Unfounded Sets for Disjunctive Logic Programs with Arbitrary Aggregates*

Wolfgang Faber

Department of Mathematics, University of Calabria, 87030 Rende (CS), Italy
faber@mat.unical.it

Abstract. Aggregates in answer set programming (ASP) have recently been studied quite intensively. The main focus of previous work has been on defining suitable semantics for programs with arbitrary, potentially recursive aggregates. By now, these efforts appear to have converged. On another line of research, the relation between unfounded sets and (aggregate-free) answer sets has lately been rediscovered. It turned out that most of the currently available answer set solvers rely on this or closely related results (e.g., loop formulas).
In this paper, we unite these lines and give a new definition of unfounded sets for disjunctive logic programs with arbitrary, possibly recursive aggregates. While being syntactically somewhat different, we can show that this definition properly generalizes all main notions of unfounded sets that have previously been defined for fragments of the language.
We demonstrate that, as for restricted languages, answer sets can be crisply characterized by unfounded sets: They are precisely the unfounded-free models. This result can be seen as a confirmation of the robustness of the definition of answer sets for arbitrary aggregates. We also provide a comprehensive complexity analysis for unfounded sets, and study its impact on answer set computation.

1 Introduction

The introduction of aggregate atoms [1–8] is one of the major linguistic extensions to Answer Set Programming of the recent years. While both semantic and computational properties of standard (aggregate-free) logic programs have been deeply investigated, relatively few works have focused on logic programs with aggregates; some of their semantic properties and their computational features are still far from being fully clarified.

The proposal for answer set semantics in [8] seems to be receiving a consensus. Recent works, such as [9, 10] give further support for the plausibility of this semantics by relating it to established constructs for aggregate-free programs. In particular, [9] presented a semantics for very general programs, and showed that it coincides with both answer sets of [8] and Smodels answer sets (the latter holds for weight constraints with positive weights only). In [10] the notion of unfounded sets is extended from aggregate-free programs to programs with aggregates in a conservative way, retaining important

* This work was supported by an APART grant of the Austrian Academy of Sciences and the European Commission under projects IST-2002-33570 INFOMIX, IST-2001-37004 WASP.

semantical and computational properties. It should be noted that unfounded sets are the basis of virtually all currently available ASP solvers [11, 4, 12–15]. Extending this notion to programs with aggregates should therefore be seen as paving the way to effective and efficient systems for programs with aggregates.

However, in [10] only a fragment of the language has been considered, namely nondisjunctive programs with monotone and antimonotone aggregates. In this paper we lift this restriction and define unfounded sets for *disjunctive* programs with *arbitrary* aggregates. To this end, some substantial change in the definition is necessary to account for nonmonotone aggregates. Nevertheless, we are able to prove that our definition is a clean extension of all main previous notions of unfounded sets: On the respective fragments, our unfounded sets always coincide with the previously proposed ones.

Importantly, we can show that our notion of unfounded sets crisply characterizes both models and answer sets of [8] for arbitrary programs. We also study complexity issues for unfounded sets and put them into perspective with respect to complexity of reasoning tasks on answer sets for various fragments of programs with aggregates. Finally, we discuss the impact of our results on computation.

Summarizing, our contributions are as follows:

- We define the notion of unfounded sets for *disjunctive* logic programs with *arbitrary* aggregates. We demonstrate that this notion is a sound generalization of all main previous concepts of unfounded sets.
- We analyze the properties of our unfounded sets, which parallel those of previous definitions. We show that a unique greatest unfounded set always exists for the class of unfounded-free interpretations.
- We characterize answer sets in terms of unfounded sets. One of the results is that a model is an answer sets iff it is unfounded-free.
- We study the complexity of determining unfounded-freeness of an interpretation, and deduce the complexity for answer set checking, which turns out to be a crucial factor for the complexity of query answering.
- We indicate applications of our results; in particular, they allow to conceive how to build efficient systems for computing answer sets for programs with aggregates.

2 Logic Programs with Aggregates

2.1 Syntax

We assume that the reader is familiar with standard LP; we refer to the respective constructs as *standard atoms*, *standard literals*, *standard rules*, and *standard programs*. Two literals are said to be complementary if they are of the form p and `not` p for some atom p. Given a literal L, $\neg.L$ denotes its complementary literal. Accordingly, given a set A of literals, $\neg.A$ denotes the set $\{\neg.L \mid L \in A\}$. For further background, see [16, 17].

Set Terms. A DLP$^{\mathcal{A}}$*set term* is either a symbolic set or a ground set. A *symbolic set* is a pair $\{\mathit{Vars} : \mathit{Conj}\}$, where Vars is a list of variables and Conj is a conjunction of

standard atoms.[1] A *ground set* is a set of pairs of the form $\langle \overline{t} : Conj \rangle$, where $\overline{t}$ is a list of constants and $Conj$ is a ground (variable free) conjunction of standard atoms.

Aggregate Functions. An *aggregate function* is of the form $f(S)$, where S is a set term, and f is an *aggregate function symbol*. Intuitively, an aggregate function can be thought of as a (possibly partial) function mapping multisets of constants to a constant.

Example 1. In the examples, we adopt the syntax of DLV *to denote aggregates.* Aggregate functions currently supported by the DLV system are: $\#\texttt{count}$ (number of terms), $\#\texttt{sum}$ (sum of non-negative integers), $\#\texttt{times}$ (product of positive integers), $\#\texttt{min}$ (minimum term), $\#\texttt{max}$ (maximum term)[2].

Aggregate Literals. An *aggregate atom* is $f(S) \prec T$, where $f(S)$ is an aggregate function, $\prec \in \{=, <, \leq, >, \geq\}$ is a predefined comparison operator, and T is a term (variable or constant) referred to as guard.

Example 2. The following aggregate atoms are in DLV notation, where the latter contains a ground set and could be a ground instance of the former:

$$\#\texttt{max}\{Z : r(Z), a(Z, V)\} > Y \qquad \#\texttt{max}\{\langle 2 : r(2), a(2, k)\rangle, \langle 2 : r(2), a(2, c)\rangle\} > 1$$

An *atom* is either a standard atom or an aggregate atom. A *literal* L is an atom A or an atom A preceded by the default negation symbol `not`; if A is an aggregate atom, L is an *aggregate literal*.

DLP$^{\mathcal{A}}$ **Programs.** A DLP$^{\mathcal{A}}$ *rule* r is a construct

$$a_1 \vee \cdots \vee a_n \text{ :- } b_1, \ldots, b_k, \texttt{ not } b_{k+1}, \ldots, \texttt{ not } b_m.$$

where $a_1, \cdots, a_n$ are standard atoms, $b_1, \cdots, b_m$ are atoms, and $n \geq 1$, $m \geq k \geq 0$. The disjunction $a_1 \vee \cdots \vee a_n$ is referred to as the *head* of r while the conjunction $b_1, ..., b_k, \texttt{ not } b_{k+1}, ..., \texttt{not } b_m$ is the *body* of r. We denote the set of head atoms by $H(r)$, and the set $\{b_1, ..., b_k, \texttt{ not } b_{k+1}, ..., \texttt{not } b_m\}$ of the body literals by $B(r)$. $B^+(r)$ and $B^-(r)$ denote, respectively, the set of positive and negative literals in $B(r)$. Note that this syntax does not explicitly allow integrity constraints (rules without head atoms). They can, however, be simulated in the usual way by using a new symbol and negation.

A DLP$^{\mathcal{A}}$ *program* is a set of DLP$^{\mathcal{A}}$ rules. In the sequel, we will often drop DLP$^{\mathcal{A}}$, when it is clear from the context. A *global* variable of a rule r appears in a standard atom of r (possibly also in other atoms); all other variables are *local* variables.

Safety. A rule r is *safe* if the following conditions hold: (i) each global variable of r appears in a positive standard literal in the body of r; (ii) each local variable of r appearing in a symbolic set $\{Vars : Conj\}$ appears in an atom of $Conj$; (iii) each guard of an aggregate atom of r is a constant or a global variable. A program $\mathcal{P}$ is safe if all $r \in \mathcal{P}$ are safe. In the following we assume that DLP$^{\mathcal{A}}$ programs are safe.

[1] Intuitively, a symbolic set $\{X : a(X, Y), p(Y)\}$ stands for the set of X-values making $a(X, Y), p(Y)$ true, i.e., $\{X \,|\, \exists Y\, s.t.\ a(X, Y), p(Y)\ is\ true\}$.

[2] The first two aggregates roughly correspond, respectively, to the cardinality and weight constraint literals of Smodels. $\#\texttt{min}$ and $\#\texttt{max}$ are undefined for empty set.

2.2 Answer Set Semantics

Universe and Base. Given a DLP$^{\mathcal{A}}$ program $\mathcal{P}$, let $U_{\mathcal{P}}$ denote the set of constants appearing in $\mathcal{P}$, and $B_{\mathcal{P}}$ be the set of standard atoms constructible from the (standard) predicates of $\mathcal{P}$ with constants in $U_{\mathcal{P}}$. Given a set X, let $\overline{2}^X$ denote the set of all multisets over elements from X. Without loss of generality, we assume that aggregate functions map to $\mathbb{I}$ (the set of integers).

Example 3. $\#\texttt{count}$ is defined over $\overline{2}^{U_{\mathcal{P}}}$, $\#\texttt{sum}$ over $\overline{2}^{\mathbb{N}}$, $\#\texttt{times}$ over $\overline{2}^{\mathbb{N}^+}$, $\#\texttt{min}$ and $\#\texttt{max}$ are defined over $\overline{2}^{\mathbb{N}} - \{\emptyset\}$.

Instantiation. A *substitution* is a mapping from a set of variables to $U_{\mathcal{P}}$. A substitution from the set of global variables of a rule r (to $U_{\mathcal{P}}$) is a *global substitution for r*; a substitution from the set of local variables of a symbolic set S (to $U_{\mathcal{P}}$) is a *local substitution for* S. Given a symbolic set without global variables $S = \{Vars : Conj\}$, the *instantiation of* S is the following ground set of pairs $inst(S)$:
$\{\langle\gamma(Vars) : \gamma(Conj)\rangle \mid \gamma$ *is a local substitution for* $S\}$.[3]
A *ground instance* of a rule r is obtained in two steps: (1) a global substitution σ for r is first applied over r; (2) every symbolic set S in $\sigma(r)$ is replaced by its instantiation $inst(S)$. The instantiation $Ground(\mathcal{P})$ of a program $\mathcal{P}$ is the set of all possible instances of the rules of $\mathcal{P}$.

Interpretations. An *interpretation* for a DLP$^{\mathcal{A}}$ program $\mathcal{P}$ is a consistent set of standard ground literals, that is $I \subseteq (B_{\mathcal{P}} \cup \neg.B_{\mathcal{P}})$ such that $I \cap \neg.I = \emptyset$. A standard ground literal L is true (resp. false) w.r.t I if $L \in I$ (resp. $L \in \neg.I$). If a standard ground literal is neither true nor false w.r.t I then it is undefined w.r.t I. We denote by I^+ (resp. I^-) the set of all atoms occurring in standard positive (resp. negative) literals in I. We denote by $\bar{I}$ the set of undefined atoms w.r.t. I (i.e. $B_{\mathcal{P}} \setminus I^+ \cup I^-$). An interpretation I is *total* if $\bar{I}$ is empty (i.e., $I^+ \cup \neg.I^- = B_{\mathcal{P}}$), otherwise I is *partial*.

An interpretation also provides a meaning for aggregate literals. Their truth value is first defined for total interpretations, and then generalized to partial ones.

Let I be a total interpretation. A standard ground conjunction is true (resp. false) w.r.t I if all its literals are true (resp. false). The meaning of a set, an aggregate function, and an aggregate atom under an interpretation, is a multiset, a value, and a truth-value, respectively. Let $f(S)$ be a an aggregate function. The valuation $I(S)$ of S w.r.t. I is the multiset of the first constant of the elements in S whose conjunction is true w.r.t. I. More precisely, let $I(S)$ denote the multiset $[t_1 \mid \langle t_1, ..., t_n : Conj\rangle \in S \wedge\ Conj$ *is true w.r.t.* $I\,]$. The valuation $I(f(S))$ of an aggregate function $f(S)$ w.r.t. I is the result of the application of f on $I(S)$. If the multiset $I(S)$ is not in the domain of f, $I(f(S)) = \bot$ (where $\bot$ is a fixed symbol not occurring in $\mathcal{P}$).

An instantiated aggregate atom A of the form $f(S) \prec k$ is *true w.r.t.* I if: (i) $I(f(S)) \neq \bot$, and, (ii) $I(f(S)) \prec k$ holds; otherwise, A is false. An instantiated aggregate literal $\texttt{not}\ A = \texttt{not}\ f(S) \prec k$ is *true w.r.t.* I if (i) $I(f(S)) \neq \bot$, and, (ii) $I(f(S)) \prec k$ does not hold; otherwise, A is false.

[3] Given a substitution σ and a DLP$^{\mathcal{A}}$ object Obj (rule, set, etc.), we denote by $\sigma(Obj)$ the object obtained by replacing each variable X in Obj by $\sigma(X)$.

If I is a *partial* interpretation, an aggregate literal A is true (resp. false) w.r.t. I if it is true (resp. false) w.r.t. *each total* interpretation J extending I (i.e., $\forall\ J$ s.t. $I\ \subseteq\ J$, A is true (resp. false) w.r.t. J); otherwise it is undefined.

Example 4. Consider the atom $A = \#\mathtt{sum}\{\langle 1 : p(2,1)\rangle, \langle 2 : p(2,2)\rangle\} > 1$. Let S be the ground set in A. For the interpretation $I = \{p(2,2)\}$, each extending total interpretation contains either $p(2,1)$ or $\mathtt{not}\ p(2,1)$. Therefore, either $I(S) = [2]$ or $I(S) = [1,2]$ and the application of $\#\mathtt{sum}$ yields either $2 > 1$ or $3 > 1$, hence A is true w.r.t. I.

Remark 1. Our definitions of interpretation and truth values preserve "knowledge monotonicity". If an interpretation J extends I (i.e., $I \subseteq J$), then each literal which is true w.r.t. I is true w.r.t. J, and each literal which is false w.r.t. I is false w.r.t. J as well.

Minimal Models. Given an interpretation I, a rule r is *satisfied w.r.t. I* if some head atom is true w.r.t. I whenever all body literals are true w.r.t. I. A total interpretation M is a *model* of a DLP$^{\mathcal{A}}$ program $\mathcal{P}$ if all $r \in Ground(\mathcal{P})$ are satisfied w.r.t. M. A model M for $\mathcal{P}$ is (subset) minimal if no model N for $\mathcal{P}$ exists such that $N^+ \subset M^+$. Note that, under these definitions, the word *interpretation* refers to a possibly partial interpretation, while a *model* is always a total interpretation.

Answer Sets. We now recall the generalization of the Gelfond-Lifschitz transformation and answer sets for DLP$^{\mathcal{A}}$ programs from [8]: Given a ground DLP$^{\mathcal{A}}$ program $\mathcal{P}$ and a total interpretation I, let $\mathcal{P}^I$ denote the transformed program obtained from $\mathcal{P}$ by deleting all rules in which a body literal is false w.r.t. I. I is an answer set of a program $\mathcal{P}$ if it is a minimal model of $Ground(\mathcal{P})^I$.

Example 5. Consider interpretation $I_1 = \{p(a)\}$, $I_2 = \{\mathtt{not}\ p(a)\}$ and two programs $P_1 = \{p(a) :\!-\ \#\mathtt{count}\{X : p(X)\} > 0.\}$ and $P_2 = \{p(a) :\!-\ \#\mathtt{count}\{X : p(X)\} < 1.\}$.

$Ground(P_1) = \{p(a) :\!-\ \#\mathtt{count}\{\langle a : p(a)\rangle\} > 0.\}$ and $Ground(P_1)^{I_1} = Ground(P_1)$, $Ground(P_1)^{I_2} = \emptyset$. Furthermore, $Ground(P_2) = \{p(a) :\!-\ \#\mathtt{count}\{\langle a : p(a)\rangle\} < 1.\}$, and $Ground(P_2)^{I_1} = \emptyset$, $Ground(P_2)^{I_2} = Ground(P_2)$ hold.

I_2 is the only answer set of P_1 (since I_1 is not a minimal model of $Ground(P_1)^{I_1}$), while P_2 admits no answer set (I_1 is not a minimal model of $Ground(P_2)^{I_1}$, and I_2 is not a model of $Ground(P_2) = Ground(P_2)^{I_2}$).

Note that any answer set A of $\mathcal{P}$ is also a model of $\mathcal{P}$ because $Ground(\mathcal{P})^A \subseteq Ground(\mathcal{P})$, and rules in $Ground(\mathcal{P}) - Ground(\mathcal{P})^A$ are satisfied w.r.t. A.

Monotonicity. Given two interpretations I and J we say that $I \leq J$ if $I^+ \subseteq J^+$ and $J^- \subseteq I^-$. A ground literal ℓ is *monotone*, if for all interpretations I, J, such that $I \leq J$, we have that: (i) ℓ true w.r.t. I implies ℓ true w.r.t. J, and (ii) ℓ false w.r.t. J implies ℓ false w.r.t. I. A ground literal ℓ is *antimonotone*, if the opposite happens, that is, for all interpretations I, J, such that $I \leq J$, we have that: (i) ℓ true w.r.t. J implies ℓ true w.r.t. I, and (ii) ℓ false w.r.t. I implies ℓ false w.r.t. J. A ground literal ℓ is *nonmonotone*, if it is neither monotone nor antimonotone.

Note that positive standard literals are monotone, whereas negative standard literals are antimonotone. Aggregate literals may be monotone, antimonotone or nonmonotone, regardless whether they are positive or negative. Nonmonotone literals include the sum over (possibly negative) integers and the average.

3 Unfounded Sets

We now give a definition of unfounded set for arbitrary $\mathrm{DLP}^{\mathcal{A}}$ programs. It should be noted that it is not possible to just take over the previous definitions in [18, 11, 10], as all of them make a distinction on the kind of atoms, be it positive and negative atoms, or the generalized version of monotone and antimonotone atoms. Just as in [8], where the same problem with the transformation of the program was lifted, we need to introduce a novel definition, which does not distinguish between the kinds of atoms.

In the following we denote by $S1 \dot\cup \neg.S2$ the set $(S_1 \setminus S_2) \cup \neg.S_2$, where S_1 and S_2 are sets of standard ground literals.

Definition 1 (Unfounded Set). *A set X of ground atoms is an unfounded set for a program $\mathcal{P}$ w.r.t. an interpretation I if, for each rule r in $Ground(\mathcal{P})$ having some atoms from X in the head, at least one of the following conditions holds:*

1. *some literal of $B(r)$ is false w.r.t. I,*
2. *some literal of $B(r)$ is false w.r.t. $I \dot\cup \neg.X$, or*
3. *some atom of $H(r) \setminus X$ is true w.r.t. I.*

Intuitively, conditions 1 and 3 state that rule satisfaction does not depend on the atoms in X, while condition 2 ensures that the rule is satisfied also if the atoms in X are switched to false. Note that $\emptyset$ is always an unfounded set, independent of interpretation and program.

Example 6. Let interpretation $I_0 = \emptyset$ and $\mathcal{P} = \{a(0) \vee a(1) \text{ :- } \#\mathtt{avg}\{X : a(X)\} = 1.,$ $a(2) \vee a(1) \text{ :- } \#\mathtt{avg}\{X : a(X)\} = 1.\}$. The unfounded sets w.r.t. I_0 are $\emptyset$, $\{a(0), a(1)\}$, $\{a(1), a(2)\}$, and $\{a(0), a(1), a(2)\}$. Only condition 2 applies in these cases. For $I_1 = \{a(0)\}$, $\{a(1), a(2)\}$, and $\{a(0), a(1), a(2)\}$ are unfounded sets. For $I_2 = \{a(1)\}$, $\{a(0)\}$, $\{a(2)\}$, $\{a(0), a(2)\}$, and $\{a(0), a(1), a(2)\}$ are unfounded sets. For $I_3 = \{\mathtt{not}\ a(0), \mathtt{not}\ a(1)\}$, $\{a(0), a(1), a(2)\}$ and all of its subsets are unfounded sets.

In the sequel, we will demonstrate the robustness of Def. 1, and show that some crucial properties of unfounded sets of nondisjunctive, aggregate-free programs continue to hold, while a few others do not, basically mirroring unfounded sets for disjunctive, aggregate-free programs. We first show that Def. 1 is a generalization of a previous definition of unfounded sets for aggregate-free programs:

Theorem 1. *For an aggregate-free program $\mathcal{P}$ and interpretation I, any unfounded set w.r.t. Def. 1 is an unfounded set as defined in [11].*

Proof. Recall that a set X of ground atoms is unfounded w.r.t. Def. 3.1 of [11], if at least one of the following conditions holds for each rule r in $Ground(P)$ having some atoms from X in the head: (a) $B(r) \cap \neg.I \neq \emptyset$, or (b) $B^+(r) \cap X \neq \emptyset$, or (c) $(H(r) \setminus X) \cap I \neq \emptyset$. On the other hand, in the aggregate-free case, Def. 1 amounts to: (1) $B(r) \cap \neg.I \neq \emptyset$, or (2) $B(r) \cap \neg.(I \dot\cup \neg.X) \neq \emptyset$, or (3) $(H(r) \setminus X) \cap I \neq \emptyset$.

Obviously, (a) is equivalent to (1), and (c) is equivalent to (2). Now observe that $B^+(r) \cap X \neq \emptyset$ implies $B(r) \cap X \neq \emptyset$, which implies $B(r) \cap (\neg.(I \setminus X) \cup X) \neq \emptyset$ which is equivalent to $B(r) \cap \neg.((I \setminus X) \cup \neg.X) \neq \emptyset \Leftrightarrow B(r) \cap \neg.(I \dot\cup \neg.X) \neq \emptyset$, so

(b) implies (2). On the other hand, (2) is equivalent to $B(r) \cap ((\neg.I \setminus \neg.X) \cup X) \neq \emptyset$, and therefore (A) $B(r) \cap (\neg.I \setminus \neg.X) \neq \emptyset$ or (B) $B(r) \cap X \neq \emptyset$ holds. (A) clearly implies (a), and (B) implies (b) because X contains only atoms, hence $B^-(r) \cap X = \emptyset$. In total, (2) implies (a) or (b). □

By Proposition 3.3 in [11], unfounded sets of [11] generalize the "original" unfounded sets of [18], which were defined for nondisjunctive programs. Therefore it follows from Theorem 1 that also unfounded sets of Def. 1 generalize those of [18] on nondisjunctive programs without aggregates.

Corollary 1. *For a nondisjunctive, aggregate-free program P and interpretation I, any unfounded set w.r.t. Def. 1 is a standard unfounded set (as defined in [18]).*

Recently, unfounded sets have been defined for nondisjunctive programs with monotone and antimonotone aggregates in [10]. Def. 1 also generalizes this notion.

Theorem 2. *For a nondisjunctive program P with only monotone and antimonotone aggregates and interpretation I, any unfounded set w.r.t. Def. 1 is an unfounded set w.r.t. [10].*

Proof. A set X of ground atoms is unfounded w.r.t. [10], if at least one of the following conditions holds for each rule r in $Ground(P)$ having some atoms from X in the head: (a) some antimonotone body literal of r is false w.r.t. I, and (b) some monotone body literal of r is false w.r.t. $I \dot\cup \neg.X$.

We first observe that condition 3 is always false for nondisjunctive programs, as $H(r) \setminus X = \emptyset$, since $H(r) \cap X \neq \emptyset$ and $|H(r)| \leq 1$.

Now, observe that $I \dot\cup \neg.X \leq I$ holds. So, if a monotone body literal of r is false w.r.t. I, it is also false w.r.t. $I \dot\cup \neg.X$, and if an antimonotone body literal of r is false w.r.t. $I \dot\cup \neg.X$, it must be false w.r.t. I. Therefore, if condition 1 holds for a monotone literal, also condition 2 and (b) hold for this literal; conversely, if condition 2 holds for an antimonotone literal, also condition 1 and (a) hold for it. So, since (a) and (b) trivially imply condition 1 and 2, respectively, we obtain equivalence. □

The union of two unfounded sets of nondisjunctive, aggregate-free programs is guaranteed to be an unfounded set as well. For disjunctive programs, this does not hold; also the addition of nonmonotone aggregates invalidates this property.

Observation 3 *If X_1 and X_2 are unfounded sets for a program $\mathcal{P}$ w.r.t. I, then $X_1 \cup X_2$ is not necessarily an unfounded set for $\mathcal{P}$ w.r.t. I, even if $\mathcal{P}$ is nondisjunctive.*

Example 7. Consider $I = \{a(0), a(1), a(-1)\}$ and $\mathcal{P}$ = $\{a(1) :- \#\mathtt{avg}\{X : a(X)\} = 0.,$ $a(-1) :- \#\mathtt{avg}\{X : a(X)\} = 0., a(0).\}$. Both $\{a(1)\}$ and $\{a(-1)\}$ are unfounded sets for $\mathcal{P}$ w.r.t. I, while $\{a(1), a(-1)\}$ is not unfounded.

In this example, some elements in unfounded sets occur also in the interpretation. This is not a coincidence, as shown by the following proposition.

Proposition 1. *If X_1 and X_2 are unfounded sets for a program $\mathcal{P}$ w.r.t. I and both $X_1 \cap I = \emptyset$ and $X_2 \cap I = \emptyset$ hold, then $X_1 \cup X_2$ is an unfounded set for $\mathcal{P}$ w.r.t. I.*

Proof. Consider a rule r where $H(r) \cap X_1 \neq \emptyset$ (symmetric arguments hold for X_2). At least one of the conditions of Def. 1 holds w.r.t. X_1. We will show that the conditions also hold w.r.t. $X_1 \cup X_2$.

If condition 1 holds w.r.t. X_1, then it trivially holds also for $X_1 \cup X_2$. If condition 2 holds, a body literal is false w.r.t. $I \dot\cup \neg.X_1$, so it is false w.r.t. $I \cup \neg.X_1$ (since $I \cap X_1 = \emptyset$). Because of Remark 1, it is then also false w.r.t. $I \cup \neg.X_1 \cup \neg.X_2 = I \dot\cup \neg.(X_1 \cup X_2)$. If condition 3 holds, some atom a of $H(r) \setminus X_1$ is true w.r.t. I, so $a \in I$. It follows that $a \notin X_2$, and so $a \in H(r) \setminus (X_1 \cup X_2)$ is still true w.r.t. I. □

We next define interpretations which never contain any element of their unfounded sets.

Definition 2 (Unfounded-free Interpretation). *Let I be an interpretation for a program $\mathcal{P}$. I is unfounded-free if $I \cap X = \emptyset$ for each unfounded set X for $\mathcal{P}$ w.r.t. I.*

For unfounded-free interpretations, Prop. 1 holds for all unfounded sets.

Corollary 2. *If X_1 and X_2 are unfounded sets for a program $\mathcal{P}$ w.r.t. an unfounded-free interpretation I, then also $X_1 \cup X_2$ is an unfounded set for $\mathcal{P}$ w.r.t. I.*

We can therefore define the *Greatest Unfounded Set* (GUS) for unfounded-free interpretations as the union of all unfounded sets. Note that for non-unfounded-free interpretations, there is in general no unique GUS, as demonstrated in Ex. 7.

Definition 3. *Given a program $\mathcal{P}$ and an unfounded-free interpretation I, let $GUS_{\mathcal{P}}(I)$ (the GUS for $\mathcal{P}$ w.r.t. I) denote the union of all unfounded sets for $\mathcal{P}$ w.r.t. I.*

These features are shared with disjunctive logic programs without aggregates, as discussed in [11]. However, while aggregate- and disjunction-free programs possess a unique GUS for arbitrary interpretations, Ex. 7 shows that this does not hold for disjunction-free programs with aggregates. By virtue of Thm. 2 and Thm. 10 of [10], which states that a unique GUS exists for nondisjunctive programs with monotone and antimonotone aggregates, we can infer that the presence of nonmonotone aggregates or disjunction and a non-unfounded-free interpretation is necessary to invalidate the existence of a unique GUS.

4 Unfounded Sets And Answer Sets

We will now use the notion of unfounded sets to characterize models and answer sets. We begin with models and show that the negative part of a model is an unfounded set and vice versa.

Theorem 4. *Given a total interpretation I and program $\mathcal{P}$, I^- is an unfounded set for $\mathcal{P}$ w.r.t. I iff I is a model of $\mathcal{P}$.*

Proof. $(\Rightarrow)$: For any rule, either (i) $H(r) \cap I^- = \emptyset$, or (ii) $H(r) \cap I^- \neq \emptyset$. If (i), then $H(r) \cap I \neq \emptyset$, i.e. the head is true and r is satisfied w.r.t. I. If (ii) then one of the conditions of Def. 1 must hold. If condition 1 holds, the body is false w.r.t. I and r is satisfied w.r.t. I. If condition 2 holds, a body literal is false w.r.t. $I \dot\cup \neg.I^- = I$, so it

coincides with condition 1. If condition 3 holds, $H(r) \cap I \neq \emptyset$, and therefore the rule is satisfied w.r.t. I. In total, if I^- is an unfounded set for $\mathcal{P}$ w.r.t. I, all rules are satisfied w.r.t. I, hence I is a model of $\mathcal{P}$.

($\Leftarrow$) : If I is a model, all rules are satisfied, so for any rule r, either (i) $H(r) \cap I \neq \emptyset$ or (ii) if $H(r) \cap I = \emptyset$ then a body literal l is false w.r.t. I. So also for any rule r with $H(r) \cap I^- \neq \emptyset$, either (i) or (ii) holds. If (i), then condition 3 of Def. 1 applies. If (ii), then condition 1 (and also condition 2, since $I \dot\cup \neg.I^- = I$) applies. Therefore I^- is an unfounded set. □

We now turn to answer sets. Each answer set is a model, so its negative part is an unfounded set. We can show that it is the greatest unfounded set. Conversely, if the negative part of a total interpretation is its greatest unfounded set, it is an answer set.

Theorem 5. *A total interpretation I is an answer set of $\mathcal{P}$ iff $I^- = GUS_{\mathcal{P}}(I)$*[4].

Proof. ($\Rightarrow$) : If I is an answer set, it is also a model of $\mathcal{P}$, so by Thm. 4, I^- is an unfounded set for $\mathcal{P}$ w.r.t. I. We next show that I is unfounded-free w.r.t. $\mathcal{P}$, from which $I^- = GUS_{\mathcal{P}}(I)$ follows. Let us assume an unfounded set X for $\mathcal{P}$ w.r.t. I exists such that $I \cap X \neq \emptyset$. We can show that then $I \dot\cup \neg.X$ is a model of $\mathcal{P}^I$, contradicting the fact that I is an answer set of $\mathcal{P}$.

First note that for any rule r in $\mathcal{P}^I$, all body literals are true w.r.t. I (by construction of P^I), and $H(r) \cap I \neq \emptyset$ (since I is a model of $\mathcal{P}^I$). We differentiate two cases: (i) $H(r) \cap (I \dot\cup \neg.X) \neq \emptyset$ and (ii) $H(r) \cap (I \dot\cup \neg.X) = \emptyset$. For (i), r is trivially satisfied by $I \dot\cup \neg.X$. For (ii), since we know $H(r) \cap I \neq \emptyset$, $H(r) \cap X \neq \emptyset$ must hold. Since X is an unfounded set w.r.t $\mathcal{P}$ and I (and $r \in \mathcal{P}$), a body literal of r must be false w.r.t. $I \dot\cup \neg.X$ (note that neither a body literal of r is false w.r.t. I since $r \in \mathcal{P}^I$, nor $(H(r) \setminus X) \cap I \neq \emptyset$ holds, otherwise $H(r) \cap (I \dot\cup \neg.X) \neq \emptyset$). So r is satisfied also in case (ii). $I \dot\cup \neg.X$ is therefore a model of $\mathcal{P}^I$, and since $(I \dot\cup \neg.X)^+ \subset I^+$, I is not a minimal model of $\mathcal{P}^I$, contradicting that I is an answer set of $\mathcal{P}$.

($\Leftarrow$) : By Thm. 4 if I^- is an unfounded set for $\mathcal{P}$ w.r.t. I, I is a model of $\mathcal{P}$, so it is also a model of $\mathcal{P}^I$. We show by contradiction that it is in fact a minimal model of $\mathcal{P}^I$.

Assume that a total interpretation J, where $J^+ \subset I^+$, is a model of $\mathcal{P}^I$. Since both J and I are total, $J^- \supset I^-$. Again by Thm. 4, J^- is an unfounded set for $\mathcal{P}^I$ w.r.t. J. We can then show that J^- is also an unfounded set for $\mathcal{P}$ w.r.t. I, contradicting the fact that I^- is $GUS_{\mathcal{P}}(I)$. For any rule in $\mathcal{P} \setminus \mathcal{P}^I$, a body literal is false w.r.t. I, so condition 1 of Def. 1 holds. For a rule $r \in \mathcal{P}^I$ such that $H(r) \cap J^- \neq \emptyset$, (a) a body literal of r is false w.r.t. J (note that $J \dot\cup \neg.J^- = J$) or (b) an atom a in $H(r) \setminus J^-$ is true w.r.t. J. Concerning (a), observe that $I \dot\cup \neg.J^- = J$ so (a) holds iff a body atom is false w.r.t. $I \dot\cup \neg.J^-$. Concerning (b), since $J^+ \subset I^+$, atom a is also true w.r.t. I. In total, we have shown that J^- is an unfounded set for $\mathcal{P}$ w.r.t. I, a contradiction to $I^- = GUS_{\mathcal{P}}(I)$. So I is indeed a minimal model of $\mathcal{P}^I$, and hence an answer set of $\mathcal{P}$. □

Since the existence of the GUS implies that the interpretation is unfounded-free, we obtain also:

Corollary 3. *A model I of a program $\mathcal{P}$ is unfounded-free iff I is an answer set of $\mathcal{P}$.*

[4] Note that by Def. 3, the existence of $GUS_{\mathcal{P}}(I)$ implies that I is unfounded-free.

5 Computational Complexity

We will now study the complexity involved with unfounded sets. In particular, we are interested in the question whether a total interpretation is unfounded-free, as by Cor. 3 this notion can be fruitfully used for computing answer sets. Throughout this section, we assume that the truth value of aggregates can be established in polynomial time, which is feasible for all aggregates currently available in ASP systems. If, however, aggregate truth valuation has a higher complexity, the total complexity will increase accordingly.

We first show membership for the full language, and then hardness for a restricted fragment, implying completeness for both languages and anything in between.

Theorem 6. *Given a ground disjunctive logic program $\mathcal{P}$, and a total interpretation I, deciding whether I is unfounded-free w.r.t. $\mathcal{P}$ is in co-NP.*

Proof. The complementary problem (deciding whether I is not unfounded-free) is in NP: Guess $X \subseteq B_{\mathcal{P}}$ and check that 1. X is an unfounded set for $\mathcal{P}$ w.r.t. I, and 2. that $X \cap I \neq \emptyset$. Both 1. and 2. are feasible in polynomial time, assuming that determining the truth value of an aggregate literal can be done in polynomial time. □

Next we show that deciding unfounded-freeness is a hard problem even for a simple class of programs, provided that nonmonotone aggregates may be present.

Theorem 7. *Given a ground nondisjunctive, negation-free logic program $\mathcal{P}$ with arbitrary aggregates, and a total interpretation I, deciding whether I is unfounded-free w.r.t. $\mathcal{P}$ is co-NP-hard.*

Proof. We give a reduction from the problem of unsatisfiability of a propositional 3CNF $\phi = (c_1^1 \vee c_2^1 \vee c_3^1) \wedge \ldots \wedge (c_1^m \vee c_2^m \vee c_3^m)$ where each c_j^i is a literal over one of n variables $V = \{x_1, \ldots, x_n\}$. We construct a program $\mathcal{P}(\phi)$:

$$
\begin{array}{lll}
x_1(1) \coloneq \#\mathtt{avg}\{X : x_1(X)\} = 0. & x_1(1) \coloneq w. & x_1(0). \ \ldots \ x_n(0). \\
x_1(-1) \coloneq \#\mathtt{avg}\{X : x_1(X)\} = 0. & x_1(-1) \coloneq w. & w \coloneq \rho(c_1^1), \rho(c_2^1), \rho(c_3^1). \\
\vdots & \vdots & \vdots \\
x_n(1) \coloneq \#\mathtt{avg}\{X : x_n(X)\} = 0. & x_n(1) \coloneq w. & w \coloneq \rho(c_1^m), \rho(c_2^m), \rho(c_3^m). \\
x_n(-1) \coloneq \#\mathtt{avg}\{X : x_n(X)\} = 0. & x_n(-1) \coloneq w. &
\end{array}
$$

where $\rho(x_i) = x_i(1)$ and $\rho(\neg x_i) = x_i(-1)$. Then ϕ is unsatisfiable iff the interpretation $I(\phi) = \{w, x_1(1), x_1(0), x_1(-1), \ldots, x_n(1), x_n(0), x_n(-1)\}$ is unfounded-free.

Indeed, if σ is a satisfying truth assignment for V, then $X^\sigma = \{w\} \cup \{x_i(1) \mid x_i \text{ true in } \sigma\} \cup \{x_i(-1) \mid x_i \text{ false in } \sigma\}$ is unfounded for $\mathcal{P}(\phi)$ w.r.t. $I(\phi)$. It is easily checked that for each rule in $\mathcal{P}(\phi)$ with a head in X^σ at least one body literal is false w.r.t. $I(\phi) \dot\cup \neg.X^\sigma$.

On the other hand, let X be a non-empty unfounded set for $\mathcal{P}(\phi)$ w.r.t. $I(\phi)$. Clearly, $x_i(0) \notin X$. If $x_i(1) \in X$, then $x_i(-1) \notin X$ and vice versa, because if both $x_i(1) \in X$ and $x_i(-1) \in X$, $\#\mathtt{avg}\{X : x_i(X)\} = 0$ is true w.r.t. $I(\phi)$ and $I(\phi) \dot\cup \neg.X$. Furthermore, if some $x_i(1) \in X$ or $x_i(-1) \in X$, then also $w \in X$. If $w \in X$, then for each clause in ϕ some corresponding $x_j(1)$ or $x_j(-1)$ must be in X. It is easy to see that

this corresponds to a (possibly) partial truth assignment satisfying ϕ, which can always be extended to a total truth assignment satisfying ϕ. So any non-empty unfounded set X (hence $X \cap I(\phi) \neq \emptyset$) implies the existence of a satisfying truth assignment for ϕ. □

These results allow us to give a complete picture of the complexity of model checking, reported on the left of Table 1. There, the rows indicate the kinds of aggregates (m – monotone, a – antimonotone, n – nonmonotone) allowed in programs, while the columns vary over the presence of negation and disjunction. All co-NP entries are completeness results. The results in the first row are well-known results of the literature (cf. [19]), the P entries for $\{m, a\}$ follow from recent results in [10], while the other results are consequences of Thms. 5, 6, 7, Cor. 3, with results from the literature.

It becomes clear from this table that a complexity increase occurs with the presence of either disjunction or nonmonotone aggregates, and, importantly, that these two factors together do not cause a further increase. Also in Table 1, on the right hand side, we have summarized results from the literature (see [19, 8, 10]) for the problem of cautious reasoning. We observe that the complexity increase occurs at the same places, and indeed one can blame the necessity of co-NP checks for the Π_2^P results.

Concerning computation, we conjecture that, given the symmetries in properties and complexity, techniques analogous to those described in [11] can be used in order to effectively and efficiently compute answer sets of programs with arbitrary aggregates.

We will briefly discuss an important issue concerning computation, though. It is striking that from Table 1 it appears that a single, apparently "innocent", aggregate like $\#\texttt{count}\{< 1 : a >, < 2 : b >\} = 1$ will increase the reasoning complexity. Obviously, this is not the case, as this aggregate can be rewritten to an equivalent conjunction $\#\texttt{count}\{< 1 : a >, < 2 : b >\} \leq 1$, $\#\texttt{count}\{< 1 : a >, < 2 : b >\} \geq 1$, thus eliminating the nonmonotone aggregate. In fact, such a decomposition is possible for each nonmonotone aggregate, if one allows the use of custom aggregates (rather than a set of fixed aggregates) and the introduction of new symbols. However, this operation is only polynomial (and hence effective) if the number of the truth value changes of the nonmonotone aggregate in the lattice of total interpretations induced by $<$ is polynomially bounded. Note that all currently implemented nonmonotone aggregates of DLV ($\#\texttt{avg}$is not) and Smodels are polynomially decomposable.

6 Related Work

To our knowledge, the only other works in which the notion of unfounded set has been defined and studied for programs with aggregates are [1, 10]. However, both works consider only nondisjunctive programs, and the latter restricts itself to monotone and antimonotone aggregates. As discussed in [10], the definition of [1] seems to ignore aggregates at crucial points of the definition, and appears to be incomparable with the one in [10], and therefore also with Def. 1. Unfounded sets for disjunctive (aggregate-free) programs had been defined and studied in [11]. In fact, several of our results parallel those of [11]. We believe that for this reason the computational techniques reported therein can be adapted to the aggregate setting.

Since unfounded sets have originally been used for defining the well-founded semantics, one could do this also with our unfounded sets. This was done (to some ex-

Checking	$\emptyset$	{not}	$\{\vee\}$	{not, $\vee$}
$\emptyset$	P	P	co-NP	co-NP
$\{m, a\}$	P	P	co-NP	co-NP
$\{m, a, n\}$	co-NP	co-NP	co-NP	co-NP

Cautious	$\emptyset$	{not}	$\{\vee\}$	{not, $\vee$}
$\emptyset$	P	co-NP	Π_2^P	Π_2^P
$\{m, a\}$	co-NP	co-NP	Π_2^P	Π_2^P
$\{m, a, n\}$	Π_2^P	Π_2^P	Π_2^P	Π_2^P

Table 1. Complexity of Answer Set Checking (left) and Cautious Reasoning (right)

tent) in [11], but the recent work in [20] argues that for disjunctive programs a somewhat refined version of unfounded sets based on so-called model sets rather than interpretations. Recently, a unifying framework for unfounded sets and loop formulas has been defined in in [15]. Also this work does not consider aggregates, but we believe that the results with aggregates should be generalizable in a similar way.

Concerning semantics for programs with aggregates, especially the last few years have seen many proposals. We refer to [8] (the definition on which our work is based) and [21] for overviews and comparisons.

7 Conclusion and Future Work

The semantics of logic programs with aggregates is not straightforward, especially in presence of recursive aggregates. The characterizations of answer sets, provided in Sec. 4, allow for a better understanding of the meaning of programs with aggregates. Our results give confidence in the appropriateness of answer sets as defined in [8].

Furthermore, our results provide a handle on effective methods for computing answer sets for disjunctive programs with (possibly recursive and nonmonotone) aggregates. An approach with a separation of model generation and model checking (which is co-NP in the worst case) is indicated by the complexity results of Sec. 5. By defining suitable operators analogously to [11], one can obtain powerful means for pruning in the generation phase, along with an effective instrument for model checking, as described in [13, 14]. Our results should also be adaptable to be used for SAT-based ASP systems, all of which rely on loop formulas, along the lines described in [15].

Our complexity results provide a clear picture of the various program fragments from the computational viewpoint. This is very useful for picking the appropriate techniques to be employed for the computation. In particular, it became clear that in the presence of only monotone and antimonotone aggregates and absence of disjunctions, an NP computing scheme can be chosen. That is, there the focus should be on answer set generation, while answer set checking is a simpler task. As soon as nonmonotone aggregates or disjunctions are present, a two-level schema has to be employed, which must focus on both answer set generation and checking, as both tasks are hard. Importantly, the presence of both nonmonotone aggregates and disjunction does not further raise the complexity. It should be noted that, as pointed out at the end of Sec. 5, many nonmonotone aggregates can be decomposed into monotone and antimonotone ones, including Smodels cardinality and weight constraints with positive weights.

A main concern for future work is therefore the exploitation of our results for the implementation of recursive aggregates in ASP systems, along with a study on how to generalize the notion for defining the well-founded semantics for the full language.

References

1. Kemp, D.B., Stuckey, P.J.: Semantics of Logic Programs with Aggregates. In: ISLP'91, MIT Press (1991) 387–401
2. Denecker, M., Pelov, N., Bruynooghe, M.: Ultimate Well-Founded and Stable Model Semantics for Logic Programs with Aggregates. In Codognet, P., ed.: ICLP-2001, (2001) 212–226
3. Gelfond, M.: Representing Knowledge in A-Prolog. In: Computational Logic. Logic Programming and Beyond. LNCS 2408 (2002) 413–451
4. Simons, P., Niemelä, I., Soininen, T.: Extending and Implementing the Stable Model Semantics. Artificial Intelligence **138** (2002) 181–234
5. Dell'Armi, T., Faber, W., Ielpa, G., Leone, N., Pfeifer, G.: Aggregate Functions in DLV. In: ASP'03, Messina, Italy (2003) 274–288 Online at `http://CEUR-WS.org/Vol-78/`.
6. Pelov, N., Truszczyński, M.: Semantics of disjunctive programs with monotone aggregates - an operator-based approach. In: NMR 2004. (2004) 327–334
7. Pelov, N., Denecker, M., Bruynooghe, M.: Partial stable models for logic programs with aggregates. In: LPNMR-7. LNCS 2923
8. Faber, W., Leone, N., Pfeifer, G.: Recursive aggregates in disjunctive logic programs: Semantics and complexity. In: JELIA 2004. LNCS 3229
9. Ferraris, P.: Answer Sets for Propositional Theories. `http://www.cs.utexas.edu/users/otto/papers/proptheories.ps` (2004)
10. Calimeri, F., Faber, W., Leone, N., Perri, S.: Declarative and Computational Properties of Logic Programs with Aggregates. In: Nineteenth International Joint Conference on Artificial Intelligence (IJCAI-05). (2005) 406–411
11. Leone, N., Rullo, P., Scarcello, F.: Disjunctive Stable Models: Unfounded Sets, Fixpoint Semantics and Computation. Information and Computation **135** (1997) 69–112
12. Calimeri, F., Faber, W., Leone, N., Pfeifer, G.: Pruning Operators for Answer Set Programming Systems. In: NMR'2002. (2002) 200–209
13. Koch, C., Leone, N., Pfeifer, G.: Enhancing Disjunctive Logic Programming Systems by SAT Checkers. Artificial Intelligence **15** (2003) 177–212
14. Pfeifer, G.: Improving the Model Generation/Checking Interplay to Enhance the Evaluation of Disjunctive Programs. In: LPNMR-7. LNCS, (2004) 220–233
15. Lee, J.: A Model-Theoretic Counterpart of Loop Formulas. `http://www.cs.utexas.edu/users/appsmurf/papers/mtclf.pdf` (2004)
16. Baral, C.: Knowledge Representation, Reasoning and Declarative Problem Solving. CUP (2002)
17. Gelfond, M., Lifschitz, V.: Classical Negation in Logic Programs and Disjunctive Databases. NGC **9** (1991) 365–385
18. Van Gelder, A., Ross, K., Schlipf, J.: The Well-Founded Semantics for General Logic Programs. JACM **38** (1991) 620–650
19. Dantsin, E., Eiter, T., Gottlob, G., Voronkov, A.: Complexity and Expressive Power of Logic Programming. ACM Computing Surveys **33** (2001) 374–425
20. Wang, K., Zhou, L.: Comparisons and Computation of Well-founded Semantics for Disjunctive Logic Programs. ACM TOCL **6** (2005)
21. Pelov, N.: Semantics of Logic Programs with Aggregates. PhD thesis, Katholieke Universiteit Leuven, Leuven, Belgium (2004)