$ with a. We call such c-atoms *elementary*.

An *abstract-constraint program*, or simply *program*, is a finite set of rules of the form

$$A_1 \vee \cdots \vee A_k \leftarrow A_{k+1}, \ldots, A_m, not\ A_{m+1}, \ldots, not\ A_n, \tag{1}$$

where $0 \leq k \leq m \leq n$ and any A_i for $1 \leq i \leq n$ is a c-atom. For a rule r of form (1), $\mathrm{B}(r) = \{A_{k+1}, \ldots, A_m, not\ A_{m+1}, \ldots, not\ A_n\}$ is the *body* of r, $\mathrm{B}^+(r) = \{A_{k+1}, \ldots, A_m\}$ is the *positive body* of r, $\mathrm{B}^-(r) = \{A_{m+1}, \ldots, A_n\}$ is the *negative body* of r, and $\mathrm{H}(r) = \{A_1, \ldots, A_k\}$ is the *head* of r. If $\mathrm{B}(r) = \emptyset$ and $\mathrm{H}(r) \neq \emptyset$, then r is a *fact*. For facts, we usually omit the symbol "$\leftarrow$". The domain of a rule r is $D_r = \bigcup_{X \in \mathrm{H}(r) \cup \mathrm{B}(r)} D_X$. A rule r of form (1) is *normal* if $k = 1$. A program is *normal* if it contains only normal rules. A program is a *logic program* if it contains only elementary c-atoms. Furthermore, a program is an *elementary-head program* if only elementary c-atoms appear in rule heads.

An *interpretation* is a set of atoms. For two sets I and X of atoms, $I|_X = I \cap X$ is the *projection* of I to X. An interpretation I *satisfies* a c-atom $\langle D, C\rangle$, symbolically $I \models \langle D, C\rangle$, if $I|_D \in C$. Moreover, $I \models not\ \langle D, C\rangle$ iff $I \not\models \langle D, C\rangle$.

A c-atom A is *monotone* if, for all interpretations I, I', if $I \subset I'$ and $I \models A$, then also $I' \models A$. A c-atom A is *convex* if, for all interpretations I, I', I'', if $I \subset I' \subset I''$, $I \models A$, and $I'' \models A$, then also $I' \models A$. Moreover, a program is monotone (resp., convex) if all contained c-atoms are monotone (resp., convex). An interpretation I satisfies a set S of c-atoms, symbolically $I \models S$, if $I \models A$ for all $A \in S$. Moreover, I satisfies a rule r, symbolically $I \models r$, if $I \models \mathrm{B}(r)$ implies $I \models A$ for some $A \in \mathrm{H}(r)$. As well, I satisfies a set Π of rules, symbolically $I \models \Pi$, if $I \models r$ for every $r \in \Pi$. If $I \models \Pi$, we say that I is a *model* of Π.

A rule r such that $I \models \mathrm{B}(r)$ is called *active under* I. The set $\Pi^I = \{r \in \Pi \mid I \models \mathrm{B}(r)\}$ of all active rules of a program Π under an interpretation I is the *FLP-reduct of* Π [6].

As mentioned in the introduction, c-atoms are used to represent special literals used in logic programming, like aggregates and weight constraints, for formal study. Such special literals have

in common that their truth values are determined by sets of atoms in an interpretation. Throughout this paper we will identify such special literals with c-atoms. As examples, since our motivation is to obtain characterisations of a semantics close to that of the popular answer-set solvers `Clasp` and `DLV`, we next illustrate how frequently-used language constructs, viz. *weight constraints* as used in `Clasp` and *aggregates* as used in `DLV`, can be represented as c-atoms. We consider variable-free variants only since variables are not needed in the remainder of the paper. Note that both `Clasp` and `DLV` rely on a grounding step before solving.

Simons, Niemelä, and Soininen [28] introduced *weight constraints* for normal logic programs. A weight constraint is an expression of form

$$l\,[a_1 = w_1, \ldots, a_k = w_k, not\ a_{k+1} = w_{k+1}, \ldots, not\ a_n = w_n]\,u\,,$$

where each a_i is an atom and each weight w_i is a real number, for $1 \leq i \leq n$. The lower bound l and the upper bound u are either a real number, ∞, or $-\infty$. However, the authors effectively require weights to be non-negative, as in their semantics negative weights are eliminated in a pre-processing step that has been claimed to lead to unintuitive results in several works [9, 8]. If all weights are non-negative, weight constraints are convex. Intuitively, the sum of weights w_i of those atoms a_i, $1 \leq i \leq k$, that are true and the weights of the atoms a_i, $k < i \leq n$, that are false must lie within the lower and the upper bound. More formally, an interpretation I satisfies a weight constraint if

$$l \leq (\sum_{1 \leq i \leq k, a_i \in I} w_i + \sum_{k < i \leq n, a_i \notin I} w_i) \leq u\,.$$

A special form of a weight constraint is a *cardinality constraint* where all weights are 1. The intuition is that lower and upper bounds define how many of the contained atoms may be true in an answer set. A further specialised form of a cardinality constraint is a *choice atom* that is of the form

$$0\,[a_1 = 1, \ldots, a_k = 1]\,k\,.$$

Choice atoms are often used in the head of a rule for non-deterministically guessing a subset of its domain $\{a_1, \ldots, a_k\}$. They are often abbreviated as $\{a_1, \ldots, a_k\}$.

A weight constraint

$$l\,[a_1 = w_1, \ldots, a_k = w_k, not\ a_{k+1} = w_{k+1}, \ldots, not\ a_n = w_n]\,u$$

corresponds to the c-atom $\langle D, C\rangle$, where $D = \{a_1, \ldots, a_n\}$ and

$$C = \{X \subseteq D \mid l \leq (\sum_{1 \leq i \leq k, a_i \in X} w_i + \sum_{k < i \leq n, a_i \notin X} w_i) \leq u\}\,.$$

We next define aggregates following Faber [5]. A *ground set* is a set of pairs of the form $\langle \vec{c} : I\rangle$, where $\vec{c}$ is a list of constants and I is a set of atoms. An *aggregate function* is of the form $f[S]$, where S is a ground set and f is an *aggregate function symbol*. Intuitively, f stands for a mapping from multisets of constants to constants. An aggregate atom is of the form $f[S] \prec c$, where $f[S]$ is an aggregate function, c is a constant called *guard*, and $\prec \in \{=, <, \leq, \geq, >\}$ is a predefined comparison operator. Given an interpretation I and a ground set S, $I(S)$ is the multiset

$$[c_1 \mid \langle c_1, \ldots, c_n : I'\rangle \in S, I' \subseteq I]\,.$$

Then, an aggregate atom $f[S] \prec c$ is satisfied by I if $f(I(S)) \prec c$. Moreover, a default negated aggregate atom $not\ f[S] \prec c$ is satisfied by I if $f[S] \prec c$ is not satisfied by I. An aggregate atom $f[S] \prec c$ can be expressed as a c-atom

$$\langle D, \{X \subseteq D \mid f(X(S)) \prec c\}\rangle\,,$$

where $D = \bigcup_{\langle \vec{c}:I' \rangle \in S} I'$.

As an example, consider the aggregate atom $\#count[S] = 1$, where

$$S = \{\langle 2 : queen\_2\_1 \rangle, \langle 2 : queen\_2\_2 \rangle, \langle 2 : queen\_2\_3 \rangle, \langle 2 : queen\_2\_4 \rangle\},$$

stemming from an instantiation of an encoding of the n-queens problem with $n = 4$. Intuitively, the aggregate atom is true when only one queen is located on row 2 of a chessboard. The aggregate function symbol $\#count$ maps a multiset of constants to its cardinality. Hence, under interpretation $I_1 = \{queen\_2\_3\}$, we have that $I_1(S) = [2]$, therefore $\#count(I_1(S)) = 1$, and hence $\#count[S] = 1$ is satisfied by I_1. For $I_2 = \{queen\_2\_3, queen\_2\_4\}$, we have $I_1(S) = [2, 2]$, therefore $\#count(I_2(S)) = 2$, and $\#count[S] = 1$ is not satisfied by I_2.

3 Reduct-Based Answer-Set Semantics

Before presenting our actual definition of an FLP-style semantics for abstract-constraint programs, we first recapitulate the FLP semantics by Faber, Pfeifer, and Leone [6] for disjunctive logic programs with aggregates appearing in rule bodies only and afterwards discuss the shortcomings of a straightforward extension of their definition to full abstract-constraint programs.

3.1 Prelude: FLP-Semantics for Elementary-Head Programs and a Straightforward Extension

As stated above, Faber, Pfeifer, and Leone [6] defined a semantics for disjunctive logic programs with aggregates appearing in rule bodies only. This class of programs, viewed as abstract-constraint programs, corresponds to the fragment of elementary-head programs. We refer to their semantics as the *FLP semantics*, defined as follows.

▶ **Definition 1** ([6])**.** Let Π be an elementary-head program. Then, an interpretation I is an *FLP answer set of* Π if $I \models \Pi^I$ and there is no $I' \subset I$ such that $I' \models \Pi^I$. The set of all FLP answer sets of Π is denoted by $AS_{FLP}(\Pi)$.

For the same class of programs, Faber [5] provided a definition of unfounded sets that we generalise to full abstract-constraint programs later on. Note that Faber considers strong negation and partial interpretations which we do not cover in this paper.

▶ **Definition 2** ([5])**.** Let Π be an elementary-head program and I an interpretation. Then, a set X of atoms is *unfounded in* Π *with respect to* I if, for each rule $r \in \Pi$ with $\mathrm{H}(r) \cap X \neq \emptyset$,

- $I \not\models \mathrm{B}(r)$,
- $I \setminus X \not\models \mathrm{B}(r)$, or
- $I \models l$, for some $l \in \mathrm{H}(r) \setminus X$.

As shown by Faber [5], a model I of a program Π is an FLP answer set of Π iff $I \cap X = \emptyset$, for each unfounded set X for Π with respect to I.

Now, let us call the *extended FLP semantics* the one obtained from Definition 1 by keeping the conditions of the definition but allowing Π to be a general abstract-constraint program. This straightforward extensions leads to undesired results, however, as we illustrate next.

As stated earlier, a popular form of aggregates used in the head of rules in ASP are choice atoms. Consider the program consisting of the fact

$$\langle \{a, b\}, \{\emptyset, \{a\}, \{b\}, \{a, b\}\} \rangle$$

which corresponds to the choice atom $\{a, b\}$. Here, the intended behaviour of a choice atom, viz. expressing a non-deterministic choice between sets $\emptyset$, $\{a\}$, $\{b\}$, and $\{a, b\}$, can only be achieved if non-minimal answer sets are permitted. The extended FLP semantics, however, allows only the empty set as an answer set of this program.

We are interested in a notion of answer set that prevents minimisation between the different satisfiers of an abstract-constraint atom and thus allows for using choice atoms with their usual meaning. This is introduced in the following.

3.2 Basic Definition and Unfounded Sets

▶ **Definition 3.** Let Π be an abstract-constraint program and I an interpretation. Then, I is an *answer set* of Π if $I \models \Pi^I$, and there is no $I' \subset I$ such that

(i) $I' \models \Pi^I$, and
(ii) for every $r \in \Pi^I$ with $I' \models \mathrm{B}(r)$, there is some $A \in \mathrm{H}(r)$ with $I' \models A$ and $I'|_{D_A} = I|_{D_A}$.

The set of answer sets of Π is denoted by $AS(\Pi)$.

This definition differs from the one of Faber, Pfeifer, and Leone [6] by the additional Condition (ii) on the spoiling interpretation I'. Intuitively, the purpose of this condition is to prevent minimisation within c-atoms.

▶ **Example 4.** Consider program Π_1 consisting of the fact

$$\langle \{a, b\}, \{\{a\}, \{b\}, \{a, b\}\} \rangle$$

that realises a choice of at least one atom from $\{a, b\}$. The answer sets of Π_1 are given by $\{a\}$, $\{b\}$, and $\{a, b\}$. Without Condition (ii), however, we would lose the answer set $\{a, b\}$ as, e.g., $\{a\} \subseteq \{a, b\}$ and $\{a\} \models \Pi^{\{a,b\}}$.

Opposed to the extended FLP semantics for programs where such a choice cannot be expressed without introducing auxiliary atoms, we do not enforce subset-minimal answer sets.

The next example illustrates that there are however minimisation effects between different c-atoms in a disjunction.

▶ **Example 5.** Consider the program

$$\Pi_2 = \langle \{a, b\}, \{\{a\}, \{b\}, \{a, b\}\} \rangle \vee \langle \{a, c\}, \{\{a, c\}\} \rangle$$

that also consist of a single (disjunctive) fact. Interpretations $\{a\}$, $\{b\}$, and $\{a, b\}$ are answer sets of Π_2. However, the satisfier $\{a, c\}$ of the second disjunct is not an answer set. Here, $\{a\}$ is the spoiling interpretation, since for

$$A = \langle \{a, b\}, \{\{a\}, \{b\}, \{a, b\}\} \rangle$$

we have $\{a\} \models A$ and $\{a\}|_{D_A} = \{a, c\}|_{D_A}$.

Often, answer sets are computed following a two-step strategy: First a model of the program is built and in the second step it is checked whether this model obeys a foundedness condition ensuring that it is an answer set. Intuitively, every set of atoms in an answer set must be "supported" by some active rule that derives one of the atoms. Here, it is important that the reason for this rule to be active does not depend on the atom it derives. Such rules are referred to as *external support* [14]. In what follows, we extend this notion to our setting.

▶ **Definition 6.** Let r be a rule, X a set of atoms, and I an interpretation. Then, r is an *external support for X with respect to I* if

- $I \models \mathrm{B}(r)$,
- $I \setminus X \models \mathrm{B}(r)$,
- there is some $A \in \mathrm{H}(r)$ with $X|_{D_A} \neq \emptyset$ and $I|_{D_A} \subseteq S$ for some $S \in C_A$, and
- for all $A \in \mathrm{H}(r)$ with $I \models A$ we have $(X \cap I)|_{D_A} \neq \emptyset$.

We next show how answer sets can be characterised in terms of external supports.

▶ **Theorem 7.** *Let Π be a program and I an interpretation. Then, I is an answer set of Π iff I is a model of Π and every X with $\emptyset \subset X \subseteq I$ has an external support $r \in \Pi$ with respect to I.*

To complete the picture, we express the absence of an external support in an interpretation by extending the concept of an *unfounded set* [17, 5] to abstract-constraint programs (for the case of total interpretations). Defining unfounded sets in terms of external supports is motivated by the duality of these notions as discussed by Lee [14].

▶ **Definition 8.** Let Π be a program, X a set of atoms, and I an interpretation. Then, X is *unfounded in Π with respect to I* if there is no rule $r \in \Pi$ that is an external support for X with respect to I.

Note that this is a conservative extension of Definition 2 for elementary-head programs.

Theorem 7 now immediately yields the following result:

▶ **Theorem 9.** *Let Π be a program and I an interpretation. Then, I is an answer set of Π iff I is a model of Π, and there is no set X with $\emptyset \subset X \subseteq I$ that is unfounded in Π with respect to I.*

Faber [5] also provides a characterisation of answer sets based on the *unfounded-freeness* property for the class of programs he considered. This concept can be lifted to the case of abstract-constraint programs under our semantics.

▶ **Definition 10.** Let Π be a program and I an interpretation. Then, I is *unfounded-free* in Π if $I \cap X = \emptyset$ for each unfounded set X in Π with respect to I.

Opposed to Theorems 7 and 9, the definition of unfounded-freeness does not restrict the considered unfounded sets to subsets of the interpretation. Therefore, it is important to note that due to the definition of external support, the part of an unfounded set contained in the interpretation is itself an unfounded set.

▶ **Proposition 11.** Let X be a set of atoms, and I an interpretation. If a rule r is an external support for $I \cap X$ with respect to I then r is an external support for X with respect to I.

▶ **Lemma 12.** *Let X be a set of atoms, Π a program, and I an interpretation. If X is unfounded in Π with respect to I then $I \cap X$ is unfounded in Π with respect to I.*

We conclude the section with the result that characterises answer sets in terms of unfounded-free models, generalising Corollary 3 of Faber [5].

▶ **Theorem 13.** *Let Π be a program and I an interpretation. Then, I is an answer set of Π iff I is a model of Π and unfounded-free in Π.*

Proof. ($\Rightarrow$) Suppose that I is an answer set of Π. By Theorem 9, I is a model of Π and it holds that ($*$) there is no set X with $\emptyset \subset X \subseteq I$ that is unfounded in Π with respect to I. Assume that I is not unfounded-free in Π. Then, there is some unfounded set X for Π with respect to I such that $I \cap X \neq \emptyset$. Hence, by Lemma 12, $I \cap X$ is an unfounded set in Π with respect to I, contradicting ($*$). ($\Leftarrow$) Towards a contradiction, assume that I is not an answer set of Π. By Theorem 9, there must be some set X with $\emptyset \subset X \subseteq I$ that is unfounded in Π with respect to I. Hence, as thus $I \cap X \neq \emptyset$, I is not unfounded-free in Π. ◀

4 Relation to other Semantics

In this section, we shed some light on commonalities and differences of our semantics with related proposals. First, we discuss relations to semantics that follow the tradition of Simons, Niemelä, and Soininen [28] and then to other FLP-style semantics. A characteristic difference of the two categories of semantics is how non-convex body literals may give support to atoms in an interpretation.

As an example, consider the program consisting of the following rules:

$$a \leftarrow \langle \{a, b\}, \{\emptyset, \{a, b\}\} \rangle,$$
$$a \leftarrow b, \quad \text{and}$$
$$b \leftarrow a.$$

While $\{a, b\}$ is an answer set under FLP-style semantics, it is not considered stable in, e.g., the semantics discussed in the following subsection.

4.1 Semantics in the Tradition of Simons, Niemelä, and Soininen

Shen, You, and Yuan [27] defined a stable model semantics for abstract-constraint programs involving disjunction, i.e., the language fragment they consider is the same as in our setting. Let us call a stable model following Shen, You, and Yuan [27] an *SYY stable model*.[2]

The following result can be shown:

▶ **Theorem 14.** *Let* Π *be a program such that all c-atoms appearing in a body of* Π *are convex. If* I *is an answer set of* Π*, then* I *is an SYY stable model of* Π.

Regarding the converse direction, an even stronger result holds:

▶ **Theorem 15.** *Let* Π *be a program. If* I *is an SYY stable model of* Π*, then* I *is an answer set of* Π.

Due to known results from the literature [27, 19, 29], Theorems 14 and 15 imply that our semantics is equivalent to a range of semantics proposed for more restricted classes of abstract-constraint programs including ones for *normal monotone abstract-constraint programs* [23, 22] and *normal convex abstract-constraint programs* [20] that are based on a non-deterministic one-step provability operator.

Furthermore, there are semantics defined for normal abstract-constraint programs where every answer set in the respective approach is an answer set as defined in our paper and where, if the considered programs are convex, also the converse holds, i.e., an answer set as defined in this paper is also an answer set in the respective approach. In particular, these include

- the approach by Liu et al. [19] based on computations,
- the work of Son, Pontelli, and Tu [29] that use the concept of conditional satisfaction of c-atoms for defining their semantics, and
- the reduct-based semantics by Shen and You [26].

Liu and Truszczyński [20] showed that their semantics for normal convex abstract-constraint programs resembles that of normal programs with weight constraints [28] with non-negative integer weights. As stated earlier, this type of weight constraints can be represented by convex abstract-constraint atoms. Due to the relation of the semantics by Liu and Truszczyński and ours, answer sets as defined in this paper coincide with stable models as defined by Simons, Niemelä, and Soininen for this class of programs. This semantics has been implemented in `smodels` and the state-of-the-art ASP solver `Clasp`.

[2] Their construction is quite involved and is omitted here for space reasons.

4.2 Semantics in the Style of Faber, Pfeifer, and Leone

The straightforwardly extended FLP semantics for abstract-constraint programs, as discussed in Section 3, and our proposed semantics are interrelated as follows.

▶ **Theorem 16.** *For any program* Π*, each extended FLP answer set of* Π *is an answer set of* Π.

As intended, for the restricted setting of elementary-head programs that was considered by Faber, Pfeifer, and Leone [6], our semantics coincides with theirs.

▶ **Theorem 17.** *For an elementary-head program* Π*, it holds that* $AS(\Pi) = AS_{FLP}(\Pi)$.

Proof. $AS_{FLP}(\Pi) \subseteq AS(\Pi)$ holds by Theorem 16. Assume now that $I \in AS(\Pi)$ but $I \notin AS_{FLP}(\Pi)$. From $I \in AS(\Pi)$ it follows that $I \models \Pi^I$. Hence, by Definition 1, there must be some $I' \subset I$ such that $I' \models \Pi^I$. Furthermore, by Definition 3, there must be some $r \in \Pi^I$ such that $I' \models \mathrm{B}(r)$ and $(*)$ for all $l \in \mathrm{H}(r)$ with $I' \models l$, $I'|_{D_l} \neq I|_{D_l}$ holds. From $r \in \Pi^I$, $I' \models \mathrm{B}(r)$, and $I' \models \Pi^I$, we get that $I' \models \mathrm{H}(r)$. Thus, there is some $l' \in \mathrm{H}(r)$ with $I' \models l$. From the definition of the satisfaction relation follows $I'|_{D_l} = \{l\}$. As $I' \subset I$ and $D_l = \{l\}$, we get $I|_{D_l} = \{l\}$, and hence $I'|_{D_l} = I|_{D_l}$. As this contradicts $(*)$, $AS(\Pi) = AS_{FLP}(\Pi)$ must hold. ◀

Truszczyński [30] introduced an FLP-style semantics for propositional theories. A main goal of his paper is to study the differences between the semantics by Faber, Pfeifer, and Leone and that of Ferraris [8]. It is worth mentioning that the same differences to the latter apply to the semantics defined in this paper. In particular, they differ in the treatment of default negated literals that are non-convex. For further information on the relation between these families of semantics, we refer to the work of Truszczyński [30] and of Lee and Meng [15] who reduce elementary-head programs under the FLP semantics to propositional formulas under the semantics of Ferraris.

For comparison with the work of Truszczyński, we consider propositional theories over the language determined by $\mathcal{A}$ and the Boolean connectives $\bot$, $\wedge$,$\vee$, and $\supset$. Moreover, we use the shorthands $\top = \bot \supset \bot$ and $\neg f = f \supset \bot$. Given an interpretation I and a formula f, the classical satisfaction relation $I \models f$ is defined as usual. Also, following custom, we identify empty disjunctions with $\bot$ and empty conjunctions with $\top$.

▶ **Definition 18** ([30])**.** Let f be a propositional formula and I an interpretation. The *T-reduct*, f^I, of f is defined inductively as follows, where a is an atom, $\circ \in \{\wedge, \vee\}$, and g and h are propositional formulas:

$$\bot^I = \bot.$$

$$a^I = \begin{cases} a & \text{if } I \models a, \\ \bot & \text{otherwise.} \end{cases}$$

$$(g \circ h)^I = \begin{cases} g^I \circ h^I & \text{if } I \models g \circ h, \\ \bot & \text{otherwise.} \end{cases}$$

$$(g \supset h)^I = \begin{cases} g \supset h^I & \text{if } I \models g \text{ and } I \models h, \\ \top & \text{if } I \not\models g, \\ \bot & \text{otherwise.} \end{cases}$$

For a propositional theory F, F^I is defined as $\{f^I \mid f \in F\}$.

▶ **Definition 19** ([30])**.** Let F be a propositional theory and I an interpretation. Then, I is a *T-answer set* of F iff I is a subset-minimal model of F^I.

Note that any T-answer set of F is also a model of F. In order to compare our semantics and the semantics by Truszczyński, we use a standard translation of abstract-constraint programs to propositional theories. To this end, we use the following representation of abstract-constraint atoms in terms of DNF formulas.

▶ **Definition 20** ([27]). Let $A = \langle D, C\rangle$ be an abstract constraint atom where D consists of atoms only. Then,

$$\varphi(A) = \bigvee_{X \in C} ((\bigwedge_{l \in X} l) \wedge (\bigwedge_{l \in D \setminus X} \neg l)) .$$

We extend the translation $\varphi(\cdot)$ to rules and abstract-constraint programs as follows.

▶ **Definition 21.** Let r be a rule of the form (1) where every A_i, $1 \leq i \leq n$, is an abstract-constraint atom whose domain is restricted to atoms. Then, $\varphi(r) = \varphi_B(r) \rightarrow \varphi_H(r)$, where

$$\varphi_H(r) = \varphi(A_1) \vee \cdots \vee \varphi(A_k) \text{ and}$$
$$\varphi_B(r) = \varphi(A_{k+1}) \wedge \cdots \wedge \varphi(A_m) \wedge \neg\varphi(A_{m+1}) \wedge \cdots \wedge \neg\varphi(A_n).$$

Finally, for a program Π, we define the propositional theory $\varphi(\Pi) = \{\varphi(r) \mid r \in \Pi\}$.

Obviously, for a rule r and an interpretation I, $I \models \mathrm{H}(r)$ iff $I \models \varphi_H(r)$, and $I \models \mathrm{B}(r)$ iff $I \models \varphi_B(r)$.

We next present the relation of our semantics to the approach of Truszczyński.

▶ **Theorem 22.** *Let Π be a program and I an interpretation. Then, I is an answer set of Π iff I is a T-answer set of $\varphi(\Pi)$.*

As Bartholomew, Lee, and Meng [1] have shown that their semantics for first-order theories with aggregates extends that of Truszczyński, the same relation applies to our approach.

5 Conclusion

In this work, we presented a new definition of answer sets for disjunctive abstract-constraint programs and a respective characterisation in terms of unfounded sets. The underlying semantics is a conservative extension of that by Faber, Pfeifer, and Leone [6] for disjunctive logic programs with aggregates in rule bodies only to the case where aggregates are also allowed in rule heads. Moreover, we showed that our semantics is also equivalent to a range of semantics that follow the understanding of Simons, Niemelä, and Soininen [28] for convex programs. Thereby, we reached our goal of providing simple definitions of an answer set that captures the essence of the semantics as implemented in popular ASP solvers like `Clasp` and `DLV`.

As regards future work, we are currently working on novel debugging techniques supporting software developers in writing answer-set programs that exploit the characterisations presented in this paper. Moreover, it would be interesting to explore how our notion of external support relates to loop formulas for abstract constraint programs as defined by You and Liu [31].

Acknowledgements

We would like to thank the reviewers for their constructive comments which helped to improve this paper.

References

1 Michael Bartholomew, Joohyung Lee, and Yunsong Meng. First-order extension of the FLP stable model semantics via modified circumscription. In *Proceedings of the 22nd International Joint Conference on Artificial Intelligence* (*IJCAI 2011*), pages 724–730. AAAI Press, 2011.

2 Christian Drescher, Martin Gebser, Torsten Grote, Benjamin Kaufmann, Arne König, Max Ostrowski, and Torsten Schaub. Conflict-driven disjunctive answer set solving. In *Proceedings of the 11th International Conference on Principles of Knowledge Representation and Reasoning* (*KR 2008*), pages 422–432. AAAI Press, 2008.

3 Thomas Eiter, Giovambattista Ianni, Roman Schindlauer, and Hans Tompits. Effective integration of declarative rules with external evaluations for semantic-web reasoning. In *Proceedings of the 3rd European Semantic Web Conference* (*ESWC 2006*), volume 4011 of *Lecture Notes in Computer Science*, pages 273–287. Springer, 2006.

4 Thomas Eiter, Nicola Leone, Cristinel Mateis, Gerald Pfeifer, and Francesco Scarcello. The KR system DLV: Progress report, comparisons and benchmarks. In *Proceedings of the 6th International Conference on Principles of Knowledge Representation and Reasoning* (*KR 1998*), pages 406–417. Morgan Kaufmann Publishers, 1998.

5 Wolfgang Faber. Unfounded sets for disjunctive logic programs with arbitrary aggregates. In *Proceedings of the 8th International Conference on Logic Programming and Nonmonotonic Reasoning* (*LPNMR 2005*), volume 3662 of *Lecture Notes in Computer Science*, pages 40–52. Springer, 2005.

6 Wolfgang Faber, Gerald Pfeifer, and Nicola Leone. Semantics and complexity of recursive aggregates in answer set programming. *Artificial Intelligence*, 175(1):278–298, 2011.

7 Paolo Ferraris. Answer sets for propositional theories. In *Proceedings of the 8th International Conference on Logic Programming and Nonmonotonic Reasoning* (*LPNMR 2005*), volume 3662 of *Lecture Notes in Computer Science*, pages 119–131. Springer, 2005.

8 Paolo Ferraris. Logic programs with propositional connectives and aggregates. *ACM Transactions on Computational Logic*, 12(4):25, 2011.

9 Paolo Ferraris and Vladimir Lifschitz. Weight constraints as nested expressions. *Theory and Practice of Logic Programming*, 5(1-2):45–74, 2005.

10 Martin Gebser, Benjamin Kaufmann, André Neumann, and Torsten Schaub. clasp: A conflict-driven answer set solver. In *Proceedings of the 9th International Conference on Logic Programming and Nonmonotonic Reasoning* (*LPNMR 2007*), volume 4483 of *Lecture Notes in Computer Science*, pages 260–265. Springer, 2007.

11 Martin Gebser, Benjamin Kaufmann, and Torsten Schaub. The conflict-driven answer set solver clasp: Progress report. In *Proceedings of the 10th International Conference on Logic Programming and Nonmonotonic Reasoning* (*LPNMR 2009*), volume 5753 of *Lecture Notes in Computer Science*, pages 509–514. Springer, 2009.

12 Michael Gelfond and Vladimir Lifschitz. Classical negation in logic programs and disjunctive databases. *New Generation Computing*, 9(3/4):365–386, 1991.

13 Tomi Janhunen, Ilkka Niemelä, Johannes Oetsch, Jörg Pührer, and Hans Tompits. On testing answer-set programs. In *Proceedings of the 19th European Conference on Artificial Intelligence* (*ECAI 2010*), volume 215 of *Frontiers in Artificial Intelligence and Applications*, pages 951–956. IOS Press, 2010.

14 Joohyung Lee. A model-theoretic counterpart of loop formulas. In Leslie Pack Kaelbling and Alessandro Saffiotti, editors, *Proceedings of the 19th International Joint Conference on Artificial Intelligence* (*IJCAI 2005*), pages 503–508, Denver, CO, USA, 2005. Professional Book Center.

15 Joohyung Lee and Yunsong Meng. On reductive semantics of aggregates in answer set programming. In *Proceedings of the 10th International Conference on Logic Programming and Nonmonotonic Reasoning* (*LPNMR 2009*), volume 5753 of *Lecture Notes in Computer Science*, pages 182–195. Springer, 2009.

16 Nicola Leone, Gerald Pfeifer, Wofgang Faber, Thomas Eiter, Georg Gottlob, Somina Perri, and Francesco Scarcello. The DLV system for knowledge representation and reasoning. *ACM Transactions on Computational Logic*, 7(3):499–562, 2006.

17 Nicola Leone, Pasquale Rullo, and Francesco Scarcello. Disjunctive stable models: Unfounded sets, fixpoint semantics, and computation. *Information and Computation*, 135(2):69–112, 1997.

18 Yuliya Lierler. CMODELS – SAT-based disjunctive answer-set solver. In *Proceedings of the 8th International Conference on Logic Programming and Nonmonotonic Reasoning* (*LPNMR 2005*), volume 3662 of *Lecture Notes in Computer Science*, pages 447–451. Springer, 2005.

19 Lengning Liu, Enrico Pontelli, Tran Cao Son, and Mirosław Truszczyński. Logic programs with abstract constraint atoms: The role of computations. *Artificial Intelligence*, 174(3-4):295–315, 2010.

20 Lengning Liu and Mirosław Truszczyński. Properties and applications of programs with monotone and convex constraints. *Journal of Artificial Intelligence Research*, 27:299–334, 2006.

21 V. Wiktor Marek and Jeffrey B. Remmel. Set constraints in logic programming. In *Proceedings of the 7th International Conference on Logic Programming and Nonmonotonic Reasoning* (*LPNMR 2004*), volume 2923 of *Lecture Notes in Computer Science*, pages 167–179. Springer, 2004.

22 Victor W. Marek, Ilkka Niemelä, and Mirosław Truszczyński. Logic programs with monotone abstract constraint atoms. *Theory and Practice of Logic Programming*, 8(2):167–199, 2008.

23 Victor W. Marek and Mirosław Truszczyński. Logic programs with abstract constraint atoms. In *Proceedings of the 19th National Conference on Artificial Intelligence* (*AAAI 2004*), pages 86–91. AAAI Press, 2004.

24 Ilkka Niemelä and Patrik Simons. Smodels - An implementation of the stable model and well-founded semantics for normal logic programs. In *Proceedings of the 4th International Conference on Logic Programming and Nonmonotonic Reasoning* (*LPNMR 1997*), volume 1265 of *Lecture Notes in Computer Science*, pages 420–429. Springer, 1997.

25 Johannes Oetsch, Jörg Pührer, and Hans Tompits. Stepping through an answer-set program. In *Proceedings of the 11th International Conference on Logic Programming and Nonmonotonic Reasoning* (*LPNMR 2011*), volume 6645 of *Lecture Notes in Computer Science*, pages 134–147. Springer, 2011.

26 Yi-Dong Shen and Jia-Huai You. A generalized Gelfond-Lifschitz transformation for logic programs with abstract constraints. In *Proceedings of the 22nd Conference on Artificial Intelligence* (*AAAI 2007*), pages 483–488. AAAI Press, 2007.

27 Yi-Dong Shen, Jia-Huai You, and Li-Yan Yuan. Characterizations of stable model semantics for logic programs with arbitrary constraint atoms. *Theory and Practice of Logic Programming*, 9(4):529–564, 2009.

28 Patrik Simons, Ilkka Niemelä, and Timo Soininen. Extending and implementing the stable model semantics. *Artificial Intelligence*, 138(1–2):181–234, 2002.

29 Tran Cao Son, Enrico Pontelli, and Phan Huy Tu. Answer sets for logic programs with arbitrary abstract constraint atoms. *Journal of Artificial Intelligence Research*, 29:353–389, 2007.

30 Mirosław Truszczyński. Reducts of propositional theories, satisfiability relations, and generalizations of semantics of logic programs. *Artificial Intelligence*, 174(16-17):1285–1306, 2010.

31 Jia-Huai You and Guohua Liu. Loop formulas for logic programs with arbitrary constraint atoms. In *Proceedings of the 23rd AAAI Conference on Artificial Intelligence* (*AAAI 2008*), pages 584–589. AAAI Press, 2008.