Aggregates in Recursion: Issues and Solutions

Wolfgang Faber

Alpen-Adria-Universität Klagenfurt

RuleML Webinar, 2018-05-25
Outline

1. Motivation

2. Basics: Answer Set Programming and Aggregates

3. Semantics
   - Stratified Aggregates
   - Unstratified Aggregates

4. Properties
   - Coincidence Results
   - Complexity Results
Motivation: Aggregates

- **Aggregates** facilitate problem representation
- **Standard** in database query languages
  For example:
  ```sql
  SELECT company.name FROM company, employee
  WHERE company.id = employee.cid AND
  COUNT(employee.id) < 10
  ```
In this talk: Combination of Logic Programming and Aggregates

For example:

SELECT company.name FROM company, employee
WHERE company.id = employee.cid AND
COUNT(employee.id) < 10

using Logic Programming (here ASP/Datalog):

\[
\text{result}(N) \leftarrow \text{company}(N, Y), \#\text{count}\{X : \text{employee}(X, Y)\} < 10.
\]

Does this look innocent?
A main feature of Logic Programming is recursion.

Example:

\[ tc(A, B) := p(A, B). \]
\[ tc(A, B) := p(A, X), tc(X, B). \]

defines the transitive closure of \( p \).

Supported also in SQL:

- Recent (since SQL-99)
- Not particularly well-known
- Common Table Expressions (CTE)
Recursion and Aggregates

- Combination of recursion and aggregates?
  - Explicitly forbidden in SQL!
- Meaning of
  \[ p(a) := \#\text{count}\{X : p(X)\} > 0. \]
- Meaning of
  \[ p(a) := \#\text{count}\{X : p(X)\} < 1. \]
- Meaning of
  \[ p(1) := \#\text{avg}\{X : p(X)\}! = 1. \]
  \[ p(-1) := \#\text{avg}\{X : p(X)\}! = 1. \]
Motivation
Basics: Answer Set Programming and Aggregates
Semantics
Properties

ASP Semantics

- Herbrand interpretations
- Reduct for interpretation $I$:
  1. Delete rules whose negative body is true in $I$.
  2. Delete negative body from all other rules.
- Interpretations $I$ which are minimal models of the reduct for $I$ are answer sets.

- [Gelfond, Lifschitz 1988] (nondisjunctive)
- [Przymusinski 1991] (disjunctive)
- [Gelfond, Lifschitz 1991] (disjunctive, 2 kinds of negation)
ASP Semantics

- Herbrand interpretations
- Reduct for interpretation $I$:
  1. Delete rules whose negative body is true in $I$.
  2. Delete negative body from all other rules.
- Interpretations $I$ which are minimal models of the reduct for $I$ are answer sets.

- [Gelfond, Lifschitz 1988] (nondisjunctive)
- [Przymusinski 1991] (disjunctive)
- [Gelfond, Lifschitz 1991] (disjunctive, 2 kinds of negation)
Aggregates

- **Aggregate Functions**: Functions over ground term (multi)sets
  - (Multi)sets specified as \{A, B : Conj\} or \{(c, d : Conj), \ldots\}
  - Evaluate \textit{Conj} w.r.t. an interpretation
  - **Aggregate Atoms**: Aggregate Function plus comparison

**Important**: Aggregate atoms depend on truth values of a set of standard atoms!
Aggregates

- Aggregate Functions: Functions over ground term (multi)sets
- (Multi)sets specified as \{A, B : Conj\} or \{⟨c, d : Conj⟩, . . .\}
- Evaluate \textit{Conj} w.r.t. an interpretation

Important: Aggregate atoms depend on truth values of a set of standard atoms!
Aggregates

- Aggregate Functions: Functions over ground term (multi)sets
- (Multi)sets specified as \{A, B : Conj\} or \{\langle c, d : Conj\rangle, \ldots\}
- Evaluate Conj w.r.t. an interpretation
- **Aggregate Atoms**: Aggregate Function plus comparison

Important: Aggregate atoms depend on truth values of a set of standard atoms!
Aggregates

- Aggregate Functions: Functions over ground term (multi)sets
- (Multi)sets specified as \{A, B : Conj\} or \{⟨c, d : Conj⟩, . . .\}
- Evaluate Conj w.r.t. an interpretation
- **Aggregate Atoms**: Aggregate Function plus comparison

**Important**: Aggregate atoms depend on truth values of a set of standard atoms!
Outline

1. Motivation
2. Basics: Answer Set Programming and Aggregates
3. Semantics
   - Stratified Aggregates
   - Unstratified Aggregates
4. Properties
   - Coincidence Results
   - Complexity Results
Aggregate Stratification

Definition

A program $\mathcal{P}$ is *stratified on an aggregate atom* $A$ if there exists a level mapping $\| \|$ from its predicates to ordinals, such that for each rule and for each of its head atoms $a$ the following holds:

1. For each predicate $b$ of standard body literals: $\| b \| \leq \| a \|$, 
2. for each predicate $b$ inside an aggregate body atom: $\| b \| < \| a \|$, and 
3. for each predicate $b$ in the head: $\| b \| = \| a \|$. 

Wolfgang Faber
Aggregates in Recursion: Issues and Solutions
Aggregate Stratification

- **Note**: Stratification is relative to a program
- Unstratified aggregate atoms occur **recursively**

**Example**

\[ a := \#\text{count}\{\langle t : b \rangle\} > 0. \]

is stratified on the aggregate atom.

\[ a := \#\text{count}\{\langle t : b \rangle\} > 0. \]
\[ b := a. \]

is not stratified on the aggregate atom.
Answer Sets for Aggregate-stratified Programs

- Basic Idea: Treat aggregate atoms like negative literals
- Reduct:
  1. Delete rules containing unsatisfied aggregates and negative literals
  2. Delete aggregates and negative literals from all other rules

- [Kemp, Stuckey 1991], [Gelfond 2002], [Dell’Armi, F., Ielpa, Leone, Pfeifer 2003]
- Many programs are aggregate stratified
- Use of aggregates often yields computational advantages
- But: Not all programs are aggregate stratified
Answer Sets for Aggregate-stratified Programs

- Basic Idea: Treat aggregate atoms like negative literals
- Reduct:
  1. Delete rules containing unsatisfied aggregates and negative literals
  2. Delete aggregates and negative literals from all other rules

- [Kemp, Stuckey 1991], [Gelfond 2002], [Dell’Armi, F., Ielpa, Leone, Pfeifer 2003]
- Many programs are aggregate stratified
- Use of aggregates often yields computational advantages
- But: Not all programs are aggregate stratified
Answer Sets for Aggregate-stratified Programs

- Basic Idea: Treat aggregate atoms like negative literals
- Reduct:
  1. Delete rules containing unsatisfied aggregates and negative literals
  2. Delete aggregates and negative literals from all other rules

- [Kemp, Stuckey 1991], [Gelfond 2002], [Dell’Armi, F., Ielpa, Leone, Pfeifer 2003]
- Many programs are aggregate stratified
- Use of aggregates often yields computational advantages
- But: Not all programs are aggregate stratified
Outline

1. Motivation
2. Basics: Answer Set Programming and Aggregates
3. Semantics
   - Stratified Aggregates
   - Unstratified Aggregates
4. Properties
   - Coincidence Results
   - Complexity Results
Unstratification

- What happens when we consider unstratified aggregates?
- Can we just keep the simple semantic definition?
Company Control

**Input:** Set of companies and shares companies hold of other companies.

**Output:** Companies controlled (direct or indirect shares > 50%) by other companies

Encoding from the literature:

\[
\text{controlsStk}(C_1, C_1, C_2, P) \gets \text{ownsStk}(C_1, C_2, P).
\]
\[
\text{controlsStk}(C_1, C_2, C_3, P) \gets \text{controls}(C_1, C_2), \text{ownsStk}(C_2, C_3, P).
\]
\[
\text{controls}(C_1, C_3) \gets \text{company}(C_1), \text{company}(C_3), \\
\#\sum\{P, C_2 : \text{controlsStk}(C_1, C_2, C_3, P)\} > 50.
\]
Company Control, Instance 1

Example (Company Control, Instance 1)

\[
\{\text{controlsStk}(a,a,b,60), \\
\text{controlsStk}(a,a,c,30), \\
\text{controlsStk}(a,a,d,15), \\
\text{controlsStk}(b,b,d,40), \\
\text{controlsStk}(c,c,d,35), \\
\text{controlsStk}(a,b,d,40), \\
\text{controls}(a,b), \\
\text{controls}(a,d)\}
\]
Company Control, Instance 1

Example (Company Control, Instance 1)

{controlsStk(a,a,b,60),
 controlsStk(a,a,c,30),
 controlsStk(a,a,d,15),
 controlsStk(b,b,d,40),
 controlsStk(c,c,d,35),
 controlsStk(a,b,d,40),
 controls(a,b),
 controls(a,d)}
Company Control, Instance 2

Example (Company Control, Instance 2)

\{\text{controlsStk}(a,a,b,40), \text{controlsStk}(a,a,c,40), \text{controlsStk}(b,b,c,20), \text{controlsStk}(c,c,b,20)\}

But also:
\{\text{controlsStk}(a,b,c,20), \text{controlsStk}(a,c,b,20), \text{controls}(a,b), \text{controls}(a,c)\}
Company Control, Instance 2

Example (Company Control, Instance 2)

{controlsStk(a,a,b,40), controlsStk(a,a,c,40), controlsStk(b,b,c,20), controlsStk(c,c,b,20)}

But also:

{controlsStk(a,b,c,20), controlsStk(a,c,b,20), controls(a,b), controls(a,c)}
Example (Company Control, Instance 2)

{controlsStk(a,a,b,40),
controlsStk(a,a,c,40),
controlsStk(b,b,c,20),
controlsStk(c,c,b,20)}

But also:
{controlsStk(a,b,c,20),
controlsStk(a,c,b,20),
controls(a,b),
controls(a,c)}
Essence

Example

\[ a \leftarrow \#\text{count}\{\langle t : a \rangle\} < 1. \]
No answer sets.

\[ a \leftarrow \#\text{count}\{\langle t : a \rangle\} > 0. \]
Answer sets: \( \emptyset, \{a\} \)?

\#\text{count}\{\langle t : a \rangle\} < 1 behaves like not \( a \)
\#\text{count}\{\langle t : a \rangle\} > 0 behaves like \( a \)
⇒ aggregates should not be treated like negative literals, but
also not like positive literals
Example

\[ a \leftarrow \# \text{count}\{\langle t : a \rangle\} < 1. \]
No answer sets.

\[ a \leftarrow \# \text{count}\{\langle t : a \rangle\} > 0. \]
Answer sets: \( \emptyset, \{a\} \)?

\# \text{count}\{\langle t : a \rangle\} < 1 \text{ behaves like } not a
\# \text{count}\{\langle t : a \rangle\} > 0 \text{ behaves like } a
⇒ \text{aggregates should not be treated like negative literals, but also not like positive literals}
Monotonicity and Antimonotonicity

- **Monotone** Literals:
  truth for interpretation $I$ implies truth for all $J \supseteq I$

- **Antimonotone** Literals:
  truth for interpretation $J$ implies truth for all $I \subseteq J$

- **Nonmonotone** Literals:
  neither monotone nor antimonotone
Monotonicity: Examples

- \#count{...} \geq 1 is monotone
- \#count{...} < 1 is antimonotone
- \#avg{...} < 3 is nonmonotone
- Positive standard literals are monotone
- Negative standard literals are antimonotone
FLP Semantics: Novel Reduct Definition

Definition of reduct according to [F., Leone, Pfeifer 2004, F., Leone, Pfeifer 2011]:

- Delete rules with a false body literal.
- That's it!

Answer Set: Subset-minimal model of the reduct

Theorem

For aggregate-free programs, answer sets under this definition coincide with the ones defined in [Gelfond, Lifschitz 1991].

(Mostly) equivalent semantics defined in [Ferraris 2005], [Ferraris 2011].
FLP Semantics: Novel Reduct Definition

Definition of reduct according to [F.,Leone,Pfeifer 2004, F., Leone, Pfeifer 2011]:

- Delete rules with a false body literal.
- That’s it!

Answer Set: Subset-minimal model of the reduct

Theorem

For aggregate-free programs, answer sets under this definition coincide with the ones defined in [Gelfond, Lifschitz 1991].

(Mostly) equivalent semantics defined in [Ferraris 2005], [Ferraris 2011].
FLP Semantics: Novel Reduct Definition

Definition of reduct according to [F., Leone, Pfeifer 2004, F.,
Leone, Pfeifer 2011]:

- Delete rules with a false body literal.
- That’s it!

Answer Set: Subset-minimal model of the reduct

Theorem

For aggregate-free programs, answer sets under this definition coincide with the ones defined in [Gelfond, Lifschitz 1991].

(Mostly) equivalent semantics defined in [Ferraris 2005],
[Ferraris 2011].
Example (Company Control, Instance 1)

\{
\text{controlsStk}(a,a,b,60),
\text{controlsStk}(a,a,c,30),
\text{controlsStk}(a,a,d,15),
\text{controlsStk}(b,b,d,40),
\text{controlsStk}(c,c,d,35),
\text{controlsStk}(a,b,d,40),
\text{controls}(a,b),
\text{controls}(a,d)
\}
Example (Company Control, Instance 1)

\{\text{controlsStk}(a,a,b,60), \text{controlsStk}(a,a,c,30), \text{controlsStk}(a,a,d,15), \text{controlsStk}(b,b,d,40), \text{controlsStk}(c,c,d,35), \text{controlsStk}(a,b,d,40), \text{controls}(a,b), \text{controls}(a,d)\}
Example (Company Control, Instance 2)

\{controlsStk(a,a,b,40),
controlsStk(a,a,c,40),
controlsStk(b,b,c,20),
controlsStk(c,c,b,20)\}

Only this answer set.
Example (Company Control, Instance 2)

{controlsStk(a,a,b,40),
controlsStk(a,a,c,40),
controlsStk(b,b,c,20),
controlsStk(c,c,b,20)}

Only this answer set.
Small examples

Example

\( a : \# \text{count}\{\langle t : a \rangle\} < 1.\)

No answer sets.

\( a : \# \text{count}\{\langle t : a \rangle\} > 0.\)

Answer sets: only \( \emptyset \)
PSP Semantics

- Alternative semantics for unstratified aggregates:
  - [Pelov 2004], [Son, Pontelli 2007], [Shen, Wang 2012]
  - Definitions use different operator-based techniques

- Evaluate aggregates for a pair of interpretations

\[(I, J) \models A \text{ iff } K \models A \text{ for all } I \subseteq K \subseteq J.\]

- PSP answer sets are those \(I\) that are fixpoints of \(K_I^{\Pi} \uparrow \emptyset\)

- Operator \(K_I^{\Pi}(X)\) collects heads of rules for which \((X, I) \models A\) for all body atoms
Small examples: PSP

**Example**

\[
\text{a} \leftarrow \# \text{count} \{ \langle t : a \rangle \} < 1.
\]

No answer sets.

\[
K_{\emptyset}^{\emptyset}(\emptyset) = \{a\}
\]

\[
K_{\{a\}}^{\emptyset}(\emptyset) = \emptyset
\]

\[
\text{a} \leftarrow \# \text{count} \{ \langle t : a \rangle \} > 0.
\]

Answer sets: only \(\emptyset\)

\[
K_{\emptyset}^{\emptyset}(\emptyset) = \emptyset
\]

\[
K_{\{a\}}^{\emptyset}(\emptyset) = \emptyset
\]
Outline

1. Motivation
2. Basics: Answer Set Programming and Aggregates
3. Semantics
   - Stratified Aggregates
   - Unstratified Aggregates
4. Properties
   - Coincidence Results
   - Complexity Results
FLP and PSP

- Small examples could suggest that they coincide.
- But this is not true in general.

Example

\[
\begin{align*}
  p(1) :&= \#avg\{X : p(X)\} \geq 0. \\
  p(1) :&= p(-1). \\
  p(-1) :&= p(1).
\end{align*}
\]

FLP answer sets: \{p(1), p(-1)\}
PSP answer sets: none

In general: each PSP answer set is also FLP, but not necessarily vice versa.
Classes on which the semantics coincide?
Aggregate-stratified Programs

Easy observation:

**Theorem**

*FLP and PSP semantics coincide on aggregate-stratified programs.*
Monotone, antimonotone, convex

\[
\{a, b, c, d\} \\
\{a, b, c\} \quad \{a, b, d\} \quad \{a, c, d\} \quad \{b, c, d\} \\
\{a, b\} \quad \{a, c\} \quad \{a, d\} \quad \{b, c\} \quad \{b, d\} \quad \{c, d\} \\
\{a\} \quad \{b\} \quad \{c\} \quad \{d\} \\
\emptyset
\]

\[1 \leq \#\text{count}\{a, b, c, d\}\]

\(S\) is monotone if
\(I \models S \land J \models S \Rightarrow K \models S\) \quad \forall K \in \uparrow I \cap \downarrow J
Motivation
Basics: Answer Set Programming and Aggregates
Semantics
Properties
Coincidence Results
Complexity Results

Monotone, antimonotone, convex

$\{a, b, c, d\} \rightarrow \{a, b, c\} \rightarrow \{a, b\} \rightarrow \{a\}$

$\{a, b, c, d\} \rightarrow \{a, b, d\} \rightarrow \{a, c, d\} \rightarrow \{b, c, d\}$

$1 \leq \#\text{count}\{a, b, c, d\}$

$S$ is monotone if
$I \models S \land J \models S \Rightarrow K \models S \quad \forall K \in \uparrow I \cap \downarrow J$
Monotone, antimonotone, convex

\# \text{count}\{a, b, c, d\} \leq 3

\text{S is antimonotone if}

\begin{align*}
I & \models S \land J \models S \\
K & \models S \quad \forall K \in \uparrow \cap \downarrow J
\end{align*}
Monotone, antimonotone, convex

\[ \# \text{count}\{a, b, c, d\} \leq 3 \]

\( S \) is antimonotone if
\( I \models S \land J \models S \land K \models S \quad \forall K \in \uparrow / \cap \downarrow J \)

Motivation
Basics: Answer Set Programming and Aggregates
Semantics
Properties
Coincidence Results
Complexity Results
Monotone, antimonotone, convex

\[ 1 \leq \# \text{count}\{a, b, c, d\} \leq 3 \]

S is convex if
\[ I \models S \land J \models S \implies K \models S \quad \forall K \in \uparrow I \cap \downarrow J \]
Monotone, antimotone, convex

\[ 1 \leq \# \text{count}\{a, b, c, d\} \leq 3 \]

\[ S \text{ is convex if } \]
\[ I \models S \land J \models S \quad \Rightarrow \quad K \models S \quad \forall K \in \uparrow I \cap \downarrow J \]
Convex Programs

Implicit in [Liu, Truszczyński 2006]:

**Theorem**

*FLP and PSP semantics coincide on programs containing only convex aggregates.*

**Corollary**

*FLP and PSP semantics coincide on programs containing only monotone and antimonotone aggregates.*
Outline

1. Motivation
2. Basics: Answer Set Programming and Aggregates
3. Semantics
   - Stratified Aggregates
   - Unstratified Aggregates
4. Properties
   - Coincidence Results
   - Complexity Results
Cautious reasoning over variable-free programs and polynomial-time computable aggregate functions.

**Input:** A ground program $\mathcal{P}$ and a ground standard atom $A$.
**Output:** Is $A$ true in all FLP answer sets of $\mathcal{P}$?

Similar results for related problems (answer set existence, brave reasoning).
Cautious reasoning over variable-free programs and polynomial-time computable aggregate functions.

**Input:** A ground program $\mathcal{P}$ and a ground standard atom $A$.

**Output:** Is $A$ true in all FLP answer sets of $\mathcal{P}$?

Similar results for related problems (answer set existence, brave reasoning).
Complexity of Cautious Reasoning

<table>
<thead>
<tr>
<th></th>
<th>{}</th>
<th>{\text{not}}</th>
<th>{\lor}</th>
<th>{\text{not,} \lor}</th>
</tr>
</thead>
<tbody>
<tr>
<td>{\text{M}}</td>
<td>\text{P}</td>
<td>\text{co-NP}</td>
<td>\text{co-NP}</td>
<td>\Pi_2^P</td>
</tr>
<tr>
<td>{\text{S}}</td>
<td>\text{P}</td>
<td>\text{co-NP}</td>
<td>\Pi_2^P</td>
<td>\Pi_2^P</td>
</tr>
<tr>
<td>{\text{C}}</td>
<td>\text{co-NP}</td>
<td>\text{co-NP}</td>
<td>\Pi_2^P</td>
<td>\Pi_2^P</td>
</tr>
<tr>
<td>{\text{N}}</td>
<td>\Pi_2^P</td>
<td>\Pi_2^P</td>
<td>\Pi_2^P</td>
<td>\Pi_2^P</td>
</tr>
</tbody>
</table>

\{\text{M}\}: monotone aggregates \{\text{S}\}: stratified aggregates \{\text{C}\}: convex aggregates \{\text{N}\}: non-convex aggregates
Results for PSP: analogous to FLP

Slight difference for non-convex aggregates [Alviano, F. 2013]:
- One non-convex aggregate is sufficient to express any problem in $\Pi_2^P$ with FLP
- Arbitrarily many are needed to do the same with PSP
### Only Aggregates?

All of the results presented here also apply to other extensions of ASP that have constructs that evaluate truth on sets of basic atoms, for example:

- Abstract Constraint Atoms \((U, V)\) [Marek, Remmel 2004]
- HEX programs [Eiter et al. 2005]
- Nested Expressions [Lifschitz et al. 1999]
- Generalized Quantifiers [Lindström 1966]
- Cardinality and Weight Constraints [Simons 2000]
Summary

- Aggregates in ASP
- FLP and PSP Semantics
- Properties: Coincidence and Complexity
More

- Beyond FLP and PSP? [Alviano, F. 2015]

**Example**

\[ a := \#\text{count}\{a, b\}! = 1. \]
\[ b := \#\text{count}\{a, b\}! = 1. \]

No FLP, no PSP answer sets! Unintuitive?

- Reduce programs with non-convex aggregates to programs with monotone aggregates in a compact way. [Alviano, F., Gebser 2015]