Using Unfounded Sets for Computing Answer Sets of Programs with Recursive Aggregates

Mario Alviano, Wolfgang Faber, and Nicola Leone

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

Abstract. Answer set programming with aggregates (ASP^A) allows for modelling many problems in a more concise way than with standard answer set programming (ASP). Previous works have focused on semantical and theoretical issues, and only few works have addressed computational issues, such as presenting algorithms and methods for implementing ASP^A.
In this paper, we fix a rich ASP language supporting recursive aggregates, and study means for implementing a reasoning engine for it. In particular, we leverage work on unfounded sets for ASP^A, and show how they can be used for characterizing and computing answer sets. Furthermore, we introduce an operator for effectively computing the greatest unfounded set (GUS), and study how it can be evaluated in a modular way, providing a means both for pruning the search space and for answer set checking. We have implemented these ideas and provide a brief sketch of the prototype architecture.

1 Introduction

The introduction of aggregate atoms [1–10] is one of the major syntactic 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. In particular, there is a lack of studies on algorithms and optimization methods for implementing recursive aggregates in ASP efficiently.

In this paper, we try to overcome this deficiency and make a step towards a more efficient implementation of recursive aggregates in ASP. To this end, we first focus on the properties of unfounded sets for programs with aggregates. Unfounded sets are at the basis of the implementation of virtually all currently available ASP solvers. Indeed, *native* ASP solvers like, e.g., DLV and Smodels, use unfounded sets for pruning the search space (through the well-founded operators); and *SAT-based* ASP solvers like, e.g., ASSAT and Cmodels, use the related concept of loop formulas [11, 12] for checking. Thus, an in-depth study of the properties of unfounded sets for programs with aggregates is a valuable contribution for the implementation efficient ASP^A systems.

We provide a new notion of unfounded sets for ASP^A, explain how unfounded sets can be profitably employed both for pruning the search space and for checking answer sets in ASP^A computations. We then design an operator for computing the greatest unfounded set (GUS), and, in order to support a more efficient implementation, we

provide a method for the modular computation of GUS, and sketch the architecture of our implementation of ASP^A.

We adopt the ASP^A semantics defined in [8], which seems to be receiving a consensus. Recent works, such as [13, 14] give further support for the plausibility of this semantics by relating it to established constructs for aggregate-free programs. In particular, [13] presented a semantics for very general programs, and showed that it coincides with [8] on ASP^A programs. We consider the rich ASP^A fragment allowing for disjunction, nonmonotonic negation, and both monotonic and anti-monotonic recursive aggregates; we denote this language by $\mathrm{DLP}^{\mathcal{A}}_{a,m}$.

Roughly, the main contributions of the paper are the following.

- We define a new and intuitive notion of unfounded set for $\mathrm{DLP}^{\mathcal{A}}_{a,m}$, relate it to previous notions showing also that it agrees with the standard notions of unfounded sets on aggregate-free programs, and characterize its properties.
- We show that unfounded sets can be profitably employed for pruning the search space in $\mathrm{DLP}^{\mathcal{A}}_{a,m}$ computations, by formally proving the properties of greatest unfounded sets (GUS) w.r.t. pruning.
- We demonstrate the formal properties allowing us to exploit greatest unfounded sets for answer-set checking in $\mathrm{DLP}^{\mathcal{A}}_{a,m}$ programs.
- We specify an operator $\mathcal{R}_{\mathcal{P},I}$ for computing the greatest unfounded sets.
- We design a modular evaluation technique for computing $\mathcal{R}_{\mathcal{P},I}$ component wise to allow for a more efficient implementation.
- We implement the above results in DLV, obtaining a system supporting the $\mathrm{DLP}^{\mathcal{A}}_{a,m}$ language, which is available for experimenting with recursive aggregates.[1]

To the best of our knowledge, our work provides the first implementation of recursive aggregates in disjunctive ASP. Previous implementations of aggregates in ASP either forbid recursive aggregates [5] or disallow disjunction [15, 16, 4, 10].[2]

2 Logic Programs with Aggregates

In this section, we recall syntax, semantics, and some basic properties of 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 [18, 19].

[1] Note that before our extension DLV supported only nonrecursive aggregates.

[2] Note that Cmodels [17] disallows aggregates in disjunctive rules.

Set Terms. A DLP$^{\mathcal{A}}$ *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.[3] 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: #`count` (number of terms), #`sum` (sum of non-negative integers), #`times` (product of positive integers), #`min` (minimum term), #`max` (maximum term)[4].

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. Also, an *aggregate atom* may have the form $T_1 \prec_1 f(S) \prec_2 T_2$, where $f(S)$ is an aggregate function, $\prec_1, \prec_2 \in \{<, \leq\}$, and T_1 and T_2 are terms.

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 $a_1, \ldots, a_n$ are standard atoms, $b_1, \ldots, b_m$ 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 sets 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.

---

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

[4] The first two aggregates roughly correspond, respectively, to the cardinality and weight constraint literals of Smodels. #`min` and #`max` are undefined for empty set.

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.

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 $U_{\mathcal{P}}$.

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* $S\}$.[5]
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*. A *totalization* of a (partial) interpretation I is a total interpretation J containing I, (i.e., J is a total interpretation and $I \subseteq J$).

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 (resp. some) of 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, $I(f(S)) = \perp$ (where $\perp$ is a fixed symbol not occurring in $\mathcal{P}$).

[5] 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 occurrence of variable X in Obj by $\sigma(X)$.

A ground 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, $\texttt{not}\ A$ 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* totalization J of I ; otherwise it is undefined.

Example 3. Consider the atom $A = \#\texttt{sum}\{\langle 1 : p(2,1)\rangle, \langle 2 : p(2,2)\rangle\} > 1$. 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 $\texttt{not}\ p(2,1)$. Therefore, either $I(S) = [2]$ or $I(S) = [1,2]$ and the application of $\#\texttt{sum}$ yields either 2 or 3, hence A is true w.r.t. I, since they are both greater than 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 and a ground rule r, the head of r is *true w.r.t.* I if some literal in $H(r)$ is true w.r.t. I; the body of r is *true w.r.t.* I if all literals in $B(r)$ are true w.r.t. I; rule r is *satisfied w.r.t.* I if its head is true w.r.t. I whenever its body is 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 4. Consider interpretation $I_1 = \{p(a)\}$, $I_2 = \{\texttt{not}\ p(a)\}$ and two programs $P_1 = \{p(a) \text{ :- } \#\texttt{count}\{X : p(X)\} > 0.\}$ and $P_2 = \{p(a) \text{ :- } \#\texttt{count}\{X : p(X)\} < 1.\}$.

$Ground(P_1) = \{p(a) \text{ :- } \#\texttt{count}\{\langle a : p(a)\rangle\} > 0.\}$ and $Ground(P_1)^{I_1} = Ground(P_1)$, $Ground(P_1)^{I_2} = \emptyset$. Furthermore, $Ground(P_2) = \{p(a) \text{ :- } \#\texttt{count}\{\langle a : p(a)\rangle\} < 1.\}$, 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 $Ground(\mathcal{P}) \setminus Ground(\mathcal{P})^A$ 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 5. 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 $\mathrm{DLP}^{\mathcal{A}}_{m,a}$ the fragment of $\mathrm{DLP}^{\mathcal{A}}$ in which monotone and antimonotone literals may occur. In the following, by *program* we will usually refer to a $\mathrm{DLP}^{\mathcal{A}}_{m,a}$ program. Given a rule r of a $\mathrm{DLP}^{\mathcal{A}}_{m,a}$ program, we denote with $Mon(B(r))$ and $Ant(B(r))$, respectively, the set of *monotone* and *antimonotone* literals in $B(r)$. Note that, as described in [20], many programs with nonmonotone literals can be polynomially rewritten into $\mathrm{DLP}^{\mathcal{A}}_{m,a}$ programs. Some important examples include programs containing aggregate atoms of the form $T_1 \prec_1 f(S) \prec_2 T_2$ and $f(S) = T$ (which per se are nonmonotone independent of $f(S)$), which can be rewritten to conjunctions $T_1 \prec_1 f(S), f(S) \prec_2 T_2$ and $f(S) \geq T, f(S) \leq T$, respectively.

3 Unfounded Sets

We now give a definition of unfounded set for $\mathrm{DLP}^{\mathcal{A}}$ programs with monotone and antimonotone aggregates, extending the one of [14].

In the following we denote by $S_1 \dot{\cup} \neg.S_2$ 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 $\mathrm{DLP}^{\mathcal{A}}_{m,a}$ program $\mathcal{P}$ w.r.t. an interpretation I if, for each rule $r \in \mathcal{P}$ such that $H(r) \cap X \neq \emptyset$, at least one of the following conditions holds:

1. $Ant(B(r))$ is false w.r.t. I.
2. $Mon(B(r))$ is false w.r.t. $I \dot{\cup} \neg.X$.
3. $H(r)$ is true w.r.t. $I \dot{\cup} \neg.X$.

While condition 1 declares that rule satisfaction does not depend on atoms in X, conditions 2 and 3 ensure that the rule is satisfied also if the atoms in X are switched to false. Note that condition 3 is equivalent to $(H(r) \setminus X) \cap I \neq \emptyset$, and $\emptyset$ is always an unfounded set, independent of interpretation and program.

Example 6. Consider the following program P:

$$a(1) \vee a(2). \qquad a(1) :- \#\texttt{count}\{\langle 1 : a(2)\rangle\} > 1. \qquad a(2) :- \#\texttt{count}\{\langle 1 : a(1)\rangle\} > 1.$$

and $I = \{a(1), a(2)\}$. Then $\{a(1)\}$ and $\{a(2)\}$ are unfounded sets for P w.r.t. I. $\{a(1), a(2)\}$ is not an unfounded set for P w.r.t. I, as for the first rule none of the three conditions holds.

Theorem 1. *A set X of ground atoms is an unfounded set for a* $\mathrm{DLP}^{\mathcal{A}}_{a,m}$ *program* $\mathcal{P}$ *w.r.t. an interpretation* I *according to Def. 1 iff it is an unfounded set for* $\mathcal{P}$ *w.r.t.* I *according to Def. 1 of [21].*

Proof. According to Def. 1 of [21], 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: a) some literal of $B(r)$ is false w.r.t. I, b) some literal of $B(r)$ is false w.r.t. $I \dot\cup \neg.X$, or c) some atom of $H(r) \setminus X$ is true w.r.t. I.

First of all, let us observe that conditions 1 and 2 of Def. 1 trivially imply conditions a) and b), respectively, and that, as noted earlier, condition 3 of Def. 1 is equivalent to condition c).

Now, observe that if a monotone body literal is false w.r.t. I, it is also false w.r.t. $I \dot\cup \neg.X$. In a similar way, if an antimonotone body literal of r is false w.r.t. $I \dot\cup \neg.X$, then it is false also w.r.t. I. Therefore, if condition a) holds for a monotone literal, condition 2 holds for this literal; if condition a) holds for an antimonotone literal, condition 1 holds. Likewise, if condition b) holds for a monotone literal, condition 2 holds; if condition b) holds for an antimonotone literal, condition 1 holds.

Thus, on $\mathrm{DLP}^{\mathcal{A}}_{m,a}$ our definition of unfounded set specializes Def. 1 of [21] by imposing stricter properties in conditions 1 and 2.

From this equivalence and results in [21] it follows that unfounded sets as defined in Def. 1 also coincide with other definitions of unfounded sets for various language fragments.

Corollary 1. *For a non-disjunctive, aggregate-free program* $\mathcal{P}$*and an interpretation* I*, any unfounded set w.r.t. Def. 1 is a standard unfounded set (as defined in [22]).*

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 [23].*

For a non-disjunctive $\mathrm{LP}^{\mathcal{A}}_{m,a}$ *program* $\mathcal{P}$*and an interpretation* I*, any unfounded set w.r.t. Def. 1 is an unfounded set as defined in [14].*

We next state an important monotonicity property of unfounded sets.

Proposition 1. *Let* I *be a partial interpretation for a* $\mathrm{DLP}^{\mathcal{A}}_{m,a}$ *program* $\mathcal{P}$ *and* X *an unfounded set for* $\mathcal{P}$ *w.r.t.* I*. Then, for each* $J \supseteq I$*,* X *is an unfounded set for* $\mathcal{P}$ *w.r.t.* J *as well.*

Proof. If X is an unfounded set for $\mathcal{P}$ w.r.t. I, then for each $a \in X$ and for each $r \in \mathcal{P}$ with $a \in H(r)$, (1) $Ant(B(r))$ is false w.r.t. I, or (2) $Mon(B(r))$ is false w.r.t. $I \dot\cup \neg.X$, or (3) $H(r)$ is true w.r.t. $I \dot\cup \neg.X$ holds. Now, note that since $I \subseteq J$ holds, then also $I \dot\cup \neg.X \subseteq J \dot\cup \neg.X$ holds. So, if (1) holds, it holds also for J, and if (2) or (3) hold, then they hold also for $J \dot\cup \neg.X$.

We next define the central notion in the remainder of this work, the *Greatest Unfounded Set* (GUS), as the union of all unfounded sets.

Definition 2. *Let I be an interpretation for a program $\mathcal{P}$. Then, 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.*

From Proposition 1 it follows that the GUS of an interpretation I is always contained in the GUS of a superset of I.

Proposition 2. *Let I be an interpretation for a program $\mathcal{P}$. Then, $GUS_{\mathcal{P}}(I) \subseteq GUS_{\mathcal{P}}(J)$, for each $J \supseteq I$.*

Note that despite its name, the GUS is not always guaranteed to be an unfounded set. In the non-disjunctive case, the union of two unfounded sets is an unfounded set as well, also in presence of monotone and antimonotone aggregates [14], and so for these programs a GUS is necessarily an unfounded set. However, in the presence of disjunctive rules, this property does no longer hold, as shown in [23]. Therefore it also does not hold for $\mathrm{DLP}^{\mathcal{A}}_{m,a}$, and as a consequence a GUS need not be an unfounded set.

Observation 2 *If X_1 and X_2 are unfounded sets for a $\mathrm{DLP}^{\mathcal{A}}_{m,a}$ program $\mathcal{P}$ w.r.t. I, then $X_1 \cup X_2$ is not necessarily an unfounded set.*

By virtue of Theorem 1, Proposition 1 of [21] carries over to unfounded sets of Definition 1.

Proposition 3. *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.*

This allows for defining the class of unfounded-free interpretations for which the GUS is guaranteed to be an unfounded set.

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.*

As an easy consequence we obtain:

Proposition 4. *Let I be an unfounded-free interpretation for a program $\mathcal{P}$. Then, $GUS_{\mathcal{P}}(I)$ is an unfounded set.*

Next, we show an interesting property for total interpretations.

Proposition 5. *Let I be a total interpretation for a program $\mathcal{P}$. Then, I is unfounded-free iff no non-empty set $X \subseteq I^+$ is an unfounded set for $\mathcal{P}$ w.r.t. I.*

Proof. ($\Longrightarrow$) If a non-empty subset Y of I^+ is an unfounded set for $\mathcal{P}$ w.r.t. I, then I is not unfounded-free.
($\Longleftarrow$) If I is not unfounded-free, then there exists a non-empty subset of I^+ which is an unfounded set for $\mathcal{P}$ w.r.t. I. Let X be an unfounded set for $\mathcal{P}$ w.r.t. I such that $Y = X \cap I \neq \emptyset$. Note that $I \dot{\cup} \neg.X = I \dot{\cup} \neg.Y$, then Y is also an unfounded set for $\mathcal{P}$ w.r.t. I.

4 Answer Set Checking via Unfounded Sets

Unfounded sets can be used to characterize models and answer sets; these characterizations can be profitably used for answer set checking. Given Theorem 1, the following results are consequences of Theorem 4 and Corollary 6 of [21].

Proposition 6. *Let M be a total interpretation for a program $\mathcal{P}$. Then M is a model for $\mathcal{P}$ iff M^- is an unfounded set for $\mathcal{P}$ w.r.t. M.*

Proposition 7. *Let M be a model for $\mathcal{P}$. M is an answer-set for $\mathcal{P}$ iff M is unfounded-free for $\mathcal{P}$.*

Furthermore, we can show that unfounded sets also characterize minimal models.

Proposition 8. *Let M be a model for a positive program $\mathcal{P}$. M is a minimal model for $\mathcal{P}$ iff it is unfounded-free.*

Proof. ($\Longleftarrow$) If M is not minimal then there exists another model M_1 such that $M_1^+ \subset M^+$, and so $X = M^+ \setminus M_1^+ \neq \emptyset$. Then, for each $r \in \mathcal{P}$ such that $H(r) \cap X \neq \emptyset$, (i) $H(r) \cap M_1^+ \neq \emptyset$, or (ii) $Ant(B(r))$ is false w.r.t. M_1, or (iii) $Mon(B(r))$ is false w.r.t. M_1. Note that $M_1 = (M \setminus X) \cup \neg.X = M \dot{\cup} \neg.X$, and then: from (i) follows that $H(r)$ is true w.r.t. $M \dot{\cup} \neg.X$, from (ii) follows that $Ant(B(r))$ is false w.r.t. M (because $M_1 \leq M$), from (iii) follows that $Mon(B(r))$ is false w.r.t. $M \dot{\cup} \neg.X$. So, X is an unfounded set for $\mathcal{P}$ w.r.t. M, and then M is not unfounded-free.
($\Longrightarrow$) Assume, by contradiction, that M is not unfounded-free. Then, by Proposition 5, there exists a non-empty $X \subseteq M^+$ which is an unfounded set for $\mathcal{P}$ w.r.t. M. Now, we show that the total interpretation $M_1 = M \dot{\cup} \neg.X$ is a model for $\mathcal{P}$ (contradicting the minimality of M). Let r be a rule of $\mathcal{P}$ such that $H(r)$ is true w.r.t. M, and $H(r)$ is false w.r.t. M_1. Then, $H(r) \cap X \neq \emptyset$. But X is an unfounded set for $\mathcal{P}$ w.r.t. M, then (1) $Ant(B(r))$ is false w.r.t. M (and then it is false w.r.t. M_1, because $M_1 \leq M$), or (2) $Mon(B(r))$ is false w.r.t. $M \dot{\cup} \neg.X = M_1$, or (3) $H(r)$ is true w.r.t. $M \dot{\cup} \neg.X = M_1$. Note that (3) cannot holds by assumption. Then, r is satisfied w.r.t. M_1 by body, contradicting the minimality of M.

We next show that the condition of being unfounded-free is invariant for a program and its reduct.

Lemma 1. *Let M be a total interpretation for a program $\mathcal{P}$. M is unfounded-free for $\mathcal{P}$ iff it is unfounded-free for $\mathcal{P}^M$.*

Proof. ($\Longrightarrow$) If X is not an unfounded set for $\mathcal{P}$ w.r.t. M, then for each $a \in X$ there exists $r \in \mathcal{P}$ such that r violates all condition of Definition 1. Then, from condition (1) and (2), $B(r)$ is true w.r.t. M. Therefore, the image r' of r is in $\mathcal{P}^M$. Clearly, r' violates all conditions of Definition 1 for $\mathcal{P}^M$ w.r.t. M. Now, if M is unfounded-free for $\mathcal{P}$, then, by Proposition 5, every non-empty $X \subseteq M^+$ is not an unfounded set for $\mathcal{P}$ w.r.t. M, and then it is not an unfounded set for $\mathcal{P}^M$ w.r.t. M. So, M is unfounded-free for $\mathcal{P}^M$.
($\Longleftarrow$) Let X be an unfounded set for $\mathcal{P}$ w.r.t. M. Then, for each $a \in X$ and for each $r \in \mathcal{P}$ with $a \in H(r)$, (1) $Ant(B(r))$ is false w.r.t. M, or (2) $Mon(B(r))$ is false w.r.t.

$M \dot\cup \neg.X$, or (3) $H(r)$ is true w.r.t. $M \dot\cup \neg.X$. Case (1) or (2) imply that r has no image in $\mathcal{P}^M$ or condition (2) holds for r'. Case (3) imply that condition (3) holds also for r'. So, X is an unfounded set also for $\mathcal{P}^M$ w.r.t. M. Therefore, if M is not unfounded-free for $\mathcal{P}$, then it is not unfounded-free for $\mathcal{P}^M$. It follows that M unfounded-free for $\mathcal{P}^M$ implies M unfounded-free for $\mathcal{P}$.

Furthermore, $GUS_{\mathcal{P}}(I)$ permits to check whether I is unfounded-free, and then whether it is an answer set.

Theorem 3. *Let I be a total interpretation for a program $\mathcal{P}$. I is unfounded-free if and only if $I^- = GUS_{\mathcal{P}}(I)$.*

Proof. ($\Longleftarrow$) It is easy to see that each unfounded set X for $\mathcal{P}$ w.r.t. I is a subset of I^-, and then $I \cap X = \emptyset$ holds.
($\Longrightarrow$) For each unfounded set X for $\mathcal{P}$ w.r.t. I, $I \cap X = \emptyset$ holds. Since I is total, this is equivalent to $X \subseteq I^-$, and then $GUS_{\mathcal{P}}(I) \subseteq I^-$. By Proposition 6, it follows that $I^- \subseteq GUS_{\mathcal{P}}(I)$, and then $I^- = GUS_{\mathcal{P}}(I)$.

Corollary 2. *Given a total interpretation I for a program $\mathcal{P}$, I is an answer set if and only if $I^- = GUS_{\mathcal{P}}(I)$ and I^- is an unfounded set w.r.t. $\mathcal{P}$ and I.*

These results allow for checking whether a model or an interpretation is an answer set just by using the notion of unfounded sets.

5 Pruning via Unfounded Sets

In this section we show some properties of $GUS_{\mathcal{P}}(I)$, which may be used during the computation of the answer sets, for pruning the search space and detecting useless branches of the computation.

Theorem 4. *Given an interpretation I for a program $\mathcal{P}$, if $I \cap GUS_{\mathcal{P}}(I) \neq \emptyset$, then no totalization of I is an answer set for $\mathcal{P}$.*

Proof. If $I \cap GUS_{\mathcal{P}}(I) \neq \emptyset$, then there exists an unfounded set X for $\mathcal{P}$ w.r.t. I such that $I \cap X \neq \emptyset$. Let J be a totalization of I. Then, by Proposition 1, X is an unfounded set for $\mathcal{P}$ w.r.t. J. Clearly, $J \cap X \neq \emptyset$, so J is not unfounded-free. By Proposition 7 J is not an answer-set for $\mathcal{P}$.

Thus, during the construction of answer sets one may want to compute the GUS with respect to the interpretation so far and test whether it contains some element of the interpretation. If so, one should abandon the construction and backtrack, as no answer set can be found in the current branch. Moreover, the GUS also serves as an inference operator for pruning the search space.

Theorem 5. *Given an interpretation I for a program $\mathcal{P}$, if J is an answer set containing I, then J contains $I \dot\cup \neg.GUS_{\mathcal{P}}(I)$ as well.*

Proof. Assume $J \not\supseteq I \dot\cup \neg.GUS_{\mathcal{P}}(I)$ then $J \cap GUS_{\mathcal{P}}(I) \neq \emptyset$. From Proposition 2 it follows that $J \cap GUS_{\mathcal{P}}(J) \neq \emptyset$, and then, by Theorem 4, J is not an answer set for $\mathcal{P}$.

In other words, $\neg.GUS_{\mathcal{P}}(I)$ is contained in all answer sets extending I, so when constructing answer set candidates we can safely add these literals to the candidate.

6 Computing Greatest Unfounded Sets

We now define an operator for computing the Greatest Unfounded Set of a $DLP^{\mathcal{A}}_{m,a}$ program $\mathcal{P}$ w.r.t. an interpretation I: the operator $\mathcal{R}_{\mathcal{P},I}$ that, given a set X of ground atoms, discards the elements in X that do not satisfy any of the unfoundedness conditions of Definition 1.

Definition 4. *Let $\mathcal{P}$ be a $DLP^{\mathcal{A}}_{m,a}$ program and I an interpretation. Then we define the operator $\mathcal{R}_{\mathcal{P},I}$ as a mapping $2^{B_\mathcal{P}} \rightarrow 2^{B_\mathcal{P}}$ as follows:*

$$\mathcal{R}_{\mathcal{P},I}(X) = \{a \in X \mid \forall r \in ground(\mathcal{P}) \; with \; a \in H(r), Ant(B(r)) \; \textit{is false w.r.t.} \; I, \\ \textit{or} \; Mon(B(r)) \; \textit{is false w.r.t.} \; I \dot\cup \neg.X, \\ \textit{or} \; H(r) \; \textit{is true w.r.t.} \; I \dot\cup \neg.\{a\}\}$$

Given a set $X \subseteq B_\mathcal{P}$, the sequence $R_0 = X$, $R_n = \mathcal{R}_{\mathcal{P},I}(R_{n-1})$ decreases monotonically and converges finitely to a limit that we denote by $\mathcal{R}^{\omega}_{\mathcal{P},I}(X)$. We next show that $\mathcal{R}^{\omega}_{\mathcal{P},I}(B_\mathcal{P} \setminus I)$ is an unfounded set, and we can therefore use this operator to detect undefined atoms that can be safely switched to false, reducing the search space.

Proposition 9. *Given a $DLP^{\mathcal{A}}_{m,a}$ program $\mathcal{P}$ and an interpretation I, $\mathcal{R}^{\omega}_{\mathcal{P},I}(B_\mathcal{P} \setminus I)$ is an unfounded set for $\mathcal{P}$ w.r.t. I.*

Proof. Let $X = \mathcal{R}^{\omega}_{\mathcal{P},I}(B_\mathcal{P} \setminus I)$. By definition of $\mathcal{R}_{\mathcal{P},I}$, we have $X \subseteq B_\mathcal{P} \setminus I$, and hence $X \cap I = \emptyset$. Now, for each $a \in X$ and for each $r \in \mathcal{P}$, $a \in H(r)$ implies that $Ant(B(r))$ is false w.r.t. I, or $Mon(B(r))$ is false w.r.t. $I \dot\cup \neg.X$, or $H(r)$ is true w.r.t. $I \dot\cup \neg.\{a\}$. If the last holds, since $X \cap I = \emptyset$, also $H(r)$ is true w.r.t. $I \dot\cup \neg.X$. Then, X is an unfounded set for $\mathcal{P}$ w.r.t. I.

Importantly, $\mathcal{R}_{\mathcal{P},I}$ does not discard any unfounded set contained in the input set.

Proposition 10. *Let $\mathcal{P}$ be a $DLP^{\mathcal{A}}_{m,a}$ program, I be an interpretation for $\mathcal{P}$, and $J \subseteq B_\mathcal{P}$. Every unfounded set for $\mathcal{P}$ w.r.t. I which is contained in J is also contained in $\mathcal{R}^{\omega}_{\mathcal{P},I}(J)$.*

Proof. Let $X \subseteq J$ be an unfounded set for $\mathcal{P}$ w.r.t. I. For each $a \in X$ and for each rule $r \in \mathcal{P}$ such that $a \in H(r)$, $Ant(B(r))$ is false w.r.t. I, or $Mon(B(r))$ is false w.r.t. $I \dot\cup \neg.X$, or $H(r)$ is true w.r.t. $I \dot\cup \neg.X$ holds. If the last holds, since $\{a\} \subseteq X$, $H(r)$ is true w.r.t. $I \dot\cup \neg.\{a\}$ as well. Then, from the definition of $\mathcal{R}_{\mathcal{P},I}$, $\mathcal{R}_{\mathcal{P},I}(X) = X$ holds and, since X is monotonic and $X \subseteq J$, $\mathcal{R}^{\omega}_{\mathcal{P},I}(J)$ must contain X.

Using the above propositions, we can prove that $\mathcal{R}^{\omega}_{\mathcal{P},I}(B_\mathcal{P} \setminus I)$ computes the greatest unfounded set for $\mathcal{P}$ w.r.t. I.

Theorem 6. *Let $\mathcal{P}$ be a $DLP^{\mathcal{A}}_{m,a}$ program and I an unfounded-free interpretation for it. Then, $\mathcal{R}^{\omega}_{\mathcal{P},I}(B_\mathcal{P} \setminus I) = GUS_\mathcal{P}(I)$.*

Proof. ($\supseteq$) Since I is unfounded-free, $I \cap X = \emptyset$ holds for each unfounded set for $\mathcal{P}$ w.r.t. I, and then $X \subseteq B_\mathcal{P} \setminus I$. So, by Proposition 10, also $X \subseteq \mathcal{R}^{\omega}_{\mathcal{P},I}(B_\mathcal{P} \setminus I)$ holds, and then $GUS_{\mathcal{P},I}(B_\mathcal{P} \setminus I)$ is contained in $\mathcal{R}^{\omega}_{\mathcal{P},I}(B_\mathcal{P} \setminus I)$.
($\subseteq$) By Proposition 9, $\mathcal{R}^{\omega}_{\mathcal{P},I}(B_\mathcal{P} \setminus I)$ is an unfounded set for $\mathcal{P}$ w.r.t. I, and then, by definition of GUS, it is contained in $GUS_\mathcal{P}(I)$.

Is is easy to see that the fixpoint of the $\mathcal{R}_{\mathcal{P},I}$ operator is efficiently computable. Thus, from the above theorem one can employ $\mathcal{R}_{\mathcal{P},I}$ as powerful and efficient pruning operator for unfounded-free interpretations. Actually, on the large class of head-cycle free programs [24], the $\mathcal{R}_{\mathcal{P},I}$ allows us to always compute the greatest unfounded set, even if the interpretation is not unfounded-free, and can be therefore employed both for pruning and answer-set checking. In the next section, we show how this can be done in an efficient way providing an algorithm for the modular computation of GUS via $\mathcal{R}_{\mathcal{P},I}$.

7 Modular Evaluation of Greatest Unfounded Sets

In this section, we show how we can localize the computation of unfounded sets. To this end, we define the notion of *dependency graph*, the strongly connected components of which define the modules, on which the local computation will work.

With every ground program $\mathcal{P}$, we associate a directed graph $DG_{\mathcal{P}} = (\mathcal{N}, E)$, called the *dependency graph* of $\mathcal{P}$, in which (i) each atom of $\mathcal{P}$ is a node in $\mathcal{N}$ and (ii) there is an arc in E directed from a node a to a node b iff there is a rule r in $\mathcal{P}$ such that $b \in H(r)$ and a is a standard atom in $Mon(B(r))$ or an atom appearing in the ground set of an aggregate literal in $Mon(B(r))$.

An important and well-known class of programs are *head-cycle-free (HCF)* programs: A program $\mathcal{P}$ is HCF iff there is no rule r in $\mathcal{P}$ such that two predicates occurring in the head of r are in the same cycle of $DG_{\mathcal{P}}$. In our implementation for $\mathrm{DLP}^{\mathcal{A}}_{m,a}$, described in Section 8, we consider only HCF programs. This class of programs has recently been shown to be the largest class of programs for which standard reasoning tasks are still in NP (cf. [25]). The main result of this section, Theorem 7, is therefore also stated for HCF programs.

We can partition the set of ground atoms occurring in $\mathcal{P}$ in strongly connected components. Two atoms a and b are in the same component if there is both a path from a to b and a path from b to a in $DG_{\mathcal{P}}$. Also, we can define a partial order $\preceq$ for components: $C_1 \preceq C_2$ iff there exist $a \in C_1$, $b \in C_2$ such that there is a path from a to b. Moreover, the subprogram $\mathcal{P}_C \subseteq \mathcal{P}$ associated with a component C consists of all rules $\mathcal{P}$ which contain an atom of C in their heads. Before introducing the algorithm, we show some properties that hold for the $\mathcal{R}_{\mathcal{P},I}$ operator.

Lemma 2. *Let $\mathcal{P}$ be a program and I be an interpretation. For each sets X and Y such that $X \subseteq Y$, $\mathcal{R}^{\omega}_{\mathcal{P},I}(X) \subseteq \mathcal{R}^{\omega}_{\mathcal{P},I}(Y)$ holds.*

Proof. By induction. The only condition of Def. 4 that depends on the starting set X is "$Mon(B(r))$ is false w.r.t. $I \dot{\cup} \neg.X$". If this holds for some atom in $\mathcal{R}_{\mathcal{P},I}(X)$ and some rule r in $\mathcal{P}$, then $Mon(B(r))$ is false also w.r.t. $I \dot{\cup} \neg.Y$, because $I \dot{\cup} \neg.Y \leq I \dot{\cup} \neg.X$. So, $\mathcal{R}_{\mathcal{P},I}(X) \subseteq \mathcal{R}_{\mathcal{P},I}(Y)$. Assuming $\mathcal{R}^{(i)}_{\mathcal{P},I}(X) \subseteq \mathcal{R}^{(i)}_{\mathcal{P},I}(Y)$, then $\mathcal{R}^{(i+1)}_{\mathcal{P},I}(X) = \mathcal{R}_{\mathcal{P},I}(\mathcal{R}^{(i)}_{\mathcal{P},I}(X)) \subseteq \mathcal{R}_{\mathcal{P},I}(\mathcal{R}^{(i)}_{\mathcal{P},I}(Y)) = \mathcal{R}^{(i+1)}_{\mathcal{P},I}(Y)$.

Lemma 3. *Let $\mathcal{P}$ be a program and I an interpretation, C a component of $\mathcal{P}$ and $\mathcal{P}_C$ the subprogram associated to C. Then, for each $X \subseteq C$, $\mathcal{R}_{\mathcal{P}_C,I}(X) = \mathcal{R}_{\mathcal{P},I}(X)$.*

Proof. Clearly, each rule r of $\mathcal{P}$ with $H(r) \cap X \neq \emptyset$ is also in $\mathcal{P}_C$.

Theorem 7. *Let $C_1, C_2, ..., C_n$ be a total order for the components of an HCF program $\mathcal{P}$ such that $C_i \preceq C_j$ implies $i \leq j$. Starting from $I_0 := I$ and then, for each $i = 1, ..., n$, computing $X_i := \mathcal{R}^{\omega}_{\mathcal{P}_{C_i}, I_{i-1}}(C_i \setminus I)$, $I_i := I_{i-1} \cup \neg.X_i$, it holds that I_n is equal to $I \cup \neg.GUS_{\mathcal{P}}(I)$.*

Proof. We prove that at each step of the computation $X_i = \mathcal{R}^{\omega}_{\mathcal{P},I}(B_{\mathcal{P}} \setminus I) \cap C_i$ holds.
Base ($\subseteq$). From Lemma 3 and Lemma 2 it follows that $X_1 = \mathcal{R}^{\omega}_{\mathcal{P}_{C_1},I}(C_1 \setminus I) = \mathcal{R}^{\omega}_{\mathcal{P},I}(C_1 \setminus I) \subseteq \mathcal{R}^{\omega}_{\mathcal{P},I}(B_{\mathcal{P}} \setminus I)$. So, $X_1 \subseteq \mathcal{R}^{\omega}_{\mathcal{P},I}(B_{\mathcal{P}} \setminus I) \cap C_1$, because $X_1 \subseteq C_1$.
Base ($\supseteq$). For each $a \in \mathcal{R}^{\omega}_{\mathcal{P},I}(B_{\mathcal{P}} \setminus I) \cap C_1$ and for each $r \in P$ with $a \in H(r)$, (1) $Ant(B(r))$ is false w.r.t. I, or (2) $Mon(B(r))$ is false w.r.t. $I \dot{\cup} \neg.\mathcal{R}^{\omega}_{\mathcal{P},I}(B_{\mathcal{P}} \setminus I)$, or (3) $H(r)$ is true w.r.t. $I \dot{\cup} \neg.\{a\}$. Note that, since $\mathcal{P}$ is HCF, a is the only atom in $H(r)$ belonging to C_1, so from (3) it follows that $H(r)$ is true w.r.t. $I \dot{\cup} \neg.(\mathcal{R}^{\omega}_{\mathcal{P},I}(B_{\mathcal{P}} \setminus I) \cap C_1)$. Also, note that $Mon(B(r))$ depends only on atoms in C_1. Then, from (2) it follows that $Mon(B(r))$ is false w.r.t. $\mathcal{R}^{\omega}_{\mathcal{P},I}(B_{\mathcal{P}} \setminus I) \cap C_1$. So, $\mathcal{R}^{\omega}_{\mathcal{P},I}(B_{\mathcal{P}} \setminus I) \cap C_1$ is an unfounded set for $\mathcal{P}_{C_1}$ w.r.t. I and, by Def. 4, it is a subset of X_1.
Suppose that $X_i = \mathcal{R}^{\omega}_{\mathcal{P},I}(B_{\mathcal{P}} \setminus I) \cap C_i$.
($\subseteq$) For each $a \in X_{i+1}$ and for each $r \in \mathcal{P}_{C_{i+1}}$ with $a \in H(r)$, $Ant(B(r))$ is false w.r.t. I_i (and w.r.t. I because $I_i \leq I$), or $Mon(B(r))$ is false w.r.t. $I_i \dot{\cup} \neg.X_{i+1}$ ($= I \dot{\cup} \neg.(X_{i+1} \cup (I_i^- \setminus I^-))$), or $H(r)$ is true w.r.t. $I_i \dot{\cup} \neg.\{a\}$ (and therefore also w.r.t. $I \dot{\cup} \neg.(X_{i+1} \cup (I_i^- \setminus I^-))$ because $I_i^+ = I^+$ and a is the only atom belonging to some C_j, for $j = 1, ..., i+1$). No other rule in $\mathcal{P} \setminus \mathcal{P}_{C_{i+1}}$ has a in head, and then $X_{i+1} \cup (I_i^- \setminus I^-)$ is an unfounded set for $\mathcal{P}$ w.r.t. I. So, from Proposition 10, X_{i+1} is a subset of $\mathcal{R}^{\omega}_{\mathcal{P},I}(B_{\mathcal{P}} \setminus I)$.
($\supseteq$) For each $a \in \mathcal{R}^{\omega}_{\mathcal{P},I}(B_{\mathcal{P}} \setminus I) \cap C_{i+1}$ and for each $r \in \mathcal{P}$ with $a \in H(r)$, (1) $Ant(B(r))$ is false w.r.t. I (and so w.r.t. I_i because $I_i \supseteq I$), or (2) $Mon(B(r))$ is false w.r.t. $I \dot{\cup} \neg.\mathcal{R}^{\omega}_{\mathcal{P},I}(B_{\mathcal{P}} \setminus I)$, or (3) $H(r)$ is true w.r.t. I (and so w.r.t. I_i). From (2) it follows that $Mon(B(r))$ is false w.r.t. $Y = I \dot{\cup} \neg.(\mathcal{R}^{\omega}_{\mathcal{P},I}(B_{\mathcal{P}} \setminus I) \cap (C_1 \cup C_2 \cup ... \cup C_{i+1})$ (because $Mon(B(r))$ depends only from atoms in $C_1, \ldots, C_{i+1}$).
But $I \dot{\cup} \neg.(\mathcal{R}^{\omega}_{\mathcal{P},I}(B_{\mathcal{P}} \setminus I) \cap (C_1 \cup C_2 \cup ... \cup C_i) = I_i$ and $X_{i+1} \subseteq \mathcal{R}^{\omega}_{\mathcal{P},I}(B_{\mathcal{P}} \setminus I) \cap C_{i+1}$. Then $Y^+ = (I_i \dot{\cup} \neg.X_{i+1})^+$ and $Y^- \supseteq (I_i \dot{\cup} \neg.X_{i+1})^-$, so $Y \leq I_i \dot{\cup} \neg.X_{i+1}$ and $Mon(B(r))$ is false also w.r.t. $I_i \dot{\cup} \neg.X_{i+1}$.

8 Prototype Architecture

We have implemented the approach described in Section 7, modifying the system DLV, which already processes nonrecursive aggregates. For a thorough description of the DLV architecture, we refer to [26]. The main structure of the system is reported in Figure 1.

The input, after having possibly been processed by some frontends, is handed to the DLV core, in particular to the grounding, which produces a ground version of the input, which is guaranteed to have the same answer sets as the input. Control is then handed over to the model generator, which performs a backtracking heuristic search for models, which serve as answer set candidates. During this search, various pruning techniques are employed, among them also unfounded set computations (cf. [27, 28]). Each of the found answer set candidates is then submitted to the model checker, which

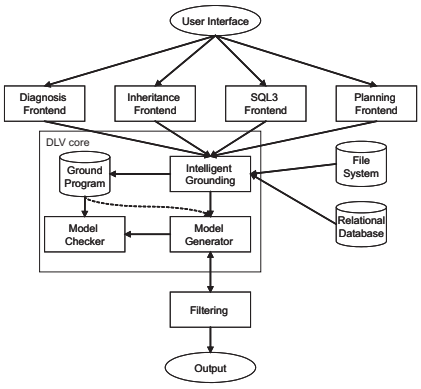

Fig. 1. DLV system architecture

verifies whether the model is an answer set (cf. [29]). When the input program is HCF, this check need not be done, as any produced candidate is known to be an answer set.

Therefore, for our prototype we had to modify the grounding and generator modules. In the grounding phase, in the case of non-recursive aggregates all instances of predicates inside an aggregate are known at the time the aggregate is instantiated. When supporting also recursive aggregates, this assumption no longer holds, and therefore a somewhat more complex grounding strategy has to be employed. Concerning the model generator, a large part of the existing machinery for aggregates could be re-used. In order to treat recursive aggregates correctly, unfounded set computation involving aggregates, which has not been present in DLV so far, is necessary. We have implemented unfounded set computations using an optimized implementation of the method described in Section 7, which further localize the computation by focusing only on the components that have been affected by the last propagation step. The system prototype is available at `http://www.dlvsystem.com/dlvRecAggr`, and supports a superset of hcf programs, requiring head-cycle freeness only on the components with recursive aggregates. Preliminary results of experiments on Companies Control examples indicate that the implementation offers good performance on medium-size instances.

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 Disjunctive Logic Programming: Semantics, Complexity, and Implementation in DLV. In: IJCAI 2003, Acapulco, Mexico,(2003) 847–852
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. Son, T.C., Pontelli, E.: A Constructive Semantic Characterization of Aggregates in ASP. Theory and Practice of Logic Programming (2007) Accepted for publication, available in CoRR as cs.AI/0601051.
10. Son, T.C., Pontelli, E., Elkabani, I.: On Logic Programming with Aggregates. Tech. Report NMSU-CS-2005-006, New Mexico State University (2005)
11. Lin, F., Zhao, Y.: ASSAT: Computing Answer Sets of a Logic Program by SAT Solvers. In: AAAI-2002, Edmonton, Alberta, Canada, AAAI Press / MIT Press (2002)
12. Lee, J., Lifschitz, V.: Loop Formulas for Disjunctive Logic Programs. In: Proceedings of the Nineteenth International Conference on Logic Programming (ICLP-03), (2003) 451–465
13. Ferraris, P.: Answer Sets for Propositional Theories. `http://www.cs.utexas.edu/users/otto/papers/proptheories.ps` (2004)
14. 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
15. Gebser, M., Kaufmann, B., Neumann, A., Schaub, T.: Conflict-driven answer set solving. In: Twentieth International Joint Conference on Artificial Intelligence (IJCAI-07),(2007) 386–392
16. Lierler, Y., Maratea, M.: Cmodels-2: SAT-based Answer Set Solver Enhanced to Non-tight Programs. In: LPNMR-7. LNCS 2923
17. Lierler, Y.: Disjunctive Answer Set Programming via Satisfiability. In: LPNMR'05. LNCS 3662
18. Baral, C.: Knowledge Representation, Reasoning and Declarative Problem Solving. CUP (2003)
19. Gelfond, M., Lifschitz, V.: Classical Negation in Logic Programs and Disjunctive Databases. NGC **9** (1991) 365–385
20. Faber, W.: Decomposition of Nonmonotone Aggregates in Logic Programming. WLP 2006 164–171
21. Faber, W.: Unfounded Sets for Disjunctive Logic Programs with Arbitrary Aggregates. In: LPNMR'05. LNCS 3662
22. Van Gelder, A., Ross, K., Schlipf, J.: The Well-Founded Semantics for General Logic Programs. JACM **38**(3) (1991) 620–650
23. Leone, N., Rullo, P., Scarcello, F.: Disjunctive Stable Models: Unfounded Sets, Fixpoint Semantics and Computation. Information and Computation **135**(2) (1997) 69–112
24. Ben-Eliyahu, R., Dechter, R.: Propositional Semantics for Disjunctive Logic Programs. AMAI **12** (1994) 53–87
25. Faber, W., Leone, N.: On the Complexity of Answer Set Programming with Aggregates. In: Logic Programming and Nonmonotonic Reasoning — 9th International Conference, LPNMR'07, Tempe, Arizona, 2007, Proceedings. (2007) To appear.
26. Leone, N., Pfeifer, G., Faber, W., Eiter, T., Gottlob, G., Perri, S., Scarcello, F.: The DLV System for Knowledge Representation and Reasoning. ACM TOCL **7**(3) (2006) 499–562
27. Faber, W.: Enhancing Efficiency and Expressiveness in Answer Set Programming Systems. PhD thesis, TU Wien (2002)
28. Calimeri, F., Faber, W., Leone, N., Pfeifer, G.: Pruning Operators for Disjunctive Logic Programming Systems. Fundamenta Informaticae **71**(2–3) (2006) 183–214
29. Koch, C., Leone, N., Pfeifer, G.: Enhancing Disjunctive Logic Programming Systems by SAT Checkers. Artificial Intelligence **15**(1–2) (2003) 177–212