TLP **7** (3): 355–375, 2007. © 2007 Cambridge University Press
doi:10.1017/S1471068406002936 Printed in the United Kingdom

TECHNICAL NOTE

A Constructive semantic characterization of aggregates in answer set programming

TRAN CAO SON and ENRICO PONTELLI

Department of Computer Science, New Mexico State University, NM, USA
(*e-mail:* {tson,epontell}@cs.nmsu.edu)

submitted 17 June 2005; revised 27 October 2005; accepted 8 February 2006

Abstract

This technical note describes a monotone and continuous fixpoint operator to compute the answer sets of programs with aggregates. The fixpoint operator relies on the notion of *aggregate solution*. Under certain conditions, this operator behaves identically to the three-valued immediate consequence operator Φ_P^{aggr} for aggregate programs, independently proposed in Pelov (2004) and Pelov *et al.* (2004). This operator allows us to closely tie the computational complexity of the answer set checking and answer sets existence problems to the cost of checking a solution of the aggregates in the program. Finally, we relate the semantics described by the operator to other proposals for logic programming with aggregates.

KEYWORDS: aggregates, answer set programming, semantics

1 Introduction

Several semantic characterizations of answer sets of logic programs with aggregates have been proposed over the years (Kemp and Stuckey 1991; Mumick *et al.* 1990; Gelfond 2002; Faber *et al.* 2004; Pelov *et al.* 2004). Most of these proposals have their roots in the answer set semantics of normal logic programs without aggregates (Gelfond and Lifschitz 1988). Nevertheless, it is known that a straightforward generalization of the definition of answer sets to programs with aggregates may yield *non-minimal* and/or unintuitive answer sets. Consider the following example.

Example 1
Let P be the program

$$\begin{array}{lll} p(1) \leftarrow & p(2) \leftarrow & p(3) \leftarrow \\ p(5) \leftarrow q & q \leftarrow \textsc{Sum}(\{X \mid p(X)\}) > 10 & \end{array}$$

The aggregate $\textsc{Sum}(\{X \mid p(X)\}) > 10$ is satisfied by any interpretation M of P where the sum of X such that $p(X)$ is true in M is greater than 10.

A straightforward extension of the original definition of answer sets (Gelfond and Lifschitz 1988) defines M to be an answer set of P if and only if M is the minimal model of the reduct P^M, where P^M is the program obtained by (*i*) removing from P all the rules containing in their body at least an aggregate or a negation-as-failure literal which is false in M; and (*ii*) removing all the aggregates and negation-as-failure literals from the remaining rules. Effectively, this definition treats aggregates in the same fashion as negation-as-failure literals.

It is easy to see that for $A = \{p(1), p(2), p(3)\}$ and $B = \{p(1), p(2), p(3), p(5), q\}$,

$$P^A = \left\{ \begin{array}{l} p(1) \leftarrow \\ p(2) \leftarrow \\ p(3) \leftarrow \\ p(5) \leftarrow q \end{array} \right\} \qquad P^B = \left\{ \begin{array}{l} p(1) \leftarrow \\ p(2) \leftarrow \\ p(3) \leftarrow \\ p(5) \leftarrow q \\ q \leftarrow \end{array} \right\}$$

and A and B are minimal model of P^A and P^B respectively. Thus, both A and B are answer sets of P. As we can see, treating aggregates like negation-as-failure literals yields non-minimal answer sets. Accepting B as an answer set seems counter-intuitive, since $p(5)$ "supports" itself through the aggregate. □

Different approaches have been proposed to deal with this problem. Early works concentrate on finding syntactic (e.g., stratification (Mumick *et al.* 1990; Kemp and Stuckey 1991)) and semantic (e.g., monotonic aggregates (Ross and Sagiv 1997; Kemp and Stuckey 1991)) restrictions on aggregates which guarantee minimality, and often uniqueness, of answer sets.

In this technical note, we present a fixpoint operator that allows us to compute answer sets of normal logic programs with *arbitrary aggregates*. It is a straightforward extension of the Gelfond-Lifschitz definition, making use of the *same* notion of reduct as in Gelfond and Lifschitz (1988), and relying on a continuous fixpoint operator for computing selected minimal models of the reduct (corresponding to our notion of answer sets). This fixpoint operator is a natural extension of the traditional immediate consequence operator T_P to programs with aggregates. It takes into consideration the provisional answer set while trying to verify that it is an answer set. This fixpoint operator makes use of the notion of *aggregate solutions*, and it captures the *unfolding semantics* for normal logic programs with aggregates, originally proposed in Elkabani *et al.* (2004) and completely developed in Son *et al.* (2005). This semantics builds on the principle of unfolding of intensional set constructions, as developed in Dovier *et al.* (2001). This operator corresponds to the Φ_P^{aggr} operator proposed in Pelov *et al.* (2004) and Pelov (2004), when ultimate approximating aggregates are employed and 2-valued stable models are considered. In particular, the two operators are identical when they are applied to the construction of a correct answer set M.

The proposed fixpoint operator allows us also to easily demonstrate the existence of a large class of logic programs with aggregates (which includes recursively defined aggregates and non-monotone aggregates) for which the problems of answer set checking and of determining the existence of an answer set is in **P** and **NP** respectively. Finally, we relate our work to recently proposed semantics for programs with aggregates (Faber *et al.* 2004; Pelov *et al.* 2004; Son *et al.* 2005).

2 Preliminary definitions

2.1 Language syntax

Let us consider a signature $\Sigma_L = \langle \mathscr{F}_L \cup \mathscr{F}_{Agg}, \mathscr{V} \cup \mathscr{V}_l, \Pi_L \rangle$, where $\mathscr{F}_L$ is a collection of constants, $\mathscr{F}_{Agg}$ is a collection of unary function symbols, $\mathscr{V} \cup \mathscr{V}_l$ is a denumerable collection of variables (such that $\mathscr{V} \cap \mathscr{V}_l = \emptyset$), and Π_L is a collection of predicate symbols. In the rest of this paper, we will always assume that the set $\mathbb{Z}$ of the integers is a subset of $\mathscr{F}_L$ – i.e., there are distinct constants representing the integer numbers. We will refer to Σ_L as the *ASP signature*. We will also refer to $\Sigma_P = \langle \mathscr{F}_P, \mathscr{V} \cup \mathscr{V}_l, \Pi_P \rangle$ as the *program signature*, where $\mathscr{F}_P \subseteq \mathscr{F}_L$, $\Pi_P \subseteq \Pi_L$, and $\mathscr{F}_P$ is finite. We will denote with $\mathscr{H}_P$ the Σ_P-Herbrand universe, containing the ground terms built using symbols of $\mathscr{F}_P$, and with $\mathscr{B}_P$ the corresponding Σ_P-Herbrand base. An ASP-atom is an atom of the form $p(t_1, \ldots, t_n)$, where $t_i \in \mathscr{F}_P \cup \mathscr{V}$ and $p \in \Pi_P$; an ASP-literal is either an ASP-atom or the negation as failure (*not* A) of an ASP-atom. We will use the traditional notation $\{t_1, \ldots, t_k\}$ to denote an extensional set of terms, and the notation $\{\!\{t_1, \ldots, t_k\}\!\}$ to denote an extensional multiset (or bag) of terms.

Definition 1 (*Intensional Sets and Multisets*)
An *intensional set* is a set of the form $\{X \mid p(X_1, \ldots, X_k)\}$ where $X \in \mathscr{V}_l$, X_i's are variables or constants (in $\mathscr{F}_P$), $\{X_1, \ldots, X_k\} \cap \mathscr{V}_l = \{X\}$, and p is a k-ary predicate in Π_P. Similarly, an *intensional multiset* is a multiset of the form

$$\{\!\{X \mid \exists Z_1, \ldots, Z_r.\, p(Y_1, \ldots, Y_m)\}\!\}$$

where $\{X, Z_1, \ldots, Z_r\} \subseteq \mathscr{V}_l$, Y_i are variables or constants (of $\mathscr{F}_P$), $\{Y_1, \ldots, Y_m\} \cap \mathscr{V}_l = \{X, Z_1, \ldots, Z_r\}$, and $X \notin \{Z_1, \ldots, Z_r\}$. We call X the *grouped variable*, $Z_1, \ldots, Z_r$ the *local variables*, and p the *grouped predicate* of the intensional set/multiset.

Intuitively, in an intensional multiset, we collect the values of X for which $p(Y_1, \ldots, Y_m)$ is true, under the assumptions that the variables $Z_1, \ldots, Z_r$ are locally, existentially quantified. Multiple occurrences of the same value of X can appear. For example, if $p(X, Z)$ is true for $X = 1, Z = 2$ and $X = 1, Z = 3$, then the multiset $\{\!\{X \mid \exists Z.p(X, Z)\}\!\}$ will correspond to $\{\!\{1, 1\}\!\}$. Definition 1 can be easily extended to allow more complex types of sets, e.g., sets with a tuple as the grouped variable and sets with conjunctions of atoms as property of the intensional construction.

Observe that the variables from $\mathscr{V}_l$ are used exclusively as grouped or local variables in defining intensional sets/multisets, and they cannot occur anywhere else.

We write $\bar{X}$ to denote $X_1, \ldots, X_n$.

Definition 2 (*Aggregate Terms/Atoms*)

- An *aggregate term* is of the form *aggr*(s), where s is an intensional set/multiset, and $aggr \in \mathscr{F}_{Agg}$ (called the *aggregate function*).
- An *aggregate atom* has the form *aggr*(s) op *Result*, where op is a relational operator in the set $\{=, \neq, <, >, \leqslant, \geqslant\}$ and $Result \in \mathscr{V} \cup (\mathbb{Z} \cap \mathscr{F}_P)$ – i.e., it is either a variable or a numeric constant.

https://doi.org/10.1017/S1471068406002936 Published online by Cambridge University Press

In our examples, we will focus on the traditional aggregate functions, e.g., COUNT, SUM, MIN. For an aggregate atom ℓ of the form $aggr(s)$ op $Result$, we refer to the grouped variable and predicate of s as the grouped variable and predicate of ℓ. The set of ASP-atoms constructed from the grouped predicate of ℓ and the terms in $\mathscr{H}_P$ is denoted by $\mathscr{H}(\ell)$.

Definition 3 (*ASP^A Rule/Program*)

- An ASP^A rule is of the form

$$A \leftarrow C_1, \ldots, C_m, A_1, \ldots, A_n, not\ B_1, \ldots, not\ B_k \qquad (1)$$

  where A, $A_1, \ldots, A_n$, $B_1, \ldots, B_k$ are ASP-atoms, while $C_1, \ldots, C_m$ are aggregate atoms ($m \geqslant 0$, $n \geqslant 0$, $k \geqslant 0$).
- An ASP^A program is a finite collection of ASP^A rules.

For an ASP^A rule r of the form (1), $head(r)$, $agg(r)$, $pos(r)$, and $neg(r)$ denote respectively A, $\{C_1, \ldots, C_m\}$, $\{A_1, \ldots, A_n\}$, and $\{B_1, \ldots, B_k\}$. Furthermore, $body(r)$ denotes the right-hand side of the rule r.

Observe that grouped and local variables in an aggregate atom ℓ have a scope limited to ℓ. As such, given an ASP^A rule, it is always possible to rename such variables occurring in the aggregate atoms $C_1, \ldots, C_m$ apart, so that they are pairwise different. Observe also that the grouped and local variables represent the only occurrences of variables from $\mathscr{V}_l$, thus they will not occur in A, $A_1, \ldots, A_n$, $B_1, \ldots, B_k$. For this reason, without loss of generality, whenever we refer to an ASP^A rule r, we will assume that the grouped and local variables of its aggregate atoms are pairwise different and do not appear in the rest of the rule.

Given a term, literal, aggregate atom, rule α, let us denote with $fvars(\alpha)$ the set of variables from $\mathscr{V}$ present in α. The entity α is ground if $fvars(\alpha) = \emptyset$.

A ground substitution σ is a set $\{X_1/c_1, \ldots, X_n/c_n\}$ where X_i's are distinct variables from $\mathscr{V}$ and c_i's are constants in $\mathscr{F}_P$. For an ASP-atom p (an aggregate atom ℓ), $p\sigma$ ($\ell\sigma$) denotes the ASP-atom (the aggregate atom) which is obtained from p (ℓ) by simultaneously replacing every occurrence of X_i with c_i.

Let r be a rule of the form (1) and $\{X_1, \ldots, X_t\}$ be the set of free variables occurring in A, $C_1, \ldots, C_m$, $A_1, \ldots, A_n$, and $B_1, \ldots, B_k$ – i.e., $fvars(r) = \{X_1, \ldots, X_t\}$. Let σ be a ground substitution $\{X_1/c_1, \ldots, X_t/c_t\}$. The ground instance of r w.r.t. σ, denoted by $r\sigma$, is the ground rule obtained from r by simultaneously replacing every occurrence of X_i with c_i.

By $ground(r)$ we denote the set of all ground instances of the rule r. For a program P, the set of all ground instances of the rules in P, denoted by $ground(P)$, is called the ground instance of P, i.e., $ground(P) = \bigcup_{r \in P} ground(r)$.

2.2 Aggregate solutions

In this subsection we provide the basic definitions of satisfaction and solution of an aggregate atom.

Definition 4 (*Interpretation Domain and Interpretation*)
The domain of our interpretations is the set $\mathscr{D} = \mathscr{H}_P \cup 2^{\mathscr{H}_P} \cup \mathscr{M}(\mathscr{H}_P)$, where $2^{\mathscr{H}_P}$ is the set of (finite) subsets of $\mathscr{H}_P$ and $\mathscr{M}(\mathscr{H}_P)$ is the set of finite multisets of elements

https://doi.org/10.1017/S1471068406002936 Published online by Cambridge University Press

from $\mathscr{H}_P$. An interpretation I is a pair $\langle \mathscr{D}, (\cdot)^I \rangle$, where $(\cdot)^I$ is a function that maps ground terms to elements of $\mathscr{D}$ and ground atoms to truth values.

Definition 5 (*Interpretation Function*)
Given a constant c, its interpretation c^I is equal to c.

Given a ground intensional set s of the form $\{X \mid p(\bar{X})\}$, its interpretation s^I is the set $\{a_1, \ldots, a_n\} \subseteq \mathscr{H}_P$, where $(p(\bar{X}))\{X/a_i\}^I$ is equal to true for $1 \leqslant i \leqslant n$, and no other value for X has such property.

Given a ground intensional multiset s of the form $\{\!\{X \mid \exists \bar{Z}.p(\bar{X}, \bar{Z})\}\!\}$, its interpretation s^I is the multiset $\{\!\{a_1, \ldots, a_k\}\!\} \in \mathscr{M}(\mathscr{H}_P)$ where, for each $1 \leqslant i \leqslant k$, there is a ground substitution η_i for $\bar{Z}$ such that $p(\bar{X}, \bar{Z})\gamma_i^I$ is true for $\gamma_i = \eta_i \cup \{X/a_i\}$, and no other elements satisfy this property.

Given the aggregate term $aggr(s)$, its interpretation is $aggr^I(s^I)$, where

$$aggr^I : 2^{\mathscr{H}_P} \cup \mathscr{M}(\mathscr{H}_P) \rightarrow \mathbb{Z}.$$

Given a ground ASP^A atom $p(t_1, \ldots, t_n)$, its interpretation is $p^I(t_1^I, \ldots, t_n^I)$, where $p^I : \mathscr{D}^n \rightarrow \{\texttt{true}, \texttt{false}\}$.

Given a ground aggregate atom ℓ of the form $aggr(s)$ op $Result$, its interpretation ℓ^I is true if $op^I(aggr(s)^I, Result^I)$ is true, where $\text{op}^I : \mathbb{Z} \times \mathbb{Z} \rightarrow \{\texttt{true}, \texttt{false}\}$.

We will assume that the traditional aggregate functions are interpreted in the usual way. E.g., $\textsc{Sum}^I$ is the function that maps a set/multiset of numbers to its sum, and $\textsc{Count}^I$ is the function that maps a set/multiset of constants to its cardinality. Similarly, we assume that the traditional relational operators (e.g., $\leqslant$, $\neq$) are interpreted according to their traditional meaning.

Given a literal *not p*, its interpretation $(not\ p)^I$ is true (false) iff p^I is false (true).

Given an atom, literal, or aggregate atom ℓ, we will denote with $I \models \ell$ the fact that ℓ^I is true.

Definition 6 (*Rule Satisfaction*)
I satisfies the body of a ground rule r (denoted by $I \models body(r)$), if

(i) $pos(r) \subseteq I$;
(ii) $neg(r) \cap I = \emptyset$;
(iii) $I \models c$ for every $c \in agg(r)$.

I satisfies a ground rule r if $I \models head(r)$ or $I \not\models body(r)$.

Having specified when an interpretation satisfies an aggregate atom or a ASP^A rule, we can define the notion of model of a program.

Definition 7 (*Model*)
Let P be an ASP^A program. An interpretation M is a *model* of P if M satisfies every rule in $ground(P)$.

In our view of interpretations, we assume that the interpretation of the aggregate functions and relational operators is fixed. In this perspective, we will still be able to keep the traditional view of interpretations as subsets of $\mathscr{B}_P$.

https://doi.org/10.1017/S1471068406002936 Published online by Cambridge University Press

Definition 8
M is a *minimal model* of P if M is a model of P and there is no proper subset of M which is also a model of P.

We will now define a notion called *aggregate solution*. Observe that the satisfaction of an ASP-atom a is *monotonic*, in the sense that if $I \models a$ and $I \subseteq I'$ then we have that $I' \models a$. On the other hand, the satisfaction of an aggregate atom is possibly non-monotonic, i.e., $I \models \ell$ and $I \subseteq I'$ do not necessarily imply $I' \models \ell$. For example, $\{p(1)\} \models \textsc{Sum}(\{X \mid p(X)\}) \neq 0$ but $\{p(1), p(-1)\} \not\models \textsc{Sum}(\{X \mid p(X)\}) \neq 0$. The notion of aggregate solution allows us to define an operator where the monotonicity of satisfaction of aggregate atoms is used in verifying an answer set.

Definition 9 (*Aggregate Solution*)
Let ℓ be a ground aggregate atom. An *aggregate solution* of ℓ is a pair $\langle S_1, S_2 \rangle$ of disjoint subsets of $\mathscr{H}(\ell)$ such that, for every interpretation I, if $S_1 \subseteq I$ and $S_2 \cap I = \emptyset$ then $I \models \ell$. $\mathscr{SOLN}(\ell)$ is the set of all the solutions of ℓ.

It is obvious that if $I \models \ell$ then $\langle I \cap \mathscr{H}(\ell), \mathscr{H}(\ell) \setminus I \rangle$ is a solution of ℓ. Let $S = \langle S_1, S_2 \rangle$ be an aggregate solution of an aggregate atom; we denote with $S.p$ and $S.n$ the two components S_1 and S_2 of the solution.

Example 2
Consider the aggregate atom $\textsc{Sum}(\{X \mid p(X)\}) > 10$ from the program in Example 1. This atom has a unique solution: $\langle \{p(1), p(2), p(3), p(5)\}, \emptyset \rangle$. On the other hand, the aggregate atom $\textsc{Sum}(\{X \mid p(X)\}) > 6$ has the following solutions:

$\langle \{p(3), p(5)\}, \emptyset \rangle$ $\langle \{p(3), p(5)\}, \{p(1), p(2)\} \rangle$
$\langle \{p(3), p(5)\}, \{p(1)\} \rangle$ $\langle \{p(3), p(5)\}, \{p(2)\} \rangle$
$\langle \{p(2), p(5)\}, \emptyset \rangle$ $\langle \{p(2), p(5)\}, \{p(1), p(3)\} \rangle$
$\langle \{p(2), p(5)\}, \{p(1)\} \rangle$ $\langle \{p(2), p(5)\}, \{p(3)\} \rangle$
$\langle \{p(1), p(2), p(5)\}, \emptyset \rangle$ $\langle \{p(1), p(2), p(5)\}, \{p(3)\} \rangle$
$\langle \{p(1), p(3), p(5)\}, \emptyset \rangle$ $\langle \{p(1), p(3), p(5)\}, \{p(2)\} \rangle$
$\langle \{p(1), p(2), p(3), p(5)\}, \emptyset \rangle$ $\langle \{p(2), p(3), p(5)\}, \emptyset \rangle$
$\langle \{p(2), p(3), p(5)\}, \{p(1)\} \rangle$

□

3 A fixpoint operator based on aggregate solutions

In this section, we construct the semantics for ASP^A programs, through the use of a monotone and continuous fixpoint operator. For the sake of simplicity, we will assume that programs, ASP-atoms, and aggregate atoms referred to in this section are ground[1]. As we will show in Section 4.3, this fixpoint operator behaves as the 3-valued immediate consequence operator of Pelov *et al.* (2004) under certain conditions (e.g., use of ultimate approximating aggregates).

[1] A program P with variables can be viewed as a shorthand for $ground(P)$.

https://doi.org/10.1017/S1471068406002936 Published online by Cambridge University Press

Definition 10 (*Reduct for ASP^A Programs*)
Let P be an ASP^A program and let M be an interpretation. The reduct of P with respect to M, denoted by ${}^MP$, is defined as

$${}^MP = \{head(r) \leftarrow pos(r), agg(r) \mid r \in ground(P), \;\; M \cap neg(r) = \emptyset\}$$

Observe that, for a program P without aggregates, the process of checking whether M is an answer set (Gelfond and Lifschitz 1988) requires first computing the Gelfond-Lifschitz reduct of P w.r.t. M (P^M), and then verifying that M is the least model of P^M. This second step is performed by using the van Emden-Kowalski operator T_{P^M} to regenerate M, by computing the least fixpoint of T_{P^M}. I.e., we compute the sequence $M_0, M_1, M_2, \ldots$ where $M_0 = \emptyset$ and $M_{i+1} = T_{P_M}(M_i)$. In every step of regenerating M, an atom a is added to M_{i+1} iff there is a rule in P^M whose head is a and whose body is contained in M_i. This process is monotonic, in the sense that, if a is added to M_i, then a will belong to M_j for all $j \geqslant i$.

Our intention is to define a T_P-like operator for programs with aggregates. Specifically, we would like to verify that M is an answer set of P by generating a monotone sequence of interpretations $M_0 \subseteq M_1 \subseteq \ldots \subseteq M_n \subseteq \ldots = M$. To do so, we need to specify when a rule of ${}^MP$ can be used, i.e., when an ASP/aggregate atom is considered satisfied by M_i. We also need to ensure that, at each step $i+1$, M_{i+1} will *still* satisfy all ASP-atoms and the aggregate atoms that are satisfied by M_i.

This observation leads us to define the notion of *conditional satisfaction* of an atom (ASP-atom or aggregate atom) over a pair of sets of atoms (I, M) – where I is an interpretation generated at some step of the verification process, and M is the answer set that needs to be verified.

Definition 11 (*Conditional Satisfaction*)
Let ℓ be an ASP-atom or an aggregate atom, and I, M be two interpretations[2]. We define the *conditional satisfaction* of ℓ w.r.t. I and M, denoted by $(I, M) \models \ell$, as:

- if ℓ is ASP-atom, then $(I, M) \models \ell \;\Leftrightarrow\; I \models \ell$
- if ℓ is an aggregate atom, then

$$(I, M) \models l \;\Leftrightarrow\; \langle I \cap M \cap \mathscr{H}(\ell),\; \mathscr{H}(\ell) \setminus M\rangle \textit{ is a solution of } \ell$$

The first bullet says that an ASP-atom is satisfied by a pair (I, M) if it is satisfied by I. The second bullet states that I contains enough information of M to guarantee that any successive expansion of I towards M will satisfy the aggregate. Conditional satisfaction is naturally extended to conjunctions of atoms. The following lemma trivially holds.

Lemma 1
Let ℓ be an ASP-atom or an aggregate atom and I, J, M be interpretations such that $I \subseteq J$. Then, $(I, M) \models \ell$ implies $(J, M) \models \ell$.

We are now ready to define the consequence operator for ASP^A programs.

[2] Recall that an interpretation is a set of atoms in $\mathscr{B}_P$.

https://doi.org/10.1017/S1471068406002936 Published online by Cambridge University Press

Definition 12 (*Consequence Operator*)
Let P be an ASP^A program and M be an interpretation. We define the consequence operator on P and M, called K_M^P, as

$$K_M^P(I) = \{\, head(r) \;\mid\; r \in {}^{M}P \;\wedge\; (I, M) \models body(r)\,\}$$

for every interpretation I of P.

By definition, we have that $K_M^P(I) = T_P(I)$ for definite programs without aggregate atoms. Thus, K_M^P can be viewed as an extension of T_P to the class of programs with aggregates. The following lemma is a consequence of Lemma 1.

Lemma 2
Let P be a program and M be an interpretation. Then, K_M^P is monotone and continuous over the lattice $\langle 2^{\mathscr{B}_P}, \subseteq \rangle$.

The above lemma allows us to conclude that the least fixpoint of K_M^P, denoted by $lfp(K_M^P)$, exists and it is equal to $K_M^P \uparrow \omega$. Here, $K_M^P \uparrow n$ denotes

$$\underbrace{K_M^P(K_M^P(\ldots(K_M^P}_{n\text{—}times\ K_M^P}(\emptyset)\ldots)))$$

and $K_M^P \uparrow \omega$ denotes $\lim_{n\to\infty} K_M^P \uparrow n$. We are now ready to define the concept of *answer set* of an ASP^A program.

Definition 13 (*Fixpoint Answer Set*)
Let P be an ASP^A program and let M be an interpretation. M is a *fixpoint answer set* of P iff $M = lfp(K_M^P)$.

Whenever it is clear from the context, we will simply talk about *answer sets* of P instead of fixpoint answer sets.

Example 3
Let us continue with the program P from Example 1. Since P does not contain negation-as-failure literals, ${}^{M}P = P$ for any interpretation M of P. Any answer set of P must contain $p(1)$, $p(2)$, and $p(3)$. We will now show that $A = \{p(1), p(2), p(3)\}$ is the unique fixpoint answer set of P. It is easy to see that

$$\begin{array}{lcl} K_A^P \uparrow 0 & = & \emptyset \\ K_A^P \uparrow 1 & = & K_A^P(K_A^P \uparrow 0) = \{p(1), p(2), p(3)\} \\ K_A^P \uparrow 2 & = & \{p(1), p(2), p(3)\} = K_A^P \uparrow 1 \end{array}$$

Thus, A is indeed a fixpoint answer set of P.

Let us consider $B = \{p(1), p(2), p(3), p(5), q\}$. We have that ${}^{B}P = P$ and it is easy to verify that $lfp(K_B^P) = \{p(1), p(2), p(3)\}$. Therefore, B is not a fixpoint answer set of P. It is easy to check that no proper superset of A is a fixpoint answer set of P, i.e., A is the unique answer set of P. □

In the next example, we show how this definition works when the programs contain negation-as-failure literals.

https://doi.org/10.1017/S1471068406002936 Published online by Cambridge University Press

Example 4
Let P be the program[3]:

$$\begin{array}{lcl} p(a) & \leftarrow & \textsc{Count}(\{X \mid p(X)\}) > 0 \\ p(b) & \leftarrow & not\ q \\ q & \leftarrow & not\ p(b) \end{array}$$

We will show now that the program has two answer sets $A = \{q\}$ and $B = \{p(b), p(a)\}$. We have that

- ${}^{A}P$ consists of the first rule and the fact q. The verification that A is an answer set of P is shown next.

$$\begin{array}{lcl} K_A^P \uparrow 0 & = & \emptyset \\ K_A^P \uparrow 1 & = & K_A^P(K_A^P \uparrow 0) = \{q\} \\ K_A^P \uparrow 2 & = & \{q\} = K_A^P \uparrow 1 \end{array}$$

  $p(a)$ cannot belong to $K_A^P \uparrow 1$ since $\langle \emptyset, \emptyset \rangle$ is not a solution of the aggregate atom $\textsc{Count}(\{X \mid p(X)\}) > 0$.
- ${}^{B}P$ consists of the first rule and the fact $p(b)$.

$$\begin{array}{lcl} K_B^P \uparrow 0 & = & \emptyset \\ K_B^P \uparrow 1 & = & K_B^P(K_B^P \uparrow 0) = \{p(b)\} \\ K_B^P \uparrow 2 & = & \{p(b), p(a)\} \\ K_B^P \uparrow 3 & = & \{p(b), p(a)\} = K_B^P \uparrow 2 \end{array}$$

  $p(a)$ belongs to $K_B^P \uparrow 2$ since $\langle \{p(b)\}, \emptyset \rangle$ is a solution of the aggregate atom $\textsc{Count}(\{X \mid p(X)\}) > 0$.

It is easy to see that P does not have any other answer sets. □

4 Related work and discussion

In this section, we will relate our proposal to the unfolding semantics presented in (Son *et al.* 2005) and to two other recently proposed semantics for programs with aggregates[4] – i.e., the ultimate stable model semantics (Pelov *et al.* 2003; Pelov *et al.* 2004; Pelov 2004) and the minimal answer set semantics (Faber *et al.* 2004). We will also investigate some of the computational complexity issues related to determining the fixpoint answer sets of ASP^A programs.

4.1 Equivalence of fixpoint semantics and unfolding semantics

We will show that the notion of fixpoint answer set corresponds to the *unfolding semantics* presented in Son *et al.* (2005). To make this note self-contained, let us

[3] We would like to thank Vladimir Lifschitz for providing us this example.
[4] A detailed comparison between the semantics in (Son *et al.* 2005) and earlier proposals for programs with aggregates can be found in the same report.

recall the basic definition of the unfolding semantics. For a ground aggregate atom c and an interpretation M, let

$$\mathscr{S}(c, M) = \{S_c \mid S_c \in \mathscr{SOLN}(c),\ S_c.p \subseteq M,\ S_c.n \cap M = \emptyset\}$$

Intuitively, $\mathscr{S}(c, M)$ is the set of solutions of c which are satisfied by M. For a solution $S_c \in \mathscr{S}(c, M)$, the unfolding of c in M w.r.t. S_c is the conjunction $\bigwedge_{a \in S_c.p} a$. We say that c' is an unfolding of c with respect to M if c' is an unfolding of c in M with respect to some $S_c \in \mathscr{S}(c, M)$. When $\mathscr{S}(c, M) = \emptyset$, we say that $false$ is the unfolding of c in M. The unfolding of a rule $r \in ground(P)$ with respect to M is the set of rules $unfolding(r, M)$ defined as follows:

1. If $neg(r) \cap M \neq \emptyset$ or there is some $c \in agg(r)$ such that $false$ is the unfolding of c in M then $unfolding(r, M) = \emptyset$;
2. If $neg(r) \cap M = \emptyset$ and $false$ is not the unfolding of c for every $c \in agg(r)$, then $r' \in unfolding(r, M)$ where
   (a) $head(r') = head(r)$
   (b) $neg(r') = neg(r)$
   (c) there is a sequence of aggregate solutions $\langle S_c \rangle_{c \in agg(r)}$ for the aggregates in $agg(r)$, such that $S_c \in \mathscr{S}(c, M)$ for every $c \in agg(r)$ and $pos(r') = pos(r) \cup \bigcup_{c \in agg(r)} S_c.p$.

For a program P, $unfolding(P, M)$ denotes the set of unfolding rules of $ground(P)$ w.r.t. M. M is an ASP^A-answer set of P iff M is an answer set of $unfolding(P, M)$.

This notion of unfolding derives from the work on unfolding of intensional sets (Dovier *et al.* 2001), and has been independently described in Pelov *et al.* (2003).

Lemma 3
Let c be an aggregate atom, let M be an interpretation, and let S_c be a solution of c such that $S_c \in \mathscr{S}(c, M)$. Then, $\langle S_c.p, \mathscr{H}(c) \setminus M \rangle$ is a solution of c.

Proof
Let us consider an interpretation I such that $S_c.p \subseteq I$ and $I \cap (\mathscr{H}(c) \setminus M) = \emptyset$. Because $S_c.n \subseteq \mathscr{H}(c) \setminus M$, $I \cap S_c.n = \emptyset$. Since S_c is a solution, $I \models c$. Since this holds for every interpretation I satisfying $S_c.p \subseteq I$ and $I \cap (\mathscr{H}(c) \setminus M) = \emptyset$, we have that $\langle S_c.p, \mathscr{H}(c) \setminus M \rangle$ is a solution of c. □

Lemma 4
Let $R = unfolding(P, M)$. Then $T_R \uparrow i = K_M^P \uparrow i$ for $i \geqslant 0$.

Proof
Let us prove the result by induction on i.
Base: for $i = 0$, we have that $T_R \uparrow 0 = \emptyset = K_M^P \uparrow 0$, and the result is obviously true. Let us consider the case $i = 1$.

- Let $p \in T_R \uparrow 1 = \{\ell \mid (\ell \leftarrow) \in R\}$. If $p \leftarrow$ is a fact in P, then it is also a fact in ${}^M P$. This means that $p \leftarrow$ is an element of ${}^M P$, and thus p is in $K_M^P \uparrow 1$. Otherwise, there is a rule r in P, such that

https://doi.org/10.1017/S1471068406002936 Published online by Cambridge University Press

- $head(r) = p$;
- $pos(r) = \emptyset$;
- $neg(r) \cap M = \emptyset$; and
- for each $\ell \in agg(r)$ we have that there exists a solution of ℓ of the form $\langle \emptyset, J \rangle$ such that $M \cap J = \emptyset$.

The rule $p \leftarrow agg(r)$ is a rule in ${}^{M}P$. From Lemma 3 we can conclude that $(\emptyset, M) \models agg(r)$, thus ensuring that $p \in K_M^P \uparrow 1$.

- Let $p \in K_M^P \uparrow 1$. Thus, there exists a rule $r' \in {}^{M}P$ such that $(\emptyset, M) \models body(r)$ and $head(r') = p$. This means that there is a rule $r \in P$ such that
  - $head(r) = head(r') = p$;
  - $M \cap neg(r) = \emptyset$;
  - $pos(r) = \emptyset$; and
  - $agg(r) = agg(r')$.

  Since $(\emptyset, M) \models agg(r)$, we have that, for each $c \in agg(r)$, $\langle \emptyset, \mathscr{H}(c) \setminus M \rangle$ is a solution of c. This means that the rule $p \leftarrow$ is in $unfolding(P, M)$. This also means that $p \in T_R \uparrow 1$.

Step: Let us assume that the result holds for $i \leqslant k$ and consider the iteration $k+1$.

- Let $p \in T_R \uparrow (k+1)$ and $p \notin T_R \uparrow k$. Thus, there is a rule r' in R such that
  - $head(r') = p$; and
  - $pos(r') \subseteq T_R \uparrow k$.

  This implies that there is a rule $r \in P$ such that
  - $head(r) = p$;
  - $pos(r) \subseteq T_R \uparrow k$;
  - $M \cap neg(r) = \emptyset$; and
  - for each $c \in agg(r)$, there is a solution S_c s.t. $S_c.p \subseteq T_R \uparrow k$ and $M \cap S_c.n = \emptyset$.

  This also means that $p \leftarrow pos(r), agg(r)$ is a rule in ${}^{M}P$.
  We already know that $pos(r) \subseteq K_M^P \uparrow k$. Now we wish to show that $(K_M^P \uparrow k, M) \models agg(r)$. Lemma 3 shows that, for each $c \in agg(r)$, $\langle S_c.p, \mathscr{H}(c) \setminus M \rangle$ is a solution of c. This allows us to conclude that $p \in K_M^P \uparrow (k+1)$.
- Let $p \in K_M^P \uparrow (k+1)$ and $p \notin K_M^P \uparrow k$. This means that there is a rule r' in ${}^{M}P$ such that
  - $head(r') = p$;
  - $pos(r') \subseteq K_M^P \uparrow k$; and
  - $(K_M^P \uparrow k, M) \models body(r')$

  This also means that there is a rule r in P such that
  - $head(r) = head(r') = p$;
  - $agg(r) = agg(r')$;
  - $pos(r) = pos(r')$;
  - $neg(r) \cap M = \emptyset$; and
  - for each $c \in agg(r)$, $S_c = \langle K_M^P \uparrow k \cap M \cap \mathscr{H}(c), \mathscr{H}(c) \setminus M \rangle$ is a solution of c.

  This means that there is a rule r'' in $unfolding(P, M)$ such that:
  - $head(r'') = p$

https://doi.org/10.1017/S1471068406002936 Published online by Cambridge University Press

- $pos(r'') = pos(r) \cup \bigcup_{c \in aggr} S_c.p$
  Since each $S_c.p \subseteq K_M^P \uparrow k = T_R \uparrow k$ for each $c \in agg(r)$ and $pos(r) \subseteq K_M^P \uparrow k = T_R \uparrow k$, we have that $p \in T_R \uparrow (k+1)$.

□

Theorem 1
Let P be a program with aggregates. M is an answer set of $unfolding(P, M)$ iff M is a fixpoint answer set of P.

Proof
Let $R = unfolding(P, M)$. We have that M is an answer set of P iff $M = T_R \uparrow \omega$ iff $M = K_M^P \uparrow \omega$ (Lemma 4). □

The results from Son *et al.* (2005) and Theorem 1 provide us a direct connection between fixpoint answer sets and other semantics for logic programs with aggregates.

4.2 *Faber* et al.*'s Minimal Model Semantics*

The notion of answer set proposed in (Faber *et al.* 2004) is based on a new notion of reduct, defined as follows. Given a program P and a set of ASP-atoms M, the *reduct of P with respect to M*, denoted by $\Gamma(M, P)$, is obtained by removing from $ground(P)$ those rules whose body cannot be satisfied by M. In other words, $\Gamma(M, P) = \{r \mid r \in ground(P), M \models body(r)\}$.

Definition 14 (*FLP-answer set (Faber et al. 2004)*)
For a program P, M is an *FLP-answer set* of P if it is a minimal model of $\Gamma(M, P)$.

The following theorem derives directly from Theorem 1 and Son *et al.* (2005).

Theorem 2
Let P be a program with aggregates. If M is a fixpoint answer set, then M is an FLP-answer set of P.

Observe that there are cases where FLP-answer sets are not fixpoint answer sets.

Example 5
Consider the program P where

$$\begin{array}{lcl} p(1) & \leftarrow & \textsc{Sum}(\{X \mid p(X)\}) \geqslant 0 \\ p(-1) & \leftarrow & p(1) \\ p(1) & \leftarrow & p(-1) \end{array}$$

It can be checked that $M = \{p(1), p(-1)\}$ is an FLP-answer set of P. It is possible to show that $\textsc{Sum}(\{X \mid p(X)\}) \geqslant 0$ has the following solutions: $\langle \emptyset, \{p(1), p(-1)\}\rangle$, $\langle \{p(1)\}, \{p(-1)\}\rangle$, $\langle \{p(1)\}, \emptyset\rangle$, and $\langle \{p(1), p(-1)\}, \emptyset\rangle$.

We have that $K_M^P(\emptyset) = \emptyset$ since $\langle \emptyset, \emptyset\rangle$ is not a solution of $\textsc{Sum}(\{X \mid p(X)\}) \geqslant 0$. This implies that $lfp(K_M^P) = \emptyset$. Thus, M is not a fixpoint answer set of P. It can be easily verified that P does not have any fixpoint answer set. □

https://doi.org/10.1017/S1471068406002936 Published online by Cambridge University Press

Remark 1
If we replace in P the rule $p(1) \leftarrow \text{SUM}(\{X \mid p(X)\}) \geqslant 0$ with the intuitively equivalent SMODELS weight constraint rule

$$p(1) \leftarrow 0[p(1) = 1, p(-1) = -1]$$

we obtain a program that does not have answer sets in SMODELS.

The above example shows that our characterization differs from (Faber *et al.* 2004). Our definition is closer to SMODELS' understanding of aggregates.

4.3 Approximation Semantics for Logic Programs with Aggregates

The work of Pelov *et al.* (Pelov *et al.* 2003; Pelov 2004; Pelov *et al.* 2004) contains an elegant generalization of several semantics of logic programs to logic programs with aggregates. The key idea in this work is the use of approximation theory in defining several semantics for logic programs with aggregates (e.g., two-valued semantics, ultimate three-valued stable semantics, three-valued stable model semantics). In particular, in (Pelov *et al.* 2004), the authors describe a fixpoint operator, called Φ_P^{appr}, operating on 3-valued interpretations and parameterized by the choice of approximating aggregates.

It is possible to show the following results:

- Whenever the approximating aggregate used in Φ_P^{appr} is the *ultimate approximating aggregate* (Pelov *et al.* 2004), then the fixpoint semantics defined by the operator K_M^P coincides with the two-valued stable model semantics defined by the operator Φ_P^{appr}.
- It is possible to prove a stronger result, showing that, if $I \subseteq M$ then $K_M^P(I) = \Phi_P^{aggr,1}(I, M)$, where $\Phi_P^{aggr,1}(I, M)$ denotes the first component of $\Phi_P^{aggr}(I, M)$. In other words, when ultimate approximating aggregates are employed and M is an answer set, then the fixpoint operator of Pelov *et al.* and K_M^P behave identically.

We will prove next the first of these two results. The proof of the second result (kindly contributed by one of the anonymous reviewers) can be found in Appendix A. We will make use of the translation of logic programs with aggregates to normal logic programs, denoted by *tr*, described in (Pelov *et al.* 2003). The translation in Pelov *et al.* (2003) and the unfolding described in the previous subsection are similar[5].

For the sake of completeness, we will review the translation of Pelov *et al.* (2003), presented using the notation of our paper. Given a ground logic program with aggregates P, $tr(P)$ denotes the ground normal logic program obtained after the translation. The process begins with the translation of each aggregate atom ℓ of the form $aggr(s)$ op $Result$ into a disjunction $tr(\ell) = \bigvee F_{(s_1,s_2)}^{\mathscr{H}(\ell)}$, where $s_1 \subseteq s_2 \subseteq \mathscr{H}(\ell)$,

[5] It should be noted that our translation builds on our previous work on semantics of logic programming with sets and aggregates (Dovier *et al.* 2001, 2003; Elkabani *et al.* 2004) and was independently developed w.r.t. the work in Pelov *et al.* (2003).

https://doi.org/10.1017/S1471068406002936 Published online by Cambridge University Press

and each $F^{\mathscr{H}(\ell)}_{(s_1,s_2)}$ is a conjunction of the form

$$\bigwedge_{l \in s_1} l \wedge \bigwedge_{l \in \mathscr{H}(\ell) \setminus s_2} not\ l$$

The construction of $tr(\ell)$ considers only the pairs (s_1, s_2) that satisfy the following condition: each interpretation I such that $s_1 \subseteq I$ and $\mathscr{H}(\ell) \setminus s_2 \cap I = \emptyset$ must satisfy ℓ. The translation $tr(P)$ is then created by replacing rules with disjunction in the body by a set of standard rules in a straightforward way. For example, the rule

$$a \leftarrow (b \vee c), d$$

is replaced by the two rules

$$a \leftarrow b, d \qquad\qquad a \leftarrow c, d$$

From the definitions of $tr(\ell)$ and of aggregate solutions, we have the following simple lemma:

Lemma 5
For every aggregate atom ℓ of the form $aggr(s)$ op $Result$, S is a solution of ℓ if and only if $F^{\mathscr{H}(\ell)}_{(S.p, \mathscr{H}(\ell) \setminus S.n)}$ is a disjunct in $tr(\ell)$.

We next show that fixed point answer sets of P are answer sets of $tr(P)$.

Lemma 6
For a program P, M is a fixpoint answer set of P iff M is an answer set of $tr(P)$.

Proof
Let M be an interpretation of P and $R = unfolding(P, M)$. We have that R is a positive program. Furthermore, let Q denote the result of the Gelfond-Lifschitz reduction of $tr(P)$ with respect to M, i.e., $Q = (tr(P))^M$. We will prove by induction on k that if M is an answer set of Q then $T_Q \uparrow k = T_R \uparrow k$ for every $k \geqslant 0$. The equation holds trivially for $k = 0$. Let us consider now the case for k, assuming that $T_Q \uparrow l = T_R \uparrow l$ for $0 \leqslant l < k$.

1. Consider $p \in T_Q \uparrow k$. This means that there exists some rule $r' \in Q$ such that $head(r') = p$ and $body(r') \subseteq T_Q \uparrow (k-1)$. $r' \in Q$ if and only if there exists some $r \in P$ such that $r' \in tr(r)$. Together with Lemma 5, we can conclude that there exists a sequence of aggregate solutions $\langle S_c \rangle_{c \in agg(r)}$ for the aggregate atoms in $body(r)$ such that $pos(r') = pos(r) \cup \bigcup_{c \in agg(r)} S_c.p$, and $(neg(r) \cup \bigcup_{c \in agg(r)} S_c.n) \cap M = \emptyset$. This implies that $r' \in R$. Together with the inductive hypothesis, we can conclude that $p \in T_R \uparrow k$.
2. Consider $p \in T_R \uparrow k$. This implies that there exists some rule $r' \in R$ such that $head(r') = p$ and $body(r') \subseteq T_R \uparrow (k-1)$. From the definition of R, we conclude that there exists some rule $r \in ground(P)$ and a sequence of aggregate solutions $\langle S_c \rangle_{c \in agg(r)}$ for the aggregate atoms in $body(r)$ such that $pos(r') = pos(r) \cup \bigcup_{c \in agg(r)} S_c.p$, and $(neg(r) \cup \bigcup_{c \in agg(r)} S_c.n) \cap M = \emptyset$. Using Lemma 5, we can conclude that $r' \in Q$. Together with the inductive hypothesis, we can conclude that $p \in T_Q \uparrow k$.

Similar arguments can be used to show that if M is an answer set of R, $T_Q \uparrow k = T_R \uparrow k$ for every $k \geqslant 0$, which means that M is an answer set of Q. □

In (Pelov *et al.* 2003), it is shown that answer sets of $tr(P)$ coincide with the *two-valued partial stable models* of P (defined by the operator Φ_P^{aggr}). This, together with the above lemma and Theorem 1, allows us to conclude the following theorem.

Theorem 3
For a program with aggregates P, M is an fixpoint answer set of P if and only if it is a fixpoint of the operator Φ_P^{aggr} of (Pelov *et al.* 2004).

4.4 Complexity considerations

We will now discuss the complexity of computing fixpoint answer sets. In what follows, we will assume that the program P is given and it is a ground program whose language is finite. By the *size* of a program, we mean the number of rules and atoms present in it, as in Faber *et al.* (2004). Observe that, in order to support the computation of the iterations of the K_M^P operator, we need the ability to determine whether a given $\langle I, J \rangle$ is a solution of an aggregate atom. For this reason, we classify programs with aggregates by the computational complexity of its aggregates. We define a notion, called C-*decidability*, where C denotes a complexity class in the complexity hierarchy, as follows.

Definition 15
Given an aggregate atom ℓ and an interpretation M, we say that ℓ is C-*decidable* if its truth value with respect to M can be decided by an oracle of the complexity C. A program P is called C-*decidable* if the aggregate atoms occurring in P are C-decidable.

It is easy to see that aggregate atoms built using the standard aggregate functions (Sum, Min, Max, Count, Avg) and relations ($=, \neq, \geqslant, >, \leqslant, <$) are polynomially decidable. The solution checking problem is defined as follows.

Definition 16 ((SCP) Solution Checking Problem)
Given an aggregate atom ℓ, its language extension $\mathscr{H}(\ell)$, and a pair of disjoint sets $I, J \subseteq \mathscr{H}(\ell)$, **Determine** whether $\langle I, J \rangle$ is a solution of ℓ.

We have the following lemma.

Lemma 7
The SCP is in **co-NP**^{C} for C-decidable aggregate atoms.

Proof
We will show that the complexity of the inverse problem of the SCP is in **NP**^{C}, i.e., determining whether $\langle I, J \rangle$ is not a solution of ℓ is in **NP**^{C}.

By definition, $\langle I, J \rangle$ is not a solution of ℓ if there exists an interpretation M such that $I \subseteq M$, $J \cap M = \emptyset$, and $M \not\models \ell$. To answer this question, we can guess an interpretation M and check whether ℓ is false in M. If it is, we conclude that $\langle I, J \rangle$ is not a solution of ℓ. Because ℓ is C-decidable and there are at most $2^{|\mathscr{H}(\ell) \setminus (I \cup J)|}$ interpretations that can be used in checking whether $\langle I, J \rangle$ is not a solution of ℓ, we conclude that the complexity of the inverse problem is in **NP**^{C}. □

https://doi.org/10.1017/S1471068406002936 Published online by Cambridge University Press

We will now address the problem of answer set checking and determining the existence of answer set.

Definition 17 (*(ACP) Answer Set Checking Problem*)
Given an interpretation M of P, **Determine** whether M is an answer set of P.

Definition 18 (*(AEP) Answer Set Existence Problem*)
Given a program P, **Determine** whether P has a fixpoint answer set.

The following theorem follows from Lemma 7.

Theorem 4
The ACP of C-decidable programs is in $\mathbf{co\text{-}NP}^C$.

Proof
The main tasks in checking whether M is an answer set of P are *(i)* computing ${}^{M}P$; and *(ii)* computing $lfp(K_M^P)$. Obviously, ${}^{M}P$ can be constructed in time linear in the size of P, since the reduction relies on the satisfiability test of a negation-as-failure literal ℓ w.r.t. M. Computing $lfp(K_M^P)$ requires at most na iterations, i.e., $lfp(K_M^P) = K_M^P \uparrow na$, where na is the number of atoms of P, each step is in $\mathbf{co\text{-}NP}^C$, due to the requirement of solution checking. □

This theorem allows us to conclude the following result.

Corollary 1
The AEP for C-decidable program is in $\mathbf{NP}^{\mathbf{co\text{-}NP}^C}$.

So far, we discussed the worst case analysis of answer set checking and determining the existing of an answer set based on a general assumption about the complexity of computing the aggregate functions and checking the truth value of aggregate atoms. Next we analyze the complexity of these problems w.r.t. the class of programs whose aggregate atoms are built using standard aggregate functions and operators.

4.4.1 Complexity of solution checking for standard aggregates

We will now focus on the class of programs defined in Section 2 with standard aggregate functions (Sum, Min, Max, Count, Avg) and relations ($=$, $\geqslant$, $>$, $\leqslant$, $<$, $\neq$). It is easy to see that all aggregate atoms involving these functions and relations are **P**-decidable. Therefore, by Lemma 7, the SCP for standard aggregates will be at most **co-NP**. We will now show that it is **co-NP**-complete.

Theorem 5
The SCP for standard aggregates is **co-NP**-complete.

Proof
Membership follows from Lemma 7. To prove hardness, we will translate a well-known **NP**-complete problem, namely the subset sum problem (Cormen *et al.* 2001), to the complement of the solution checking problem. An instance Q of the subset sum problem is given by a set of non-negative integers S and an integer t, and the

question is to determine whether there exists any non-empty subset A of S such that $\sum_{x \in A} x = t$.

Let $\mathscr{H}(\ell) = \{p(x) \mid x \in S\}$ for some unary predicate p. We define an instance of the solution checking problem, $s(Q)$, by setting $I = \emptyset$, $J = \emptyset$, and $\ell = \textsc{Sum}(\{X \mid p(X)\}) \neq t$. It is easy to see that $s(Q)$ is equivalent to Q as follows: if $\langle I, J\rangle$ is a solution of ℓ then Q does not have an answer; if $\langle I, J\rangle$ is not a solution to ℓ then Q has an answer. This proves the desired result. $\square$

The above theorem shows that, in general, the inclusion of standard aggregates implies that the answer set checking problem and the problem of determining the existing of an answer set are in **co-NP** and $\mathbf{NP}^{\mathbf{co\text{-}NP}}$ respectively. Fortunately, there is a large class of programs with standard aggregates for which the complexity of these two problems are in **P** and **NP** respectively, as shown next.

Lemma 8

Let ℓ be an aggregate of the form $\textsc{Sum}(\{X \mid p(X)\}) = v$, where v is a constant in $\mathbf{R}$. Let $I, J \subseteq \mathscr{H}(\ell)$ such that $I \cap J = \emptyset$. Then, determining whether $\langle I, J\rangle$ is a solution of ℓ can be done in time polynomial in the size of $\mathscr{H}(\ell)$.

Proof

Let us denote with π the function that projects an element p of $\mathscr{H}(\ell)$ to the value that p assigns to the collected variable. This value will be denoted by $\pi(p)$. We prove the lemma by providing a polynomial algorithm for determining whether $\langle I, J\rangle$ is a solution of ℓ.

```
function Check_Solution (v, ⟨I,J⟩, ℋ(ℓ))
    compute s = Σ_{p∈I} π(p)
    if s ≠ v then return false
    if ℋ(ℓ) \ (I ∪ J) = ∅ then return true;
    forall (p ∈ ℋ(ℓ) \ (I ∪ J))
        if π(p) ≠ 0 then return false
    endfor
    return true
```

It is easy to see that the above algorithm returns true (resp. false) if and only if $\langle I, J\rangle$ is (resp. is not) a solution of ℓ. Furthermore, the time complexity of the above algorithm is polynomial in the size of $\mathscr{H}(\ell)$. This proves the lemma. $\square$

The above lemma shows that the solution checking problem can be solved in polynomial time for a special type of standard aggregate atoms. Indeed, this can be proven for all standard aggregates but those of the form $\textsc{Sum} \neq v$ and $\textsc{Avg} \neq v$.

Lemma 9

Let ℓ be the aggregate $agg(s)$ **op** v where $agg \notin \{\textsc{Sum}, \textsc{Avg}\}$ or $agg \in \{\textsc{Sum}, \textsc{Avg}\}$ and **op** is not '$\neq$'. Let $I, J \subseteq \mathscr{H}(\ell)$, $I \cap J = \emptyset$, and $v \in \mathbf{R}$. Then, checking if $\langle I, J\rangle$ is a solution of ℓ can be done in time polynomial in the size of $\mathscr{H}(\ell)$.

https://doi.org/10.1017/S1471068406002936 Published online by Cambridge University Press

Proof

The proof can be done similarly to the proof of Lemma 8: for each type of atom, we develop an algorithm, which returns true (resp. false) if $\langle I, J\rangle$ is (resp. is not) a solution of ℓ. For brevity, we only discuss the steps which need to be done. It should be noted that each of these steps can be done in polynomial time in the size of $\mathscr{H}(\ell)$, which implies the conclusion of the lemma.

- SUM: Let $s = \sum_{p \in I} \pi(p)$. All cases can be handled in time $O(|\mathscr{H}(\ell)|)$. Let us consider the various cases for **op**.
  - The case **op** is '=' has been discussed in Lemma 8.
  - For **op** $\in \{\geqslant, >\}$, let $H_1 = \{p \mid p \in \mathscr{H}(\ell) \setminus (I \cup J),\ \pi(p) < 0\}$. We have that $\langle I, J\rangle$ is a solution of ℓ if and only if s **op** v and $\sum_{p \in H_1} \pi(p) + s$ **op** v.
  - For **op** $\in \{\leqslant, <\}$, let $H_1 = \{p \mid p \in \mathscr{H}(\ell) \setminus (I \cup J),\ \pi(p) > 0\}$. We have that $\langle I, J\rangle$ is a solution of ℓ if and only if s **op** v and $\sum_{p \in H_1} \pi(p) + s$ **op** v.
- COUNT: Let $c = |I|$ and $H_1 = \mathscr{H}(\ell) \setminus (I \cup J)$. All cases can be handled in time $O(|\mathscr{H}(\ell)|)$.
  - If **op** $\in \{>, \geqslant\}$, then $\langle I, J\rangle$ is a solution of ℓ if and only if c **op** v.
  - If **op** $\in \{=, <, \leqslant\}$, then $\langle I, J\rangle$ is a solution of ℓ if and only if c **op** v and $c + |H_1|$ **op** v.
  - If **op** is $\neq$, then $\langle I, J\rangle$ is a solution of ℓ if and only if either *(i)* $|I| > v$; or *(ii)* $|I| < v$ and $|H_1| < v - |I|$.
- MIN: Let $c = \min\{\pi(p) \mid p \in I\}$ and $c_1 = \min\{\pi(p) \mid p \in \mathscr{H}(\ell) \setminus (I \cup J)\}$. All cases can be handled in time $O(|\mathscr{H}(\ell)|)$.
  - If **op** is $=$ then we have that $\langle I, J\rangle$ is a solution of ℓ if and only if $c = v$ and $c_1 \geqslant v$.
  - If **op** $\in \{\leqslant, <\}$ then $\langle I, J\rangle$ is a solution of ℓ if and only if c **op** v.
  - If **op** $\in \{\geqslant, >\}$ then $\langle I, J\rangle$ is a solution of ℓ if and only if c **op** v and c_1 **op** v.
  - If **op** is $\neq$ then $\langle I, J\rangle$ is a solution of ℓ if and only if either *(i)* $c < v$; or *(ii)* $c > v$ and for every $p \in H_1$, $\pi(p) \neq v$.
- MAX: Let $c = \max\{\pi(p) \mid p \in I\}$ and $c_1 = \max\{\pi(p) \mid p \in \mathscr{H}(\ell) \setminus (I \cup J)\}$. All cases can be handled in time $O(|\mathscr{H}(\ell)|)$.
  - If **op** is $=$ then $\langle I, J\rangle$ is a solution of ℓ if and only if $c = v$ and $c_1 \leqslant v$.
  - If **op** $\in \{\geqslant, >\}$ then $\langle I, J\rangle$ is a solution of ℓ if and only if c **op** v.
  - If **op** $\in \{\leqslant, <\}$ then $\langle I, J\rangle$ is a solution of ℓ if and only if c **op** v and c_1 **op** v.
  - If **op** is $\neq$ then $\langle I, J\rangle$ is a solution of ℓ if and only if either *(i)* $c > v$; or *(ii)* $c < v$ and for every $p \in H_1$, $\pi(p) \neq v$.
- AVG: Let $a = \frac{\sum_{p \in I}\{\pi(p)\}}{|I|}$ and $H_1 = \mathscr{H}(\ell) \setminus (I \cup J)$.
  - If **op** is $=$ then $\langle I, J\rangle$ is a solution of ℓ if and only if $a = v$ and for every $p \in H_1$, $\pi(p) = v$. This can be done in time $O(|\mathscr{H}(\ell)|)$.
  - If **op** $\in \{\geqslant, >\}$ then let $e_1, \ldots, e_r$ be an enumeration of H_1 such that $\pi(e_i) \leqslant \pi(e_{i+1})$ for $1 \leqslant i \leqslant r - 1$. $\langle I, J\rangle$ is a solution of ℓ if and only if a **op** v and for

https://doi.org/10.1017/S1471068406002936 Published online by Cambridge University Press

each $0 \leqslant h \leqslant r$,

$$\sum_{p \in I} \pi(p) + \sum_{i=1}^{h} \pi(e_i) \ \mathbf{op} \ v \cdot |I| + v \cdot h.$$

This can be accomplished in time $O(|\mathscr{H}(\ell)|^2)$.

- If $\mathbf{op} \in \{\leqslant, <\}$ then let $e_1, \ldots, e_r$ be an enumeration of H_1 such that $\pi(e_i) \geqslant \pi(e_{i+1})$ for $1 \leqslant i \leqslant r-1$. $\langle I, J \rangle$ is a solution of ℓ if and only if a $\mathbf{op}$ v and for each $0 \leqslant h \leqslant r$,

$$\sum_{p \in I} \pi(p) + \sum_{i=1}^{h} \pi(e_i) \ \mathbf{op} \ v \cdot |I| + v \cdot h.$$

This can be accomplished in time $O(|\mathscr{H}(\ell)|^2)$.

□

The above lemma shows that there is a large class of programs with aggregates for which the problem of checking an answer set and the problem of determining the existence of an answer set belongs to the class **P** and **NP** respectively.

Observe that similar results can be extrapolated from the discussion in Pelov's doctoral dissertation (Pelov 2004).

5 Conclusions and future work

In this technical note, we defined K_M^P, a fixpoint operator for verifying answer sets of programs with aggregates. We showed that the semantics for programs with aggregates described by this operator provides a new characterization of the semantics of Son *et al.* (2005) for logic programs with aggregates. This operator converges to the same semantics as in Pelov (2004) when ultimate approximating aggregates are used. We also related this semantics to recently proposed semantics for aggregate programs. We discussed the complexity of the answer set checking problem and the problem of determining the existence of an answer set. We showed that, for the class of programs with standard aggregates without the relation $\neq$ for Sum and Avg, the complexity of these two problems remains unchanged comparing to that of normal logic programs. In the future, we would like to use this idea in an efficient implementation of answer set solvers with aggregates.

Appendix: Correspondence between K_M^P and Φ_P^{aggr}

We assume that the readers are familiar with the notations and definitions introduced in Pelov *et al.* (2004).

The three-valued immediate consequence operator Φ_P^{aggr} of a program P in (Pelov *et al.* 2004), maps 3-valued interpretations to 3-valued interpretations. But 3-valued interpretations can be split up in pairs (I, J) of two valued interpretations such that $I \subseteq J$. Hence, an operator Φ_P^{aggr} can be viewed as an operator from pairs (I, J) to pairs $\Phi_P^{aggr}(I, J) = (I', J')$ of 2-valued interpretations. It follows that Φ_P^{aggr}

determines two component operators $\Phi_P^{aggr,1}(I,J) = I'$ and $\Phi_P^{aggr,2}(I,J) = J'$. The correspondence between K_M^P and Φ_P^{aggr} is shown in the following claim.

Claim. For every $I \subseteq M$, $K_M^P(I) = \Phi_P^{aggr,1}(I,M)$.

Proof

First, let us identify the aggregate atoms $agg(s)$ **op** v in this paper with aggregate atoms $R(s,v)$ of Pelov *et al.* (2004). For example, $\textsc{Max}(s) = v$ corresponds to $\textsc{Max}(s,v)$; $\textsc{Max}(s) \leqslant v$ corresponds to $\textsc{Max}_{\leqslant}(s,v)$. Now we compare the definition of K_M^P and $\Phi_P^{aggr,1}$ in the case that $I \subseteq M$. For simplicity let us assume that atom a is defined by only one ground rule, say r.

$a \in K_M^P(I)$ iff $pos(r)$ is true in I, $neg(r)$ is false in M, and for each $\ell \in aggr(r)$, l has a solution $(I \cap M \cap \mathscr{H}(\ell), \mathscr{H}(\ell) \setminus M)$.

$a \in \Phi_P^{aggr,1}(I,M)$ iff $pos(r)$ is true in I, $neg(r)$ is false in M, and for each $\ell \in aggr(r)$, l evaluates to true, i.e., if $U_R^1(s^{(I,M)})) = t$. Here, U_R^1 is the first component of the three-valued aggregate, and $s^{(I,M)}$ is the evaluation of the set expression under the 3-valued interpretation (I,M).

All that remains to be done is to show that $(I \cap M \cap \mathscr{H}(\ell), \mathscr{H}(\ell) \setminus M)$ is a solution for l iff $U_R^1(s^{(I,M)}) = t$. Recall that we are considering the case where $I \subseteq M$, therefore the first expression simplifies to $(I \cap \mathscr{H}(\ell), \mathscr{H}(\ell) \setminus M)$.

Let us focus on set aggregates but the argument for multisets is the same. Let us consider an aggregate atom

$$\ell = agg(s) \ \textbf{op} \ v$$

where

$$s = \{X \mid p(d_1, \ldots, d_{i-1}, X, d_{i+1}, \ldots, d_n)\}$$

and X is the only variable and $d_1, \ldots, d_n$ are members of the Herbrand universe. For any $I \subseteq M$,

$$\begin{array}{c} (I \cap \mathscr{H}(\ell), \mathscr{H}(\ell) \setminus M) \text{ is a solution for } \ell \\ \text{iff for each } J \text{ such that } I \cap \mathscr{H}(\ell) \subseteq J \text{ and } J \cap (\mathscr{H}(\ell) \setminus M) = \emptyset,\ J \models \ell \\ \text{iff for each } J \text{ such that } I \subseteq J \subseteq M,\ J \models \ell. \end{array}$$

The latter equivalence is perhaps not entirely trivial but it follows easily from the fact that $J \models \ell \Leftrightarrow J' \models \ell$ whenever $J \cap \mathscr{H}(\ell) = J' \cap \mathscr{H}(\ell)$.

In Pelov *et al.* (2004), the value $s^{(I,M)}$ is a three-valued (multi-)set, which can be written as a pair of two valued sets (S_1, S_2) where

$$S_1 = \{d \mid I \models p(d_1, \ldots, d_{i-1}, d, d_{i+1}, \ldots, d_n)\}$$

and

$$S_2 = \{d \mid M \models p(d_1, \ldots, d_{i-1}, d, d_{i+1}, \ldots, d_n)\}.$$

By definition of U_R^1, $U_R^1(s^{(I,M)})) = t$ iff for each set S such that $S_1 \subseteq S \subseteq S_2$, $R(S,v)$ is true. It is straightforward to see that the conditions in this paragraph and the previous one are equivalent. □

Acknowledgments

The authors wish to thank the anonymous reviewers for their insightful comments and for pointing out relationships with existing literature, and Dr. Hing Leung for his suggestions.

The research has been partially supported by NSF grants HRD-0420407, CNS-0454066, and CNS-0220590.

References

Cormen, T. H., Leiserson, C. E., Rivest, R. L. and Stein, C. 2001. *Introduction to Algorithms, 2nd Edition*. MIT Press, Cambridge, MA.

Dovier, A., Pontelli, E., and Rossi, G. 2001. Constructive negation and constraint logic programming with sets. *New Generation Comput. 19,* 3, 209–256.

Dovier, A., Pontelli, E., and Rossi, G. 2003. Intensional Sets in CLP. In *International Conference on Logic Programming*, Springer, 284–299.

Elkabani, I., Pontelli, E., and Son, T. C. 2004. Smodels with CLP and its Applications: a Simple and Effective Approach to Aggregates in ASP. In *International Conference on Logic Programming*, Springer, 73–89.

Faber, W., Leone, N., and Pfeifer, G. 2004. Recursive Aggregates in Disjunctive Logic Programs: Semantics and Complexity. In *JELIA*, Springer, 200–212.

Gelfond, M. 2002. Representing Knowledge in A-Prolog. In *Computational Logic: Logic Programming and Beyond*, Springer Verlag, 413–451.

Gelfond, M. and Lifschitz, V. 1988. The Stable Model Semantics for Logic Programming. In *International Conf. and Symp. on Logic Programming*, MIT Press, 1070–1080.

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

Lloyd, J. 1987. *Foundations of Logic Programming*. Springer Verlag.

Mumick, I. S., Pirahesh, H., and Ramakrishnan, R. 1990. The Magic of Duplicates and Aggregates. In *Int. Conf. on Very Large Data Bases*, Morgan Kaufmann, 264–277.

Pelov, N. 2004. Semantic of Logic Programs with Aggregates. Ph.D. thesis, Katholieke Universiteit Leuven.

Pelov, N., Denecker, M., and Bruynooghe, M. 2003. Translation of Aggregate Programs to Normal Logic Programs. In *ASP: Advances in Theory and Implementation*, CEUR Workshop Proceedings. 29–42.

Pelov, N., Denecker, M., and Bruynooghe, M. 2004. Partial Stable Models for Logic Programs with Aggregates. In *LPNMR*, Springer, 207–219.

Ross, K. A. and Sagiv, Y. 1997. Monotonic Aggregation in Deductive Database. *J. Comput. Syst. Sci. 54,* 1, 79–97.

Son, T. C., Pontelli, E., and Elkabani, I. 2005. A Translational Semantics for Aggregates in Logic Programming . Tech. Rep. CS-2005-006, New Mexico State University. `www.cs.nmsu.edu/CSWS/php/techReports.php?rpt_year=2005`.

Zaniolo, C., Arni, N., and Ong, K. 1993. Negation and Aggregates in Recursive Rules: the LDL++ Approach. In *DOOD*. 204–221.