Chain Answer Sets for Logic Programs with Generalized Atoms – How Not To Fix a Semantic Problem

Mario Alviano[1][0000−0002−2052−2063] and Wolfgang Faber[2][0000−0002−0330−5868]

[1] University of Calabria, Italy alviano@mat.unical.it
[2] Alpen-Adria-Universität Klagenfurt, Austria wf@wfaber.com

Abstract. Answer Set Programming (ASP) has seen several extensions by generalizing the notion of atom used in these programs, for example dl-atoms, aggregate atoms, HEX atoms, generalized quantifiers, and abstract constraints, referred to collectively as generalized atoms in this paper. The idea common to all of these constructs is that their satisfaction depends on the truth values of a set of (non-generalized) atoms, rather than the truth value of a single (non-generalized) atom. In a previous work, it was argued that for some of the more intricate generalized atoms, the previously suggested semantics provide unintuitive results, and an alternative semantics called supportedly stable was suggested. Unfortunately, this semantics had a few issues on its own and also did not have a particularly natural definition. In this paper, we present a new attempt called Chain Answer Sets, which has a simple, but somewhat unusual definition. We show several properties of the new semantics, but it turns out that they are undesirable as well. Also the computational complexity of the associated reasoning tasks belong to a higher complexity class. This paper therefore shows that existing semantics have undesirable properties, and also shows how one attempt to resolve the issues fails in other ways.

1 Introduction

The basic language of Answer Set Programming (ASP) relies on Datalog with negation in rule bodies and possibly disjunction in rule heads. When actually using the language for representing practical knowledge, it became apparent that generalizations of the basic language are necessary for usability. Among the suggested extensions are aggregate atoms (similar to aggregations in database queries) [19, 18, 7, 14] and atoms that rely on external truth valuations [6, 9–11, 8]. These extensions are characterized by the fact that deciding the truth values of the new kinds of atoms depends on the truth values of a set of traditional atoms rather than a single traditional atom. We will refer to such atoms as *generalized atoms*, which cover also several other extensions such as abstract constraints, generalized quantifiers, and HEX atoms.

Concerning semantics for programs containing generalized atoms, there have been several different proposals. All of these appear to coincide for programs that do not contain generalized atoms in recursive definitions. The two main semantics that emerged as standards are the PSP semantics [20, 21, 23], and the FLP semantics [12, 13] (the

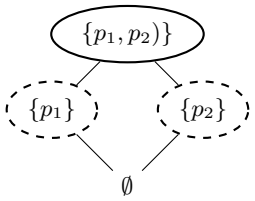

Fig. 1. Interpretations, supported (solid) and unsupported models (dashed) of the prisoners' dilemma example, where p_1 and p_2 are the propositions "the first player confesses" and "the second player confesses", respectively.

latter coinciding with Ferraris stable models [15] for the language considered in this paper). In [3] it was shown that the semantics coincide up to convex generalized atoms. It was already established earlier that each PSP answer set is also an FLP answer set, but not vice versa. So for programs containing non-convex generalized atoms, some FLP answer sets are not PSP answer sets. In particular, there are programs that have FLP answer sets but no PSP answer sets. In [4] it was argued that the FLP semantics is still too restrictive, and an attempt to improve the situation was made, defining the supportedly stable or SFLP (supportedly FLP) semantics. However, while SFLP solves some issues, it also introduces new ones.

Let us first review the reason why FLP is too restrictive. Consider a coordination game that is remotely inspired by the prisoners' dilemma. There are two players, each of which has the option to confess or defect. Let us also assume that both players have a fixed strategy already, which however still depends on the choice of the other player as well. In particular, each player will confess exactly if both players choose the same option, that is, if both players confess or both defect. This situation can be represented using two propositional atoms for "the first player confesses" and "the second player confesses", which must be derived true when "both players choose the same option", a composed proposition encoded by a generalized atom. A program encoding this scenario will not permit any answer set under the FLP, PSP, or any other semantics that we are aware of, except for SFLP. Also the more recent well-justified FLP [22] selects among the FLP answer sets and hence will have no answer set for this program either.

We point out that this is peculiar, as the scenario in which both players confess is a reasonable one; indeed, even a simple inflationary operator would result in this solution: starting from the empty set, the generalized atom associated with "both players choose the same option" is true; therefore, the atoms associated with "the first player confesses" and "the second player confesses" are derived true on the first application of the operator, which is also its fixpoint.

Looking at the reason why this is not an FLP answer set, we observe that it has two countermodels that prevent it from being an answer set, one in which only the first player confesses, and another one in which only the second player confesses (see Figure 1). Both of these countermodels are models in the classical sense, but they are weak in the sense that they are not supported, meaning that there is no rule justifying their truth. In [4], the attempt to rectify this was by requiring countermodels to be supported

as well, but it has clear weaknesses, most prominently that adding "tautological" rules like $p \leftarrow p$ can change the semantics of the program.

In this paper, we define an even stronger version of this semantics, called Chain Answer Set Semantics, which requires that countermodels are themselves answer sets of the reduct program. While at first sight it resolves the issues of SFLP, it turns out that it has rather peculiar properties, such as not guaranteeing supportedness, not satisfying the anti-chain property, and deciding the existence of Chain Answer Sets is PSPACE-complete, which make it not particularly desirable.

The remainder of this paper is structured as follows. In Section 2, we present the notation and the FLP and SFLP semantics for programs with generalized atoms. In Section 3 we define Chain Answer Sets and show how it behaves on programs that motivated its definition. In Section 4, we analyze properties of the new semantics. Finally, in Section 6, we discuss our results.

2 Background

In this section we present the notation used in this paper and present the FLP semantics [12, 13]. To ease the presentation, we will directly describe a propositional language here. This can be easily extended to the more usual ASP notations of programs involving variables, which stand for their ground versions (that are equivalent to a propositional program).

2.1 Notation

Let $\mathcal{B}$ be a countable set of *propositional atoms*. A *generalized atom* A on $\mathcal{B}$ is a pair (D_A, f_A), where $D_A \subseteq \mathcal{B}$ is the *domain* of A, and f_A is a mapping from 2^{D_A} to Boolean truth values $\{\mathbf{T}, \mathbf{F}\}$. To ease the presentation, we assume that the domain of each generalized atom is a finite set.

Example 1. Let p_1 represent the proposition "the first player confesses", and p_2 represent the proposition "the second player confesses." A generalized atom A representing the composed proposition "both players choose the same option" is such that $D_A = \{p_1, p_2\}$, $f_A(\{\}) = f_A(\{p_1, p_2\}) = \mathbf{T}$, and $f_A(\{p_1\}) = f_A(\{p_2\}) = \mathbf{F}$.

A general rule r is of the following form:

$$H(r) \leftarrow B(r) \tag{1}$$

where $H(r)$ is a disjunction $a_1 \vee \cdots \vee a_n$ $(n \geq 0)$ of propositional atoms in $\mathcal{B}$ referred to as the head of r, and $B(r)$ is a generalized atom on $\mathcal{B}$ called the body of r. For convenience, $H(r)$ is sometimes considered a set of propositional atoms. A general program P is a set of general rules. Let $At(P)$ denote the set of propositional atoms occurring in P.

It should be noted that this is a very abstract notation, aiming to be general enough to encompass many concrete languages. Languages adopted in practical systems will feature concrete syntax in place of generalized atoms, for example aggregate atoms or dl-atoms. In the sequel, we will at times also use more concrete notation in examples to ease reading.

2.2 FLP Semantics

An *interpretation* I is a subset of $\mathcal{B}$. I is a *model* for a generalized atom A, denoted $I \models A$, if $f_A(I \cap D_A) = \mathbf{T}$. Otherwise, if $f_A(I \cap D_A) = \mathbf{F}$, I is not a model of A, denoted $I \not\models A$. I is a model of a rule r of the form (1), denoted $I \models r$, if $H(r) \cap I \neq \emptyset$ whenever $I \models B(r)$. I is a model of a program P, denoted $I \models P$, if $I \models r$ for every rule $r \in P$.

Note that the fact that rule bodies are forced to be a single generalized atom is not really a limitation, and will ease the presentation of the results in the paper. In fact, a single generalized atom is sufficient for modeling conjunctions, default negation, aggregates and similar constructs.

Example 2. A conjunction $p_1 \wedge \cdots \wedge p_n$ of $n \geq 1$ propositional atoms is equivalently represented by a generalized atom A such that $D_A = \{p_1, \ldots, p_n\}$, and $f_A(B) = \mathbf{T}$ if and only if $B = \{p_1, \ldots, p_n\}$.

A conjunction $p_1, \ldots, p_m, {\sim}p_{m+1}, \ldots, {\sim}p_n$ of literals, where $n \geq m \geq 0$, $p_1, \ldots, p_n$ are propositional atoms and $\sim$ denotes *negation as failure*, is equivalently represented by a generalized atom A such that $D_A = \{p_1, \ldots, p_n\}$, and $f_A(B) = \mathbf{T}$ if and only if $\{p_1, \ldots, p_m\} \subseteq B$ and $B \cap \{p_{m+1}, \ldots, p_n\} = \emptyset$.

An aggregate $COUNT(\{p_1, \ldots, p_n\}) \neq k$, where $n \geq k \geq 0$, and $p_1, \ldots, p_n$ are propositional atoms, is equivalently represented by a generalized atom A such that $D_A = \{p_1, \ldots, p_n\}$, and $f_A(B) = \mathbf{T}$ if and only if $|B \cap D_A| \neq k$.

In the following, when convenient, we will represent generalized atoms as conjunctions of literals or aggregate atoms. Subsets of $\mathcal{B}$ mapped to true by such generalized atoms will be those satisfying the associated conjunction.

Example 3. Consider the following rules:

$$r_1 :\ a \leftarrow COUNT(\{a, b\}) \neq 1 \qquad r_2 :\ b \leftarrow COUNT(\{a, b\}) \neq 1$$

The following are general programs that will be used for illustrating the differences between the semantics considered in this paper:

$$\begin{array}{ll} P_1 := \{r_1; r_2\} & P_4 := \{r_1; r_2; a \vee b \leftarrow\} \\ P_2 := \{r_1; r_2; a \leftarrow b; b \leftarrow a\} & P_5 := \{r_1; r_2; a \leftarrow {\sim}b\} \\ P_3 := \{r_1; r_2; \leftarrow {\sim}a; \leftarrow {\sim}b\} & P_6 := \{r_1; r_2; a \leftarrow a\} \end{array}$$

Note that if a and b are replaced by p_1 and p_2, the aggregate $COUNT(\{a, b\}) \neq 1$ is equivalent to the generalized atom A from Example 1, and therefore program P_1 encodes the coordination game depicted in the introduction.

Generalized atoms can be partitioned into two classes, referred to as *convex* and *non-convex*, according to the following definition: A generalized atom A is convex if for all triples I, J, K of interpretations such that $I \subset J \subset K$, $I \models A$ and $K \models A$ implies $J \models A$. A convex program is a general program whose rules have convex bodies. Note that convex generalized atoms are closed under conjunction, but not under disjunction or complementation. In more detail, the conjunction of two generalized

atoms A, A', denoted $A \wedge A'$, is such that $D_{A \wedge A'} = D_A \cup D_{A'}$, and for all $I \subseteq D_{A \wedge A'}$, $f_{A \wedge A'}(I) = f_A(I \cap D_A) \wedge f_{A'}(I \cap D_{A'})$. The disjunction $D_{A \vee A'}$ is defined similarly, and the complementation $\overline{A}$ of A is such that $D_{\overline{A}} = D_A$, and for all $I \subseteq D_A$, $f_{\overline{A}}(I) = \neg f_A(I)$. To show that convex generalized atoms are not closed under disjunction and complementation, an example is sufficient. Let A, A' be such that $D_A = D_{A'} = \{a, b\}$, $f_A(I) = \mathbf{T}$ if and only if $I = \emptyset$, and $f_{A'}(I) = \mathbf{T}$ if and only if $I = \{a, b\}$. Hence, A, A' are convex, but $A \vee A'$ is not. However, its complement $\overline{A \vee A'}$ is convex because true only for $\{a\}$ and $\{b\}$. Closure with respect to conjunction is proved by the following claim.

Lemma 1. *The conjunction $A \wedge A'$ of two convex generalized atoms is a convex generalized atom.*

Proof. Let $I \subset J \subset K$ be such that $I \models A \wedge A'$ and $K \models A \wedge A'$. Hence, $I \models A$, $K \models A$, $I \models A'$, and $K \models A'$ by definition of $A \wedge A'$. Since A and A' are convex, we have $J \models A$ and $J \models A'$, which in turn imply $J \models A \wedge A'$. □

We now describe the FLP semantics, introduced and analyzed in [12, 13].

Definition 1 (FLP Reduct). *The FLP reduct P^I of a program P with respect to I is defined as the set $\{r \in P \mid I \models B(r)\}$.*

Definition 2 (FLP Answer Sets). *I is an FLP answer set of P if $I \models P$ and for each $J \subset I$ it holds that $J \not\models P^I$. Let $FLP(P)$ denote the set of FLP answer sets of P.*

Example 4. Consider the programs from Example 3:

- The models of P_1 are $\{a\}$, $\{b\}$ and $\{a, b\}$, none of which is an FLP answer set. Indeed, $P_1^{\{a\}} = P_1^{\{b\}} = \emptyset$, which have the trivial model $\emptyset$, which is of course a subset of $\{a\}$ and $\{b\}$. On the other hand $P_1^{\{a,b\}} = P_1$, and so $\{a\} \models P_1^{\{a,b\}}$, where $\{a\} \subset \{a, b\}$. We will discuss in the next section why this is a questionable situation.
- Concerning P_2, it has one model, namely $\{a, b\}$, which is also its unique FLP answer set. Indeed, $P_2^{\{a,b\}} = P_2$, and hence the only model of $P_2^{\{a,b\}}$ is $\{a, b\}$.
- Interpretation $\{a, b\}$ is also the unique model of program P_3, which however has no FLP answer sets. Here, $P_3^{\{a,b\}} = P_1$, hence similar to P_1, $\{a\} \models P_3^{\{a,b\}}$ and $\{a\} \subset \{a, b\}$.
- P_4 instead has two FLP answer sets, namely $\{a\}$ and $\{b\}$, and a further model $\{a, b\}$. In this case, $P_4^{\{a\}} = \{a \vee b \leftarrow\}$, and no proper subset of $\{a\}$ satisfies it. Also $P_4^{\{b\}} = \{a \vee b \leftarrow\}$, and no proper subset of $\{b\}$ satisfies it. Instead, for $\{a, b\}$, we have $P_4^{\{a,b\}} = P_4$, and hence $\{a\} \models P_4^{\{a,b\}}$ and $\{a\} \subset \{a, b\}$.
- P_5 has three models, $\{a\}$, $\{b\}$ and $\{a, b\}$, but only one FLP answer set, namely $\{a\}$. In fact, $P_5^{\{a\}} = \{a \leftarrow \sim b\}$ and $\emptyset$ is not a model of the reduct. On the other hand, $\emptyset$ is a model of $P_5^{\{b\}} = \emptyset$, and $\{a\}$ is a model of $P_5^{\{a,b\}} = P_1$.
- P_6 has the same models as P_1 and also no FLP answer set.

Models and FLP answer sets of these programs are summarized in Table 1.

Table 1. (Supported) models and (S)FLP answer sets of programs in Example 3, where A is the generalized atom $COUNT(\{a, b\}) \neq 1$.

	Rules	**Models**	**FLP**	**Supported Models**	**SFLP**
P_1	$a \leftarrow A \quad b \leftarrow A$	$\{a\}, \{b\}, \{a, b\}$	—	$\{a, b\}$	$\{a, b\}$
P_2	$a \leftarrow A \quad b \leftarrow A$ $a \leftarrow b \quad b \leftarrow a$	$\{a, b\}$	$\{a, b\}$	$\{a, b\}$	$\{a, b\}$
P_3	$a \leftarrow A \quad b \leftarrow A$ $\leftarrow \sim a \quad \leftarrow \sim b$	$\{a, b\}$	—	$\{a, b\}$	$\{a, b\}$
P_4	$a \leftarrow A \quad b \leftarrow A$ $a \vee b \leftarrow$	$\{a\}, \{b\}, \{a, b\}$	$\{a\}, \{b\}$	$\{a\}, \{b\}, \{a, b\}$	$\{a\}, \{b\}$
P_5	$a \leftarrow A \quad b \leftarrow A$ $a \leftarrow \sim b$	$\{a\}, \{b\}, \{a, b\}$	$\{a\}$	$\{a\}, \{a, b\}$	$\{a\}, \{a, b\}$
P_6	$a \leftarrow A \quad b \leftarrow A$ $a \leftarrow a$	$\{a\}, \{b\}, \{a, b\}$	—	$\{a\}, \{a, b\}$	—

2.3 SFLP Semantics

Let us now review the SFLP semantics of [4]. As noted in the introduction, the fact that P_1 has no FLP answer sets is striking. If we first assume that both a and b are false (interpretation $\emptyset$), and then apply a generalization of the well-known one-step derivability operator, we obtain truth of both a and b (interpretation $\{a, b\}$). Applying this operator once more again yields the same interpretation, a fix-point. Interpretation $\{a, b\}$ is also a supported model, that is, for all true atoms there exists a rule in which this atom is the only true head atom, and in which the body is true.

It is instructive to examine why this seemingly robust model is not an FLP answer set. Its reduct is equal to the original program, $P_1^{\{a,b\}} = P_1$. There are therefore two models of P_1, $\{a\}$ and $\{b\}$, that are subsets of $\{a, b\}$ and therefore inhibit $\{a, b\}$ from being an FLP answer set. The problem is that, contrary to $\{a, b\}$, these two models are rather weak, in the sense that they are not supported. Indeed, when considering $\{a\}$, there is no rule in P_1 such that a is the only true atom in the rule head and the body is true in $\{a\}$: The only available rule with a in the head has a false body. The situation for $\{b\}$ is symmetric.

SFLP stipulates that one should only consider supported models for finding inhibitors of answer sets. In other words, one does not need to worry about unsupported models of the reduct, even if they are subsets of the candidate. First, define supported models.

Definition 3 (Supportedness). *A model I of a program P is supported if for each $a \in I$ there is a rule $r \in P$ such that $I \cap H(r) = \{a\}$ and $I \models B(r)$. In this case we will write $I \models_s P$.*

Example 5. Continuing Example 4, programs P_1, P_2, and P_3 have one supported model, namely $\{a, b\}$. The model $\{a\}$ of P_1 is not supported because the body of the the rule with a in the head has a false body with respect to $\{a\}$. For a symmetric argument,

model $\{b\}$ of P_1 is not supported either. The supported models of P_4, instead, are $\{a\}$, $\{b\}$, and $\{a, b\}$, so all models of the program are supported. Note that both models $\{a\}$ and $\{b\}$ have the disjunctive rule as the only supporting rule for the respective single true atom, while for $\{a, b\}$, the two rules with generalized atoms serve as supporting rules for a and b. Finally, the supported models of P_5 and P_6 are $\{a\}$ and $\{a, b\}$. Supported models of these programs are summarized in Table 1.

Now let us recall SFLP answer sets from [4].

Definition 4 (SFLP Answer Sets). *I is an SFLP answer set of P if $I \models_s P$ and for each $J \subset I$ it holds that $J \not\models_s P^I$. Let $SFLP(P)$ denote the set of SFLP answer sets of P.*

Example 6. Consider again the programs from Example 3.

- Recall that P_1 has only one supported model, namely $\{a, b\}$, and $P_1^{\{a,b\}} = P_1$, but $\emptyset \not\models_s P_1^{\{a,b\}}$, $\{a\} \not\models_s P_1^{\{a,b\}}$, and $\{b\} \not\models_s P_1^{\{a,b\}}$, therefore no proper subset of $\{a, b\}$ is a supported model. Hence, it is an SFLP answer set.
- Concerning P_2, it has one model, namely $\{a, b\}$, which is supported and also its unique SFLP answer set. Indeed, recall that $P_2^{\{a,b\}} = P_2$, and hence no proper subset of $\{a, b\}$ can be a model (let alone a supported model) of $P_2^{\{a,b\}}$.
- Interpretation $\{a, b\}$ is the unique model of program P_3, which is supported and also its SFLP answer set. In fact, $P_3^{\{a,b\}} = P_1$.
- P_4 has two SFLP answer sets, namely $\{a\}$ and $\{b\}$. In this case, recall $P_4^{\{a\}} = \{a \vee b \leftarrow\}$, and no proper subset of $\{a\}$ satisfies it. Also $P_4^{\{b\}} = \{a \vee b \leftarrow\}$, and no proper subset of $\{b\}$ satisfies it. Instead, for $\{a, b\}$, we have $P_4^{\{a,b\}} = P_4$, hence since $\{a\} \models_s P_4^{\{a,b\}}$, and $\{b\} \models_s P_4^{\{a,b\}}$, we obtain that $\{a, b\}$ is not an SFLP answer set.
- P_5 has two SFLP answer sets, namely $\{a\}$ and $\{a, b\}$. In fact, $P_5^{\{a\}} = \{a \leftarrow {\sim}b\}$ and $P_5^{\{a,b\}} = P_1$.
- Finally, P_6 has no SFLP answer set. $\{a\}$ and $\{a, b\}$ are supported models. $P_6^{\{a,b\}} = P_6$, so $\{a\}$ prevents $\{a, b\}$ from being an SFLP answer set. $P_6^{\{a\}} = \{a \leftarrow a\}$, so $\emptyset \models P_6^{\{a\}}$ and so trivially also $\emptyset \models_s P_6^{\{a\}}$, preventing also $\{a\}$ from being an SFLP answer set.

The programs, models, FLP answer sets, supported models, and SFLP answer sets are summarized in Table 1.

3 Chain Answer Set Semantics

Looking at P_1 and P_6 in Table 1, it is clear that SFLP answer sets have a problem. Adding a tautological rule, which should intuitively not have any effect, causes an SFLP answer set to be invalidated. In [4] we had suggested to consider "stronger notions of supportedness" for countermodels to possibly overcome this. We next try this with a radical step: requiring countermodels to be answer sets of the reduct.

Definition 5 (Chain Answer Sets). *I is a Chain Answer Set of P if $I \models P$ and no $J \subset I$ is a Chain Answer Set of P^I. Let $CHAS(P)$ denote the set of Chain Answer Sets of P.*

Example 7. Reconsider the programs from Example 3.

- We get $CHAS(P_1) = \{\{a, b\}\}$. Indeed, for $\{a, b\}$ we have $\{a, b\} \models P_1$ and $P_1^{\{a,b\}} = P_1$. None of the subsets of $\{a, b\}$ ($\{a\}$,$\{b\}$,$\emptyset$) is in $CHAS(P_1^{\{a,b\}})$. $\emptyset \not\models P_1$, as the body of both rules is true, but their heads are false. Further, while $\{a\} \models P_1$ and $\{b\} \models P_1$, we observe that the bodies of both rules are false for these interpretations, so $P_1^{\{a\}} = P_1^{\{b\}} = \emptyset$, of which $\emptyset$ (a subset of both $\{a\}$ and $\{b\}$) is a trivial answer set. So $\{a\} \notin CHAS(P_1^{\{a,b\}})$ and $\{b\} \notin CHAS(P_1^{\{a,b\}})$.
- Also $CHAS(P_2) = \{\{a, b\}\}$. Indeed, $\{a, b\}$ is the only model of P_2, and since $P_2^{\{a,b\}} = P_2$ and we know that no subset of $\{a, b\}$ is a model for P_2, also no subset can be in $CHAS(P_2^{\{a,b\}})$.
- Once more, $CHAS(P_3) = \{\{a, b\}\}$. Also in this case, $\{a, b\}$ is the only model of P_3, and $P_3^{\{a,b\}} = P_1$, for which we have already established $CHAS(P_1) = \{\{a, b\}\}$.
- For P_4, there are three models $\{a\}$,$\{b\}$, $\{a, b\}$. We have $P_4^{\{a\}} = P_4^{\{b\}} = \{a \vee b \leftarrow\}$, and clearly $\emptyset \not\models \{a \vee b \leftarrow\}$, so $\{\{a\}, \{b\}\} \subseteq CHAS(P_4)$. Now, since $P_4^{\{a,b\}} = P_4$ and $\{\{a\}, \{b\}\} \subseteq CHAS(P_4)$, $\{a, b\} \notin CHAS(P_4)$. So $CHAS(P_4) = \{\{a\}, \{b\}\}$.
- Also P_5 has three models $\{a\}$,$\{b\}$, $\{a, b\}$. $P_5^{\{a\}} = \{a \leftarrow {\sim}b\}$ and $\emptyset \not\models P_5^{\{a\}}$, so $\emptyset \notin CHAS(P_5^{\{a\}})$ and $\{a\} \in CHAS(P_5)$. $P_5^{\{b\}} = \emptyset$, so trivially $\emptyset \in CHAS(P_5^{\{b\}})$ and $\{b\} \notin CHAS(P_5)$ $P_5^{\{a,b\}} = P_1$ Finally, $P_5^{\{a,b\}} = P_1$ and we already know $CHAS(P_1) = \{\{a, b\}\}$, so $\{a, b\} \in CHAS(P_5)$.
- P_6 has the same three models $\{a\}$,$\{b\}$, $\{a, b\}$. $P_6^{\{a\}} = \{a \leftarrow a\}$, so of course $\emptyset \models P_6^{\{a\}}$, so $\{a\} \notin CHAS(P_6)$. $P_6^{\{b\}} = \emptyset$, so of course again $\emptyset \models P_6^{\{b\}}$, so $\{b\} \notin CHAS(P_6)$. $P_6^{\{a,b\}} = P_6$, so we have already established that no subset of $\{a, b\}$ is in $CHAS(P_6^{\{a,b\}})$, and hence $CHAS(P_6) = \{\{a, b\}\}$.

4 Properties of Chain Answer Sets

4.1 Supportedness, Anti-chain Property, Relationship to FLP

Chain Answer Sets are not necessarily supported, as the following example shows.

Example 8. Consider $P_u = \{r_\alpha; r_\beta\}$, where

$$r_\alpha : a \leftarrow COUNT(\{a, b\}) \neq 1 \qquad r_\beta : b \leftarrow COUNT(\{a, b\}) < 2.$$

We have $\{a, b\} \models P_u$, and $P_u^{\{a,b\}} = \{r_\alpha\}$ consists only of the first rule. Again, $\emptyset \not\models P_u^{\{a,b\}}$, so $\emptyset \notin CHAS(P_u^{\{a,b\}})$. While $\{a\} \models P_u^{\{a,b\}}$ and $\{b\} \models P_u^{\{a,b\}}$, $P_u^{\{a,b\}^{\{a\}}}$ and $P_u^{\{a,b\}^{\{b\}}}$ are both empty, hence $\emptyset$ is a Chain Answer Set of both, and

Table 2. Chain Answer Sets and (S)FLP answer sets of programs in Example 3, where A is the generalized atom $COUNT(\{a, b\}) \neq 1$.

	Rules	**Models**	**FLP**	**SFLP**	**CHAS**
P_1	$a \leftarrow A \quad b \leftarrow A$	$\{a\}, \{b\}, \{a, b\}$	—	$\{a, b\}$	$\{a, b\}$
P_2	$a \leftarrow A \quad b \leftarrow A$ $a \leftarrow b \quad b \leftarrow a$	$\{a, b\}$	$\{a, b\}$	$\{a, b\}$	$\{a, b\}$
P_3	$a \leftarrow A \quad b \leftarrow A$ $\leftarrow \sim a \quad \leftarrow \sim b$	$\{a, b\}$	—	$\{a, b\}$	$\{a, b\}$
P_4	$a \leftarrow A \quad b \leftarrow A$ $a \vee b \leftarrow$	$\{a\}, \{b\}, \{a, b\}$	$\{a\}, \{b\}$	$\{a\}, \{b\}$	$\{a\}, \{b\}$
P_5	$a \leftarrow A \quad b \leftarrow A$ $a \leftarrow \sim b$	$\{a\}, \{b\}, \{a, b\}$	$\{a\}$	$\{a\}, \{a, b\}$	$\{a\}, \{a, b\}$
P_6	$a \leftarrow A \quad b \leftarrow A$ $a \leftarrow a$	$\{a\}, \{b\}, \{a, b\}$	—	—	$\{a, b\}$

thus $\{a\} \notin CHAS(P_u^{\{a,b\}})$ and $\{b\} \notin CHAS(P_u^{\{a,b\}})$, which in turn implies $\{a, b\} \in CHAS(P_u)$.

However, $\{a, b\} \not\models_s P_u$, as for b, while $\{a, b\} \cap H(r_\beta) = \{b\}$, clearly $\{a, b\} \not\models B(r_\beta)$.

The same example (and also P_5 of Example 3) shows that Chain Answer Sets do not guarantee the anti-chain property (that for any program, no Chain Answer Set is a subset of another Chain Answer Set).

Example 9. Reconsider P_u from Example 8 and let us determine $CHAS(P_u)$. In Example 8 we have already shown that $\{a, b\} \in CHAS(P_u)$. Clearly, $\emptyset \not\models P_u$ and $\{a\} \not\models P_u$, therefore $\emptyset \notin CHAS(P_u)$ and $\{a\} \notin CHAS(P_u)$.

For the remaining interpretation $\{b\}$, we observe $\{b\} \models P_u$, and $P_u^{\{b\}}$ consists only of the second rule. But then $\emptyset \not\models P_u^{\{b\}}$, so $\emptyset \notin CHAS(P_u^{\{b\}})$, and $\{b\} \in CHAS(P_u)$.

We therefore obtain $CHAS(P_u) = \{\{b\}, \{a, b\}\}$, showing that Chain Answer Sets do not guarantee the anti-chain property.

The fact that the definition of Chain Answer Sets does not guarantee supportedness is quite disappointing. The absence of the anti-chain property is also not nice, but seems better motivated (also SFLP does not guarantee the anti-chain property), as we shall discuss in Section 6.

As suggested by the programs of Example 3, FLP answer sets are Chain Answer Sets, but the inverse does not necessarily hold.

Proposition 1. *For any program P, $FLP(P) \subseteq CHAS(P)$.*

Proof. By Definition 2, if $I \in FLP(P)$ then $I \models P$ and for each $J \subset I$ it holds that $J \not\models P^I$. But then no such J can be in $CHAS(P^I)$, and hence according to Definition 5, $I \in CHAS(P)$. □

There are programs for which the inclusion is proper, as witnessed by P_1, P_3, P_5, and P_6 of Example 3.

Concerning the relationship to SFLP, Table 2 suggests that SFLP answer sets are Chain Answer Sets as well, but we did not prove this. As witnessed by P_6 of Example 3, there are programs that have Chain Answer Sets that are not SFLP answer sets.

5 Computational Complexity

As for the membership, we can show that checking the existence of chain answer set for a program P belong to the complexity class Σ_n^P, where n is $|heads(P)|$. The following lemma is functional to the membership result.

Lemma 2. *Let I be a set of atoms, and P be a program. Checking the existence of $J \subset I$ such that $J \in CHAS(P)$ belongs to $\Sigma^P_{|I|-1}$.*

Proof. By induction on $n \geq 1$. For $n = 1$, the only candidate is $\emptyset$, and can be checked in polynomial time; hence, the problem is in NP (actually in P). The general case holds as well because, for a guessed $J \subset I$ such that $I \models P$, we have to check that there is no $K \subset J$ such that $K \in CHAS(P^J)$; since $|J| < |I|$, the induction hypothesis tells us that the latter check can be done in $\Pi^P_{|J|}$. □

Theorem 1. *Let P be a program, and n be $|At(P)|$. Checking $CHAS(P) \neq \emptyset$ belongs to Σ_n^P.*

Proof. From Lemma 2 by noting that any $I \in CHAS(P)$ is such that $I \subseteq At(P)$. □

As for the hardness, we show a reduction from QBF validity. Let ψ be

$$\nexists \overline{x_1} \cdots \nexists \overline{x_m}\ \phi(\overline{x_0}, \ldots, \overline{x_m}) \tag{2}$$

($m \geq 2$), where ϕ is quantifier-free, and $\overline{x_i}$ ($i \in [0..m]$) are distinct sets of variables; specifically, $\overline{x_0}$ are the free variables of ψ. For a given assignment $\nu_{\overline{x_0}}$ for the variables in $\overline{x_0}$, checking $\nu_{\overline{x_0}}(\psi) = 1$ is PSPACE-complete.

We define the following program $pr(\psi)$:

$$\begin{array}{ll}
x_i^t \vee x_i^f \leftarrow & \forall i \in [0..m], x_i \in \overline{x_i} \\
x_i^t \leftarrow sat_i & \forall i \in [0..m], x_i \in \overline{x_i} \\
x_i^f \leftarrow sat_i & \forall i \in [0..m], x_i \in \overline{x_i} \\
sat_i \leftarrow [sat_{i-1} \vee {\sim} sat_i] & \forall i \in [1..m] \\
sat_m \leftarrow (D_\phi, f_\phi) &
\end{array}$$

where (D_ϕ, f_ϕ) is a generalized atom with domain $D_\phi := \{x_i^t, x_i^f \mid i \in [0..m], x_i \in \overline{x_i}\}$, and such that $f_\phi(I) = \mathbf{T}$ if and only if the following conditions are satisfied: (i) $|\{x_i^t, x_i^f\} \cap I| = 1$ for all $i \in [0..m]$ and $x_i \in \overline{x_i}$; (ii) let ν_I be such that $\nu_I(x_i)$ is 1 if $x_i^t \in I$, and 0 otherwise; then, $\nu_I(\phi) = 1$.

Moreover, we define the following mapping $int(\psi, \nu_{\overline{x_0}})$ from assignments for $\overline{x_0}$ to interpretations:

$$\{x_0^t \mid x_0 \in \overline{x_0}, \nu(x_0) = 1\} \cup$$
$$\{x_0^f \mid x_0 \in \overline{x_0}, \nu(x_0) = 0\} \cup$$
$$\{x_i^t, x_i^f \mid i \in [1..m], x_i \in \overline{x_i}\}$$
$$\{sat_i \mid i \in [1..m]\}.$$

We can establish the following link between reducts.

Lemma 3. *Program* $pr(\psi)^{int(\psi,\nu_{\overline{x_0}})}$ *is equal to* $pr(\nexists\overline{x_2} \cdots \nexists\overline{x_m}\ \phi)$.

Proof. Note that $sat_0 \notin int(\psi, \nu_{\overline{x_0}})$ and $sat_1 \in int(\psi, \nu_{\overline{x_0}})$ by construction. Therefore, the following rules are not in the reduct: any $x_0^t \leftarrow sat_0$, any $x_0^f \leftarrow sat_0$, and $sat_1 \leftarrow [sat_0 \vee {\sim}sat_1]$. □

We can now establish the link between the two problems.

Lemma 4. *For any* ψ *and* $\nu_{\overline{x_0}}$, $\nu_{\overline{x_0}}(\psi) = 1$ *if and only if* $int(\psi, \nu_{\overline{x_0}}) \in CHAS(pr(\psi))$.

Proof. By induction on m. The base case for $m = 0$ is trivial. Let us assume the claim for $m \geq 0$ and consider the case $m + 1$. Let I be $int(\psi, \nu_{\overline{x_0}}) \in CHAS(pr(\psi))$.
($\Rightarrow$) Let $\nu_{\overline{x_0}}(\psi)$ be 1. By contradiction, if $I \notin CHAS(pr(\psi))$, then there is $J \subset I$ such that $J \in CHAS(pr(\psi)^I)$. By Lemma 3, $J \in CHAS(pr(\nexists\overline{x_2} \cdots \nexists\overline{x_m}\ \phi))$. Hence, we can apply the induction hypothesis: Let $\nu_{\overline{x_1}}(x_1)$ be 1 if $x_1^t \in J$, and 0 otherwise, for all $x_1 \in \overline{x_1}$; $\nu_{\overline{x_0}} \circ \nu_{\overline{x_1}}(\nexists\overline{x_2} \cdots \nexists\overline{x_m}\ \phi) = 1$, a contradiction.
($\Leftarrow$) Let $\nu_{\overline{x_0}}(\psi)$ be 0. Hence, there is $\nu_{\overline{x_1}}$ such that $\nu_{\overline{x_0}} \circ \nu_{\overline{x_1}}(\nexists\overline{x_2} \cdots \nexists\overline{x_m}\ \phi) = 0$. Let J be $int(\psi, \nu_{\overline{x_0}} \circ \nu_{\overline{x_1}})$. Thus, $J \subset I$ by construction, and by combining the induction hypothesis and Lemma 3 we have that $J \in CHAS(pr(\psi)^I)$. That is, $I \notin CHAS(pr(\psi))$. □

We can also observe that any answer set of $pr(\psi)$ must be the image of some assignment.

Lemma 5. $I \in CHAS(pr(\psi))$ *implies the existence of* $\nu_{\overline{x_0}}$ *such that* $int(\psi, \nu_{\overline{x_0}}) = I$.

Theorem 2. *Let* P *be a program. Checking* $CHAS(P) \neq \emptyset$ *is PSPACE-complete.*

Proof. Membership is given as Theorem 1. Hardness follows from Lemma 4 and Lemma 5. □

6 Conclusion and Discussion

In this paper, we have first motivated why existing semantics for logic programs with generalized atoms do not seem satisfactory for all programs. An existing proposal to amend the issues, SFLP answer sets, introduces unintuitive results while fixing the highlighted issues. In this paper, we present another attempt at defining a semantics that repairs the issues, named Chain Answer Sets (CHAS). The definition of CHAS looks a

bit striking at first, as it refers to the defined concept itself. It is however well-defined, as the definition descends along the subset relation (even if for infinite Herbrand bases this may cause practical problems for computation).

However, it turns out that also CHAS has some peculiar properties. First of all, Chain Answer Sets are not necessarily supported. This is disappointing, as we hoped that the definition would guarantee supportedness. It could be fixed by explicitly requiring CHAS to be supported models (also SFLP explicitly required this, but there is was stipulated for symmetry reasons in the definition), but it does not seem particularly elegant.

Also, Chain Answer Sets do not guarantee the anti-chain property (and behaves like SFLP in this respect). This seems to be less disappointing, and might actually be a feature. Indeed, looking at program P_5 of Example 3, which has CHAS $\{a\}$ and $\{a, b\}$, the two answer sets stabilize in different ways, as the reducts for these two interpretations are disjoint.

A rather serious issue with Chain Answer Sets is that the CHAS existence problem is PSPACE-complete. This result suggests that the problem is most likely computationally more complex than the other semantics, and that implementations are likewise most likely to be more resource-intensive as well.

Still, as future work, implementing a reasoner supporting the new semantics would be of interest, for example by compiling the new semantics in FLP, so to use current ASP solvers such as DLV [5], CMODELS [17], CLASP [16], and WASP [1, 2]. An application area would be systems that loosely couple OWL ontologies with rule bases, for instance by means of HEX programs. As we have shown earlier, HEX atoms interfacing to ontologies will in general not be convex, and therefore using them in recursive definitions falls into our framework, where the FLP and SFLP semantics differ.

We also believe that it would be important to collect example programs that contain non-convex generalized atoms in recursive definitions. We have experimented with a few simple domains stemming from game theory (as outlined in the introduction), but we are not aware of many other attempts. Our intuition is that such programs would be written in several domains that describe features with feedback loops, which applies to many so-called complex systems. Also computing or checking properties of neural networks might be a possible application in this area.

References

1. Alviano, M., Dodaro, C., Faber, W., Leone, N., Ricca, F.: WASP: A native ASP solver based on constraint learning. In: Cabalar, P., Son, T.C. (eds.) Logic Programming and Nonmonotonic Reasoning, 12th International Conference, LPNMR 2013, Corunna, Spain, September 15-19, 2013. Proceedings. Lecture Notes in Computer Science, vol. 8148, pp. 54–66. Springer (2013). https://doi.org/10.1007/978-3-642-40564-8_6, http://dx.doi.org/10.1007/978-3-642-40564-8_6
2. Alviano, M., Dodaro, C., Ricca, F.: Anytime computation of cautious consequences in answer set programming. TPLP **14**(4-5), 755–770 (2014). https://doi.org/10.1017/S1471068414000325, http://dx.doi.org/10.1017/S1471068414000325
3. Alviano, M., Faber, W.: The complexity boundary of answer set programming with generalized atoms under the flp semantics. In: Cabalar, P., Tran, S.C. (eds.) Logic Programming

and Nonmonotonic Reasoning — 12th International Conference (LPNMR 2013). pp. 67–72. No. 8148 in Lecture Notes in AI (LNAI), Springer Verlag (Sep 2013). https://doi.org/10.1007/978-3-642-40564-8_7

4. Alviano, M., Faber, W.: Supportedly stable answer sets for logic programs with generalized atoms. In: ten Cate, B., Mileo, A. (eds.) 9th International Conference on Web Reasoning and Rule Systems (RR 2015). Lecture Notes in Computer Science, vol. 9209, pp. 30–44. Springer Verlag (Aug 2015). https://doi.org/10.1007/978-3-319-22002-4_4
5. Alviano, M., Faber, W., Leone, N., Perri, S., Pfeifer, G., Terracina, G.: The disjunctive datalog system DLV. In: Gottlob, G. (ed.) Datalog 2.0, Lecture Notes in Computer Science, vol. 6702, pp. 282–301. Springer Berlin/Heidelberg (2011)
6. Calimeri, F., Cozza, S., Ianni, G.: External sources of knowledge and value invention in logic programming. Annals of Mathematics and Artificial Intelligence **50**(3–4), 333–361 (2007)
7. Dell'Armi, T., Faber, W., Ielpa, G., Leone, N., Pfeifer, G.: Aggregate Functions in Disjunctive Logic Programming: Semantics, Complexity, and Implementation in DLV. In: Proceedings of the 18th International Joint Conference on Artificial Intelligence (IJCAI) 2003. pp. 847–852. Morgan Kaufmann Publishers, Acapulco, Mexico (Aug 2003)
8. Eiter, T., Fink, M., Ianni, G., Krennwallner, T., Redl, C., Schüller, P.: A model building framework for answer set programming with external computations. TPLP **16**(4), 418–464 (2016). https://doi.org/10.1017/S1471068415000113, https://doi.org/10.1017/S1471068415000113
9. Eiter, T., Ianni, G., Lukasiewicz, T., Schindlauer, R., Tompits, H.: Combining answer set programming with description logics for the semantic web. Artificial Intelligence **172**(12-13), 1495–1539 (2008). https://doi.org/10.1016/j.artint.2008.04.002, http://dx.doi.org/10.1016/j.artint.2008.04.002
10. Eiter, T., Ianni, G., Schindlauer, R., Tompits, H.: A Uniform Integration of Higher-Order Reasoning and External Evaluations in Answer Set Programming. In: International Joint Conference on Artificial Intelligence (IJCAI) 2005. pp. 90–96. Edinburgh, UK (Aug 2005)
11. Eiter, T., Lukasiewicz, T., Schindlauer, R., Tompits, H.: Combining Answer Set Programming with Description Logics for the Semantic Web. In: Principles of Knowledge Representation and Reasoning: Proceedings of the Ninth International Conference (KR2004), Whistler, Canada. pp. 141–151 (2004), extended Report RR-1843-03-13, Institut für Informationssysteme, TU Wien, 2003.
12. Faber, W., Leone, N., Pfeifer, G.: Recursive aggregates in disjunctive logic programs: Semantics and complexity. In: Alferes, J.J., Leite, J. (eds.) Proceedings of the 9th European Conference on Artificial Intelligence (JELIA 2004). Lecture Notes in AI (LNAI), vol. 3229, pp. 200–212. Springer Verlag (Sep 2004)
13. Faber, W., Leone, N., Pfeifer, G.: Semantics and complexity of recursive aggregates in answer set programming. Artificial Intelligence **175**(1), 278–298 (2011). https://doi.org/10.1016/j.artint.2010.04.002, special Issue: John McCarthy's Legacy
14. Faber, W., Pfeifer, G., Leone, N., Dell'Armi, T., Ielpa, G.: Design and implementation of aggregate functions in the dlv system. Theory and Practice of Logic Programming **8**(5–6), 545–580 (2008). https://doi.org/10.1017/S1471068408003323
15. Ferraris, P.: Logic programs with propositional connectives and aggregates. ACM Trans. Comput. Log. **12**(4), 25 (2011). https://doi.org/10.1145/1970398.1970401, http://doi.acm.org/10.1145/1970398.1970401
16. Gebser, M., Kaufmann, B., Schaub, T.: Conflict-driven answer set solving: From theory to practice. Artif. Intell. **187**, 52–89 (2012)
17. Lierler, Y., Maratea, M.: Cmodels-2: Sat-based answer set solver enhanced to non-tight programs. In: Lifschitz, V., Niemelä, I. (eds.) Logic Programming and Nonmonotonic Reasoning, 7th International Conference, LPNMR 2004, Fort Lauderdale, FL, USA, January 6-8, 2004, Proceedings. Lecture Notes in Computer Science, vol. 2923, pp. 346–350.

Springer (2004). https://doi.org/10.1007/978-3-540-24609-1_32, http://dx.doi.org/10.1007/978-3-540-24609-1_32
18. Niemelä, I., Simons, P.: Extending the Smodels System with Cardinality and Weight Constraints. In: Minker, J. (ed.) Logic-Based Artificial Intelligence, pp. 491–521. Kluwer Academic Publishers, Dordrecht (2000), citeseer.ist.psu.edu/niemel00extending.html
19. Niemelä, I., Simons, P., Soininen, T.: Stable Model Semantics of Weight Constraint Rules. In: Gelfond, M., Leone, N., Pfeifer, G. (eds.) Proceedings of the 5th International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR'99). Lecture Notes in AI (LNAI), vol. 1730, pp. 107–116. Springer Verlag, El Paso, Texas, USA (Dec 1999)
20. Pelov, N.: Semantics of Logic Programs with Aggregates. Ph.D. thesis, Katholieke Universiteit Leuven, Leuven, Belgium (Apr 2004)
21. Pelov, N., Denecker, M., Bruynooghe, M.: Well-founded and Stable Semantics of Logic Programs with Aggregates. Theory and Practice of Logic Programming **7**(3), 301–353 (2007)
22. Shen, Y., Wang, K., Eiter, T., Fink, M., Redl, C., Krennwallner, T., Deng, J.: FLP answer set semantics without circular justifications for general logic programs. Artificial Intelligence **213**, 1–41 (2014). https://doi.org/10.1016/j.artint.2014.05.001, http://dx.doi.org/10.1016/j.artint.2014.05.001
23. Son, T.C., Pontelli, E.: A Constructive Semantic Characterization of Aggregates in ASP. Theory and Practice of Logic Programming **7**, 355–375 (May 2007)