Declarative and Computational Properties of Logic Programs with Aggregates*

Francesco Calimeri and **Wolfgang Faber**† and **Nicola Leone** and **Simona Perri**
Department of Mathematics, University of Calabria, I-87030 Rende (CS), Italy
Email: {calimeri,faber,leone,perri}@mat.unical.it

Abstract

We investigate the properties of logic programs with aggregates. We mainly focus on programs with monotone and antimonotone aggregates ($\mathrm{LP}^{\mathcal{A}}_{m,a}$ programs). We define a new notion of unfounded set for $\mathrm{LP}^{\mathcal{A}}_{m,a}$ programs, and prove that it is a sound generalization of the standard notion of unfounded set for aggregate-free programs. We show that the answer sets of an $\mathrm{LP}^{\mathcal{A}}_{m,a}$ program are precisely its unfounded-free models.

We define a well-founded operator $\mathcal{W}_\mathcal{P}$ for $\mathrm{LP}^{\mathcal{A}}_{m,a}$ programs; we prove that its total fixpoints are precisely the answer sets of $\mathcal{P}$, and its least fixpoint $\mathcal{W}^\omega_\mathcal{P}(\emptyset)$ is contained in the intersection of all answer sets (if $\mathcal{P}$ admits an answer set). $\mathcal{W}^\omega_\mathcal{P}(\emptyset)$ is efficiently computable, and for aggregate-free programs it coincides with the well-founded model.

We carry out an in-depth complexity analysis in the general framework, including also nonmonotone aggregates. We prove that monotone and antimonotone aggregates do not increase the complexity of cautious reasoning, which remains in co-NP. Nonmonotone aggregates, instead, do increase the complexity by one level in the polynomial hierarchy. Our results allow also to generalize and speed-up ASP systems with aggregates.

1 Introduction

The introduction of aggregates atoms [Kemp and Stuckey, 1991; Denecker *et al.*, 2001; Gelfond, 2002; Simons *et al.*, 2002; Dell'Armi *et al.*, 2003; Pelov and Truszczyński, 2004; Pelov *et al.*, 2004] 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, only few works have focused on logic programs with aggregates; their behaviour, their semantic properties, and their computational features are still far from being fully clarified. A recent proposal for answer set semantics is receiving a consensus [Faber *et al.*, 2004]. However, unfounded sets and the well-founded operator [Van Gelder *et al.*, 1991], which are important for both the characterization and the computation of standard LPs [Leone *et al.*, 1997; Simons *et al.*, 2002; Calimeri *et al.*, 2002; Koch *et al.*, 2003; Pfeifer, 2004], have not been defined in a satisfactory manner for logic programs with aggregates. Moreover, the impact of aggregates on the computational complexity of the reasoning tasks has not been analyzed in depth.

This paper makes a first step to overcome this deficiency, improving the characterization of programs with aggregates, for both declarative and computational purposes. The main contributions of the paper are as follows.

• We define the notion of unfounded set for logic programs with both monotone and antimonotone aggregates ($\mathrm{LP}^{\mathcal{A}}_{m,a}$ programs). This notion is a sound generalization of the concept of unfounded sets previously given for programs without aggregates. We show that our definition coincides with the original definition of unfounded sets of [Van Gelder *et al.*, 1991] on the class of normal (aggregate-free) programs, and shares its nice properties (like, e.g., the existence of the greatest unfounded set).

• We provide a declarative characterization of answer sets in terms of unfounded sets. In particular, answer sets are precisely unfounded-free models of an $\mathrm{LP}^{\mathcal{A}}_{m,a}$ program.

• We define a well-founded operator $\mathcal{W}_\mathcal{P}$ for logic programs with aggregates, which extends the classical well-founded operator [Van Gelder *et al.*, 1991]. The total fixpoints of $\mathcal{W}_\mathcal{P}$ are exactly the answer sets of $\mathcal{P}$, and its least fixpoint $\mathcal{W}^\omega_\mathcal{P}(\emptyset)$ is contained in the intersection of all answer sets. Importantly, $\mathcal{W}^\omega_\mathcal{P}(\emptyset)$ is polynomial-time computable.

• We analyze the complexity of logic programs with arbitrary (also nonmonotone) aggregates and fragments thereof. Both monotone and antimonotone aggregates do not affect the complexity of answer set semantics, which remains co-NP-complete (for cautious reasoning). Nonmonotone aggregates, instead, do increase the complexity, jumping to the second level of the polynomial hierarchy (Π^P_2).

For space limitations, some proofs are sketched.

2 Logic Programs with Aggregates

In this section, we recall syntax, semantics, and some basic properties of logic programs with aggregates.

*This work was partially supported by the EC under projects IST-2002-33570 INFOMIX and IST-2001-37004 WASP, and by FWF under projects P16536-N04 and P17212-N04.

†Funded by APART of the Austrian Academy of Sciences.

2.1 Syntax

We assume that the reader is familiar with standard LP; we refer to atoms, literals, rules, and programs of LP, as *standard atoms, standard literals, standard rules*, and *standard programs,* respectively. Two literals are said to be complementary if they are of the form p and $\texttt{not}\ p$ 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 [Baral, 2002; Gelfond and Lifschitz, 1991].

Set Terms. A (LP$^{\mathcal{A}}$) *set term* is either a symbolic set or a ground set. A *symbolic set* is a pair $\{Vars : Conj\}$, 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 $\langle \bar{t} : Conj \rangle$, where $\bar{t}$ 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, undefined for empty set), $\#\texttt{max}$ (maximum term, undefined for empty set)[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 in DLV notation, where the latter contains a ground set and could be a ground instance of the former:

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

An *atom* is either a standard (LP) atom or an aggregate atom. A *literal* L is an atom A or an atom A preceded by the default negation symbol $\texttt{not}$; if A is an aggregate atom, L is an *aggregate literal*.

LP$^{\mathcal{A}}$ Programs. A *(*LP$^{\mathcal{A}}$*) rule* r is a construct

$$a :- b_1, \cdots, b_k, \texttt{not}\ b_{k+1}, \cdots, \texttt{not}\ b_m.$$

where a is a standard atom, $b_1, \cdots, b_m$ are atoms, and $m \geq k \geq 0$. The atom a 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 head atom by $H(r)$, and the set $\{b_1, ..., b_k, \texttt{not}\ b_{k+1}, ..., \texttt{not}\ b_m\}$ of the body literals by $B(r)$.

[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., $\{X \mid \exists Y s.t.\ a(X, Y), p(Y)\ is\ true\}$.

[2]The first two aggregates correspond, respectively, to the cardinality and weight constraint literals of Smodels.

A *(*LP$^{\mathcal{A}}$*) program* is a set of LP$^{\mathcal{A}}$ rules. A *global* variable of a rule r is a variable appearing in a standard atom of r; 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 LP$^{\mathcal{A}}$ programs are safe.

2.2 Answer Set Semantics

Universe and Base. Given a LP$^{\mathcal{A}}$ 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 $inst(S)$:
$\{\langle \gamma(Vars) : \gamma(Conj) \rangle \mid \gamma\ \textit{is a local substitution for}\ S\}$.[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 $\mathcal{P}$.

Example 4 Consider the following program $\mathcal{P}_1$:

$$q(1) \vee p(2, 2). \qquad q(2) \vee p(2, 1).$$
$$t(X) :- q(X), \#\texttt{sum}\{Y : p(X, Y)\} > 1.$$

The instantiation $Ground(\mathcal{P}_1)$ is the following:

$$q(1) \vee p(2, 2). t(1) :- q(1), \#\texttt{sum}\{\langle 1 : p(1, 1)\rangle, \langle 2 : p(1, 2)\rangle\} > 1.$$
$$q(2) \vee p(2, 1). t(2) :- q(2), \#\texttt{sum}\{\langle 1 : p(2, 1)\rangle, \langle 2 : p(2, 2)\rangle\} > 1.$$

Interpretations. An *interpretation* for a LP$^{\mathcal{A}}$ 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 or false w.r.t I then it is undefined w.r.t I. We denote by I^+ (resp. I^-) the set of all standard positive (resp. negative) literals occurring in I. We also denote with $\bar{I}$ the set of undefined standard literals w.r.t I. An interpretation I is *total* if $\bar{I}$ is empty (i.e., $I^+ \cup \neg.I^- = B_{\mathcal{P}}$), otherwise I is *partial*.

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

An interpretation also provides a meaning for aggregate literals. Their truth value is first defined for total interpretations, and then induced for 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. 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 $\mathcal{P}$).[4]

An instantiated aggregate atom $A = f(S) \prec k$ 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.

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 w.r.t. J); otherwise it is undefined.

Example 5 Consider the atom $A = \#\texttt{sum}\{\langle 1 : p(2,1)\rangle, \langle 2 : p(2,2)\rangle\} > 1$ from Example 4. 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 $\texttt{not}\ p(2,1)$. Therefore, either $I(S) = [2]$ or $I(S) = [1,2]$ and the application of $\#\texttt{sum}$ yields either $2 > 1$ or $3 > 1$, hence A is true w.r.t. I.

Minimal Models. Given an interpretation I, a rule r is *satisfied w.r.t.* I if the 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 $\mathrm{LP}^{\mathcal{A}}$ 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 for $\mathrm{LP}^{\mathcal{A}}$ programs from [Faber *et al.*, 2004].

Definition 1 ([Faber *et al.*, 2004]) Given a ground $\mathrm{LP}^{\mathcal{A}}$ 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 6 Consider the following two programs:

$$P_1 : \{p(a) {:-} \#\texttt{count}\{X : p(X)\} > 0.\}$$
$$P_2 : \{p(a) {:-} \#\texttt{count}\{X : p(X)\} < 1.\}$$

$Ground(P_1) = \{p(a) {:-} \#\texttt{count}\{\langle a : p(a)\rangle\} > 0.\}$ and $Ground(P_2) = \{p(a) {:-} \#\texttt{count}\{\langle a : p(a)\rangle\} < 1.\}$, and interpretation $I_1 = \{p(a)\}$, $I_2 = \emptyset$. Then, $Ground(P_1)^{I_1} = Ground(P_1)$, $Ground(P_1)^{I_2} = \emptyset$, 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 (because 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.

Example 7 All ground instances of $\#\texttt{count}\{Z : r(Z)\} > 1$ and $\texttt{not}\ \#\texttt{count}\{Z : r(Z)\} < 1$ are monotone, while for $\#\texttt{count}\{Z : r(Z)\} < 1$, and $\texttt{not}\ \#\texttt{count}\{Z : r(Z)\} > 1$ they are antimonotone.

We denote by $\mathrm{LP}^{\mathcal{A}}_m$ ($\mathrm{LP}^{\mathcal{A}}_a$/$\mathrm{LP}^{\mathcal{A}}_n$) the fragment of $\mathrm{LP}^{\mathcal{A}}$ where only monotone (antimonotone/nonmonotone) aggregates are allowed; $\mathrm{LP}^{\mathcal{A}}_{m,a}$ allows for both monotone and antimonotone aggregates.

Remark 1 Observe that 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.

3 Unfounded Sets

In this section, we extend the notion of unfounded sets, given in [Van Gelder *et al.*, 1991] for aggregate-free programs, to the framework of $\mathrm{LP}^{\mathcal{A}}_{a,m}$ programs. Let us denote by $S1 \,\dot\cup\, \neg.S2$ the set $(S_1 - S_2) \cup \neg.S_2$, where S_1 and S_2 are sets of standard ground literals.

Definition 2 (Unfounded Set) A set X of ground atoms is an unfounded set for an $\mathrm{LP}^{\mathcal{A}}_{a,m}$ program $\mathcal{P}$ w.r.t. I if, for each rule $r \in \mathcal{P}$ having the head atom belonging to X, at least one of the following conditions holds: 1. some antimonotone body literal of r is false w.r.t. I, and 2. some monotone body literal of r is false w.r.t. $I \,\dot\cup\, \neg.X$.

Example 8 Consider the following program P:

$r_1:\ a(1) {:-} \#\texttt{count}\{\langle 1 : a(1)\rangle, \langle 2 : a(2)\rangle, \langle 3 : a(3)\rangle\} > 2.$
$r_2:\ a(2).$
$r_3:\ a(3) {:-} \#\texttt{count}\{\langle 1 : a(1)\rangle, \langle 2 : a(2)\rangle, \langle 3 : a(3)\rangle\} > 2.$

[4] In this paper, we assume that the value of an aggregate function can be computed in time polynomial in the size of the input multiset.

and $I = \{a(1), a(2), a(3)\}$. Then $X = \{a(1)\}$ is an unfounded set w.r.t. $I, \mathcal{P}$, since condition 2 holds for r_1. Also $\{a(3)\}$, and $\{a(1), a(3)\}$ are unfounded.

We can show that Definition 2 generalizes the one given in [Van Gelder *et al.*, 1991] for aggregate-free programs.

Theorem 9 For an aggregate-free $\text{LP}^{\mathcal{A}}_{a,m}$ program $\mathcal{P}$, Definition 2 is equivalent to the one of [Van Gelder *et al.*, 1991].

Thus, Definition 2 is an alternative characterization of unfounded sets for standards literals. In fact, while condition 1 of Definition 2 does not exactly cover the first one in [Van Gelder *et al.*, 1991], condition 2 catches all cases of the second in [Van Gelder *et al.*, 1991] and those "lost" by condition 1. This separates positive and negative literals, allowing to distinguish between the behavior of monotone and antimonotone literals.

Theorem 10 If X_1 and X_2 are unfounded sets w.r.t. an interpretation I for a $\text{LP}^{\mathcal{A}}_{a,m}$ program $\mathcal{P}$, then also $X_1 \cup X_2$ is an unfounded set w.r.t. I for $\mathcal{P}$.

Proof sketch. For Condition 2 of Def. 2, observe that $I \dot\cup \neg.(X_1 \cup X_2) \leq I \dot\cup \neg.X_1$. Therefore, if a monotone literal ℓ is false w.r.t. $I \dot\cup \neg.X_1$, then it is false w.r.t. $I \dot\cup \neg.(X_1 \cup X_2)$. Symmetrically for X_2. □

By virtue of Theorem 10, the union of all unfounded sets for $\mathcal{P}$ w.r.t. I is an unfounded set. We call it the *Greatest Unfounded Set* of $\mathcal{P}$ w.r.t. I, and denote it by $GUS_{\mathcal{P}}(I)$.

Proposition 1 Let I and J be interpretations for an $\text{LP}^{\mathcal{A}}_{a,m}$ program $\mathcal{P}$. If $I \subseteq J$, then $GUS_{\mathcal{P}}(I) \subseteq GUS_{\mathcal{P}}(J)$.

Proof sketch. Follows from Remark 1, since $I \dot\cup \neg.X \subseteq J \dot\cup \neg.X$. □

4 Answer Sets and Unfounded Sets

In this section, we provide a couple of characterizations of answer sets in terms of unfounded sets.

Definition 3 (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.

The next lemma gives an equivalent characterization of the unfounded-free property for total interpretations.

Lemma 11 Let I be a total interpretation for a program $\mathcal{P}$. I is unfounded-free iff no nonempty set of atoms contained in I is an unfounded set for $\mathcal{P}$ w.r.t.I.

Proof sketch. If I is not unfounded-free, an unfounded set X for $\mathcal{P}$ w.r.t. I s.t. $X \cap I \neq \emptyset$ exists. Then, $X \cap I$ is also an unfounded set. □

Theorem 12 A model M is an answer set of an $\text{LP}^{\mathcal{A}}_{a,m}$ program $\mathcal{P}$ if and only if M is unfounded-free.

Proof sketch. If M is a model and $X \subseteq M$ is a non-empty unfounded set w.r.t. M, then it can be shown that $M \dot\cup \neg.X$ is a model of $\mathcal{P}^M$, hence M is no answer set. On the other hand, if M is an unfounded-free model but not an answer set, a model N of $\mathcal{P}^M$ s.t. $N^+ \subset M^+$ must exist. We can show that $M^+ - N^+$ is an unfounded set. □

Now we give another interesting characterization of answer sets. A total interpretation is an answer set if and only if its false literals are unfounded.

Lemma 13 A total interpretation M is a model for $\mathcal{P}$ iff $\neg.M^-$ is an unfounded set for $\mathcal{P}$ w.r.t. M.

Proof sketch. The result follows from the fact that $M = M \dot\cup \neg.(\neg.M^-)$. □

Theorem 14 Let M be a total interpretation for a program $\mathcal{P}$. M is an answer set iff $\neg.M^- = GUS_{\mathcal{P}}(M)$.

Proof sketch. Can be shown using Lemmata 13 and 11 and Theorem 12. □

5 Well-Founded Semantics

In this section we extend the $\mathcal{W}_{\mathcal{P}}$ defined in [Van Gelder *et al.*, 1991] for aggregate-free programs to $\text{LP}^{\mathcal{A}}_{a,m}$ programs. Then, we show that the answer sets of an $\text{LP}^{\mathcal{A}}_{a,m}$ program $\mathcal{P}$ coincide exactly with the total fixpoints of $\mathcal{W}_{\mathcal{P}}$.

We start by providing an extension to $\text{LP}^{\mathcal{A}}_{a,m}$ programs of the immediate consequence operator $T_{\mathcal{P}}$ defined in [Van Gelder *et al.*, 1991] for three-valued interpretations of standard logic programs.

Definition 4 Let $\mathcal{P}$ be a $\text{LP}^{\mathcal{A}}_{a,m}$ program. Define the operators $\mathcal{T}_{\mathcal{P}}$ and $\mathcal{W}_{\mathcal{P}}$ from $2^{B_{\mathcal{P}} \cup \neg.B_{\mathcal{P}}}$ to $2^{B_{\mathcal{P}}}$ and $2^{B_{\mathcal{P}} \cup \neg.B_{\mathcal{P}}}$, respectively, as follows.

$$\begin{aligned} \mathcal{T}_{\mathcal{P}}(I) = \{a \in B_{\mathcal{P}} \mid \exists r \in Ground(\mathcal{P}) \; s.t. \; a = H(r) \\ and \; B(r) \; is \; true \; w.r.t. \; I\} \\ \mathcal{W}_{\mathcal{P}}(I) = \mathcal{T}_{\mathcal{P}}(I) \cup \neg.GUS_{\mathcal{P}}(I). \end{aligned}$$

Theorem 15 Let M be a total interpretation for a program $\mathcal{P}$. M is an answer set for $\mathcal{P}$ iff M is a fixpoint of $\mathcal{W}_{\mathcal{P}}$.

Proof sketch. $M^- = \neg.GUS_{\mathcal{P}}(M)$ holds by virtue of Theorem 14, $M^+ = \mathcal{T}_{\mathcal{P}}(M)$ can be shown using Lemma 11 and Definition 2. □

The $\mathcal{W}_{\mathcal{P}}$ operator is clearly monotone on a meet semilattice, and it therefore admits a least fixpoint [Tarski, 1955]. This fixpoint can be computed iteratively starting from the empty set, and approximates the intersection of all answer sets (if any).

Theorem 16 Given an $\text{LP}^{\mathcal{A}}_{m,a}$ program $\mathcal{P}$, let $\{W_n\}_{n \in \mathcal{N}}$ be the sequence whose n-th term is the n-fold application of the $\mathcal{W}_{\mathcal{P}}$ operator on the empty set (i.e., $W_0 = \emptyset$, $W_n = \mathcal{W}_{\mathcal{P}}(W_{n-1})$). Then (a) $\{W_n\}_{n \in \mathcal{N}}$ converges to a limit $\mathcal{W}^{\omega}_{\mathcal{P}}(\emptyset)$, and (b) for each answer set M for $\mathcal{P}$, $M \supseteq \mathcal{W}^{\omega}_{\mathcal{P}}(\emptyset)$.

Proof sketch. (a) follows from the monotonicity of $\mathcal{W}_{\mathcal{P}}$ and the finiteness of $B_{\mathcal{P}}$. (b) holds since all atoms computed by $\mathcal{T}_{\mathcal{P}}$ belong to any answer set and because of Theorem 14. □

From Theorem 15 and 16, the following easily follows.

Corollary 17 *If $\mathcal{W}^{\omega}_{\mathcal{P}}(\emptyset)$ is a total interpretation, then it is the unique answer set of $\mathcal{P}$.*

The following proposition confirms the intuition that Definition 4 extends the $\mathcal{W}_{\mathcal{P}}$ operator, as defined in [Van Gelder *et al.*, 1991] for standard programs , to $\text{LP}^{\mathcal{A}}_{m,a}$ programs.

Proposition 2 Let $\mathcal{P}$ be an aggregate-free program. Then the $\mathcal{W}_{\mathcal{P}}$ operator of Definition 4 exactly coincides with $\mathcal{W}_{\mathcal{P}}$ operator defined in [Van Gelder *et al.*, 1991].

Corollary 18 On aggregate-free programs, $\mathcal{W}^{\omega}_{\mathcal{P}}(\emptyset)$ coincides with the well-founded model of [Van Gelder *et al.*, 1991].

Moreover, there are simple cases where $\mathcal{W}_{\mathcal{P}}^{\omega}(\emptyset)$ captures the intended meaning of the program.

Example 19 Consider the following program $\mathcal{P}$:

$$a(1) :- \#\mathtt{sum}\{\langle 1 : a(1)\rangle, \langle 2 : a(2)\rangle\} > 1.$$
$$b :- \mathtt{not}\ a(1). \quad a(2) :- b. \quad b :- \mathtt{not}\ c.$$

We have $\mathcal{W}_{\mathcal{P}}(\emptyset) = \{\mathtt{not}\ c\}$, then $\mathcal{W}_{\mathcal{P}}(\{\mathtt{not}\ c\}) = \{b, \mathtt{not}\ c, \mathtt{not}\ a(1)\}$, then $\mathcal{W}_{\mathcal{P}}(\{b, \mathtt{not}\ c, \mathtt{not}\ a(1)\}) = \{b, \mathtt{not}\ c, \mathtt{not}\ a(1), a(2)\} = \mathcal{W}_{\mathcal{P}}^{\omega}(\emptyset)$.
It is easy to verify that $\mathcal{W}_{\mathcal{P}}^{\omega}(\emptyset)$ (which here is total) is an answer set for $\mathcal{P}$.

6 Computational Complexity

We first show the tractability of the well-founded semantics for $\mathrm{LP}_{m,a}^{\mathcal{A}}$, and we then analyze the complexity of answer set semantics for general $\mathrm{LP}^{\mathcal{A}}$ programs. We consider the propositional case, hence, throughout this section we assume that programs are ground.

Theorem 20 Given a ground $\mathrm{LP}_{m,a}^{\mathcal{A}}$ program $\mathcal{P}$: 1. The greatest unfounded set $GUS_{\mathcal{P}}(I)$ of $\mathcal{P}$ w.r.t. a given interpretation I is polynomial-time computable; 2. $\mathcal{W}_{\mathcal{P}}^{\omega}(\emptyset)$ is polynomial-time computable.

Proof sketch. We define an operator Φ_I from $B_{\mathcal{P}}$ to $B_{\mathcal{P}}$ as follows: $\Phi_I(Y) = \{a \mid \exists\ r \in \mathcal{P}$ with $a = H(r)$ s.t. no antimonotone body literal of r $false\ w.r.t.\ I\ \wedge$ all monotone body literals of r are $true\ w.r.t.\ Y - \neg.I^-\}$. The sequence $\phi_0 = \emptyset$, $\phi_k = \Phi_I(\phi_{k-1})$ is monotonically increasing and converges finitely to a limit ϕ_λ, for which $\phi_\lambda = B_{\mathcal{P}} - GUS_{\mathcal{P}}(I)$ can be shown. Furthermore, each application of Φ_I is polynomial in our setting[5], and also λ is polynomial in $|B_{\mathcal{P}}|$. From this, the result follows. □

This result has a positive impact also for the computation of the answer set semantics of logic programs with aggregates. Indeed, as stated in Theorem 16, $\mathcal{W}_{\mathcal{P}}^{\omega}(\emptyset)$ approximates the intersection of all answer sets from the bottom, and can be therefore used to efficiently prune the search space.

We next analyze the complexity of the answer set semantics of general $LP^{\mathcal{A}}$ programs. In [Faber *et al.*, 2004], the authors have shown that arbitrary (including nonmonotone) aggregates do not increase the complexity of disjunctive programs. However, nonmonotone aggregates do increase the complexity of reasoning on Or-free programs.[6]

Theorem 21 *Cautious reasoning over* $\mathrm{LP}_{m,a,n}^{\mathcal{A}}$ *is* Π_2^P*-complete.*

Proof. Membership follows directly from the results in [Faber *et al.*, 2004]. Concerning hardness, we provide a reduction from 2QBF. Let $\Psi = \forall x_1, \ldots, x_m \exists y_1, \ldots, y_n : \Phi$, where w.l.o.g. Φ is a propositional formula in 3CNF format, over precisely the variables $x_1, \ldots, x_m, y_1, \ldots, y_n$. Observe that Ψ is equivalent to $\neg\exists x_1, \ldots, x_m \forall y_1, \ldots, y_n : \neg\Phi$, and that $\neg\Phi$ is equivalent to a 3DNF where every literal has reversed polarity w.r.t. Φ. Let the $\mathrm{LP}_{m,a,n}^{\mathcal{A}}$ program Π^Ψ be:

$$t(x_i, 1) : -\#\mathtt{sum}\{V : t(x_i, V)\} >= 0.$$
$$t(x_i, -1) : -\#\mathtt{sum}\{V : t(x_i, V)\} <= 0.$$
$$t(y_i, 1) : -\#\mathtt{sum}\{V : t(y_i, V)\} >= 0.$$
$$t(y_i, -1) : -\#\mathtt{sum}\{V : t(y_i, V)\} <= 0.$$
$$t(y_i, 1) : -unsat. \quad t(y_i, -1) : -unsat.$$

For each clause $c_i = l_{i,1} \vee l_{i,2} \vee l_{i,3}$ of the original Φ, we add: $unsat : -\mu(l_{i,1}), \mu(l_{i,2}), \mu(l_{i,3})$, where $\mu(l)$ is $t(l, -1)$ if l is positive, and $t(l, 1)$ otherwise.

The query $\mathtt{not}\ unsat?$ holds for Π^Ψ, iff Ψ is satisfiable. □

Monotone and antimonotone aggregates, instead, behave as in the disjunctive case, not causing any complexity gap.

Theorem 22 *Cautious reasoning over* $\mathrm{LP}_{m,a}^{\mathcal{A}}$ *is co-NP-complete.*

Proof. Hardness follows from the co-NP-hardness of cautious reasoning over normal (aggregate-free) logic programs [Marek and Truszczyński, 1991; Schlipf, 1995]. For membership, we guess a total interpretation M, and check that: (i) $A \in M$, and (ii) $\neg.M^- = GUS_{\mathcal{P}}(M)$ (by Theorem 14 M is then an answer set). By Theorem 20, the checks are feasible in polynomial time. □

7 Related Work

To our knowledge, the only other work in which the notion of unfounded set has been defined for programs with aggregates is [Kemp and Stuckey, 1991]. However, their definition ignores aggregates in the second condition for unfounded sets. For the program $a(1){:}{-}\#\mathtt{count}\{X : a(X)\} > 0.$ the well-founded model of [Kemp and Stuckey, 1991] is $\emptyset$, leaving $a(1)$ undefined. Our well-founded model is $\{\neg a(1)\}$. Most of the results reported in this paper do not hold for unfounded sets as defined in [Kemp and Stuckey, 1991].

There have been several attempts to define well-founded semantics for programs with aggregates, not relying on unfounded sets. Several early approaches which are defined on a limited framework are discussed in [Kemp and Stuckey, 1991]. In [Van Gelder, 1992] a semantics is defined by compiling aggregates to rules with standard atoms. This approach was generalized in [Osorio and Jayaraman, 1999]. In any case, this strategy works only for certain classes of programs. In [Ross and Sagiv, 1997] an operator-based definition is given, which also works only on a restricted set of programs. In [Pelov, 2004] a well-founded semantics has been defined based on an approximating operator. Since this definition is substantially different from the one in this paper, we leave a comparison for future work.

Other works attempted to define stronger notions of well-founded semantics (also for programs with aggregates), among them the Ultimate Well-Founded Semantics of [Denecker *et al.*, 2001] and WFS^1 and WFS^2 of [Dix and Osorio, 1997]. Whether a characterization in terms of unfounded sets can exist for these semantics is not clear, and even if such generalized unfounded sets would exist, it is likely that some of the theorems in this paper will no longer hold, given that these semantics assign truth or falsity for more atoms.

In [Ferraris, 2004] it was shown that the semantics of Smodels programs with positive weight constraints is equal to answer sets as defined in [Faber *et al.*, 2004] on the respective fragment. Since by Theorem 16 $\mathcal{W}_{\mathcal{P}}^{\omega}(\emptyset)$ approximates

---

[5] Recall that we are dealing only with aggregates whose function is computable in polynomial time.

[6] In [Ferraris, 2004] it was independently shown that deciding answer set existence for a program with weight constraints (possibly containing negative weights) is Σ_2^P-complete.

answer sets of [Faber *et al.*, 2004], $\mathcal{W}^{\omega}_{\mathcal{P}}(\emptyset)$ can be used also as an approximating operator for the respective Smodels programs. Indeed, we can show that the *AtMost* pruning operator of Smodels [Simons *et al.*, 2002] is a special case of the Φ_I operator (defined in the proof sketch for Theorem 20).

8 Applications and Conclusion

The semantics of logic programs with aggregates is not straightforward, especially in presence of recursive aggregates. The declarative and fixpoint characterizations of answer sets, provided in Sections 4 and 5, allow for a better understanding of the meaning of programs with aggregates, and provide a handle on effective methods for computing answer sets for programs with (recursive) aggregates. In particular, the operator $\mathcal{W}^{\omega}_{\mathcal{P}}(\emptyset)$ can be used first to compute what must be in any answer set. Later in the computation, it can be used as a pruning operator and for answer set checking (as described in [Koch *et al.*, 2003; Pfeifer, 2004]).

Furthermore, since loop formulas encode unfounded sets [Lee, 2004], our results should be adaptable also to SAT-based ASP systems, all of which rely on loop formulas.

The complexity results make a clear demarcation between aggregates from the computational viewpoint, which is very useful to pick the appropriate techniques to be employed for the computation. The well-founded semantics of $\mathrm{LP}^{\mathcal{A}}_{m,a}$ is efficiently computable. Answer set semantics is in co-NP for $\mathrm{LP}^{\mathcal{A}}_{m,a}$, while nonmonotone aggregates bring about a complexity gap, and cannot be easily accommodated in NP systems.

A main concern for future work is therefore the exploitation of our results for the implementation of recursive aggregates in ASP systems.

References

[Baral, 2002] C. Baral. *Knowledge Representation, Reasoning and Declarative Problem Solving*. CUP, 2002.

[Calimeri *et al.*, 2002] F. Calimeri, W. Faber, N. Leone, and G. Pfeifer. Pruning Operators for Answer Set Programming Systems. In *NMR'2002*, pp. 200–209, 2002.

[Dell'Armi *et al.*, 2003] T. Dell'Armi, W. Faber, G. Ielpa, N. Leone, and G. Pfeifer. Aggregate Functions in DLV. In *ASP'03*, pp. 274–288, Messina, Italy, 2003. Online at `http://CEUR-WS.org/Vol-78/`.

[Denecker *et al.*, 2001] M. Denecker, N. Pelov, and M. Bruynooghe. Ultimate Well-Founded and Stable Model Semantics for Logic Programs with Aggregates. In *ICLP-2001*, pp. 212–226. 2001.

[Dix and Osorio, 1997] J. Dix and M. Osorio. On Well-Behaved Semantics Suitable for Aggregation. In *ILPS '97*, Port Jefferson, N.Y., 1997.

[Faber *et al.*, 2004] W. Faber, N. Leone, and G. Pfeifer. Recursive aggregates in disjunctive logic programs: Semantics and complexity. In *JELIA 2004*, pp. 200–212. 2004.

[Ferraris, 2004] P. Ferraris. Answer Sets for Propositional Theories. `http://www.cs.utexas.edu/users/otto/papers/proptheories.ps`, 2004.

[Gelfond and Lifschitz, 1991] M. Gelfond and V. Lifschitz. Classical Negation in Logic Programs and Disjunctive Databases. *NGC*, 9:365–385, 1991.

[Gelfond, 2002] M. Gelfond. Representing Knowledge in A-Prolog. In *Computational Logic. Logic Programming and Beyond*, LNCS 2408

[Kemp and Stuckey, 1991] D.B. Kemp and P.J. Stuckey. Semantics of Logic Programs with Aggregates. In *ISLP'91*, pp. 387–401. MIT Press, 1991.

[Koch *et al.*, 2003] C. Koch, N. Leone, and G. Pfeifer. Enhancing Disjunctive Logic Programming Systems by SAT Checkers. *Artificial Intelligence*, 15(1–2):177–212, 2003.

[Lee, 2004] J. Lee. A Model-Theoretic Counterpart of Loop Formulas. `http://www.cs.utexas.edu/users/appsmurf/papers/mtclf.pdf`, 2004.

[Leone *et al.*, 1997] N. Leone, P. Rullo, and F. Scarcello. Disjunctive Stable Models: Unfounded Sets, Fixpoint Semantics and Computation. *Information and Computation*, 135(2):69–112, 1997.

[Marek and Truszczyński, 1991] V.W. Marek and M. Truszczyński. Computing Intersection of Autoepistemic Expansions. In *LPNMR'91*, pp. 37–50, 1991.

[Osorio and Jayaraman, 1999] M. Osorio and B. Jayaraman. Aggregation and Negation-As-Failure. *NGC*, 17(3):255–284, 1999.

[Pelov and Truszczyński, 2004] N. Pelov and M. Truszczyński. Semantics of disjunctive programs with monotone aggregates - an operator-based approach. In *NMR 2004*, pp. 327–334, 2004.

[Pelov *et al.*, 2004] N. Pelov, M. Denecker, and M. Bruynooghe. Partial stable models for logic programs with aggregates. In *LPNMR-7*, LNCS 2923

[Pelov, 2004] N. Pelov. *Semantics of Logic Programs with Aggregates*. PhD thesis, Katholieke Universiteit Leuven, Leuven, Belgium, 2004.

[Pfeifer, 2004] G. Pfeifer. Improving the Model Generation/Checking Interplay to Enhance the Evaluation of Disjunctive Programs. In *LPNMR-7*, LNCS, pp. 220–233. 2004.

[Ross and Sagiv, 1997] K. A. Ross and Y. Sagiv. Monotonic Aggregation in Deductive Databases. *JCSS*, 54(1):79–97, 1997.

[Schlipf, 1995] J.S. Schlipf. The Expressive Powers of Logic Programming Semantics. *JCSS*, 51(1):64–86, 1995.

[Simons *et al.*, 2002] P. Simons, I. Niemelä, and T. Soininen. Extending and Implementing the Stable Model Semantics. *Artificial Intelligence*, 138:181–234, 2002.

[Tarski, 1955] A. Tarski. A lattice-theoretical fixpoint theorem and its applications. *Pacific J. Math*, 5:285–309, 1955.

[Van Gelder *et al.*, 1991] A. Van Gelder, K.A. Ross, and J.S. Schlipf. The Well-Founded Semantics for General Logic Programs. *JACM*, 38(3):620–650, 1991.

[Van Gelder, 1992] Allen Van Gelder. The Well-Founded Semantics of Aggregation. In *PODS'92*, pp. 127–138. ACM Press, 1992.