Sm\textit{models}^A — A system for Computing Answer Sets of Logic Programs with Aggregates

Islam Elkabani, Enrico Pontelli, Son Cao Tran

Outline

- Motivation
- Introduction
- New Semantics
- Examples
- Sm\textit{models}^A System
- Evaluation
- Conclusion and Future work
Motivation

- Many proposals introduced to handle aggregates in Logic Programming in the late 80’s and early 90’s.
- Most of these proposals focused on providing a sensible semantics for programs with recursive aggregates.
- Recently a number of proposals based on the spirit of the answer set semantics are provided.
- Most of the implementations build on these proposals did not handle programs with recursive aggregates (e.g., DLV^2).

Example (Company Control)

\[
\text{control\_shares}(X,Y,N) :\text{-- owns}(X,Y,N).
\]

\[
\text{control\_shares}(X,Y,N) :\text{-- company}(X), \text{control}(X,Z), \text{owns}(Z,Y,N).
\]

\[
\text{control}(X,Y) :\text{-- company}(X), \text{company}(Y), \text{sum}({\{A, \text{control\_shares}(X,Y,A)\}})>50.
\]

```
A --- 0.60 --- B
|      |     |
|      | 0.20 |
|      |     |
| 0.40 |     |
|      | 0.16 |
C
```
Introduction

- ASP-CLP(Agg) was capable of computing answer sets of arbitrary programs with aggregates without any syntactical restrictions imposed on the inputs, i.e., aggregates stratification.

- However, the ASP-CLP(Agg) system is based on a semantics that does not guarantee minimality of answer sets.
  - Example:

\[
\begin{align*}
& p(1), \quad p(2), \quad p(3), \\
& q \leftarrow \text{sum}(\{X, p(X)\}) > 10, \\
& (5) \leftarrow q.
\end{align*}
\]

\[
M_1 = \{p(1), p(2), p(3)\} \quad \text{and} \quad M_2 = \{p(1), p(2), p(3), p(5), q\}.
\]

- Furthermore, our experiments with ASP-CLP(Agg) indicate that the cost of communication between the constraint solver and the answer set solvers is significant for large instances.

New Semantics

- In this work, we explore an alternative to ASP-CLP, called Smodels\textsuperscript{A}, that follows a new semantics.

- Aggregate Solution:

- A solution of an aggregate $c$ is a pair $< S_1, S_2 >$ of disjoint sets of ground atoms such that for every interpretation $M$, if $S_1 \subseteq M$ and $S_2 \cap M = \emptyset$ then $c$ is satisfied by $M$.

- Let $\text{SOLN}(c)$ denotes the set of all solutions of $c$.

- Example:

\[
\begin{align*}
& \text{Let } c = \text{sum}(\{X, p(X)\}) < 5 \quad \text{and let } B_p = \{p(1), p(2), p(3)\} \\
& \text{SOLN}(c) = \{ \\
& <\{p(1)\}, \{p(2)\}>, <\{p(1)\}, \{p(3)\}>, <\{p(1)\}, \{p(2), p(3)\}>, \\
& <\{p(2)\}, \{p(3), p(1)\}>, <\{p(2)\}, \{p(3)\}>, <\{p(2)\}, \{p(3)\}>, \\
& <\{p(3)\}, \{p(2), p(1)\}>, <\emptyset, \{p(2)\}>, <\emptyset, \{p(2), p(1)\}>, <\emptyset, \{p(3)\}>, <\emptyset, \{p(3)\}>, \}
\end{align*}
\]
New Semantics

- Set of minimal solutions of $c$ is $S_c = \{\langle \emptyset, \{p(2)\}\rangle, \langle \emptyset, \{p(3)\}\rangle\}$.

- **Unfolding of an Aggregate:**
  The unfolding of an aggregate $c$ w.r.t. its solution $S = <S_1, S_2>$, denoted by $c(S)$, is the conjunction $S_1 \land \neg S_2$.

- **Unfolding of a Rule:**
  The unfolding of a rule $r$ of the form:
  
  $a : - c_1, \ldots, c_k, a_1, \ldots, a_m, \neg b_1, \ldots, \neg b_m$
  
  consists of rules of the form:
  
  $a : - c'_1, \ldots, c'_k, a_1, \ldots, a_m, \neg b_1, \ldots, \neg b_m$
  
  where each $c'_i$ is an unfolding of $c_i$ w.r.t. some solution $c_i$.

Examples

- Let $P_1$ be the program
  
  $q : - \text{sum}({\{X, p(X)\}}) > 10.$
  
  The only solution of $\text{sum}({\{X, p(X)\}}) > 10$ is $\langle \{p(1), p(2), p(3), p(5), \emptyset\}\rangle$ and unfolding($P_1$) contains:
  
  
  which has $M_1 = \{p(1), p(2), p(3)\}$ as its only answer set.

- Let $P_2$ be the program
  
  $p(2), p(1) : - \text{min}({X, p(X)}) \geq 2.$
  
  The only solution of $\text{min}({X, p(X)}) \geq 2$ is $\langle \{p(2)\}, \{p(1)\}\rangle$ and unfolding($P_2$) = $\{p(2), p(1) : - p(2), \neg p(1)\}$. unfolding($P_2$) does not have answer sets.
**Smodels** System

- The implementation of the **Smodels** is straightforward and follows the semantics described earlier by:
  - Computing the minimal solution set of aggregate literals.
  - Computing the unfolding of the program based on the notion of the minimal solution sets. The unfolding of a program with aggregates is a normal logic program.
  - Computing the answer sets of the resulting unfolded program using off-the-shelf systems.


---

**Overall System Structure**

- The overall structure of the system consists of five stages.
- The **Preprocessor Module**, in the 1\textsuperscript{st} stage, is mainly used for rewriting the aggregate literals in a format acceptable by **LPARSE**.
- In the 2\textsuperscript{nd} and 4\textsuperscript{th} stages, **LPARSE** is used. In the last stage, **Smodels** is used to compute the answer sets for the unfolded program.
- In the 3\textsuperscript{rd} stage, the **Transformer Modules**, which is the major component in our system, is used for computing the unfolding of the input programs.
Transformer Module

Evaluation

<table>
<thead>
<tr>
<th>Program</th>
<th>Instance</th>
<th>Smodels Time</th>
<th>Conqex Time</th>
<th>Transformer Time</th>
<th>DLV4* Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Company Control</td>
<td>20</td>
<td>0.010</td>
<td>0.00</td>
<td>0.00</td>
<td>N/A</td>
</tr>
<tr>
<td>Company Control</td>
<td>40</td>
<td>0.020</td>
<td>0.00</td>
<td>0.00</td>
<td>N/A</td>
</tr>
<tr>
<td>Company Control</td>
<td>80</td>
<td>0.030</td>
<td>0.00</td>
<td>2.850</td>
<td>N/A</td>
</tr>
<tr>
<td>Company Control</td>
<td>120</td>
<td>0.040</td>
<td>0.00</td>
<td>12.100</td>
<td>N/A</td>
</tr>
<tr>
<td>Shortest Path</td>
<td>20</td>
<td>0.220</td>
<td>0.05</td>
<td>0.740</td>
<td>N/A</td>
</tr>
<tr>
<td>Shortest Path</td>
<td>50</td>
<td>0.700</td>
<td>0.13</td>
<td>2.640</td>
<td>N/A</td>
</tr>
<tr>
<td>Shortest Path</td>
<td>50</td>
<td>3.519</td>
<td>0.51</td>
<td>13.400</td>
<td>N/A</td>
</tr>
<tr>
<td>Shortest Path (All Pairs)</td>
<td>20</td>
<td>6.020</td>
<td>1.15</td>
<td>25.400</td>
<td>N/A</td>
</tr>
<tr>
<td>Party Invitations</td>
<td>40</td>
<td>0.040</td>
<td>0.00</td>
<td>0.00</td>
<td>N/A</td>
</tr>
<tr>
<td>Party Invitations</td>
<td>80</td>
<td>0.020</td>
<td>0.01</td>
<td>0.00</td>
<td>N/A</td>
</tr>
<tr>
<td>Party Invitations</td>
<td>160</td>
<td>0.050</td>
<td>0.02</td>
<td>0.050</td>
<td>N/A</td>
</tr>
<tr>
<td>Seating</td>
<td>16-4/4</td>
<td>11.40</td>
<td>3.72</td>
<td>0.330</td>
<td>4.337</td>
</tr>
<tr>
<td>Employee Raise</td>
<td>15/5</td>
<td>0.87</td>
<td>0.97</td>
<td>0.140</td>
<td>2.780</td>
</tr>
<tr>
<td>Employee Raise</td>
<td>21/15</td>
<td>2.88</td>
<td>1.75</td>
<td>1.770</td>
<td>6.255</td>
</tr>
<tr>
<td>Employee Raise</td>
<td>24/20</td>
<td>3.13</td>
<td>25.03</td>
<td>2.420</td>
<td>26.50</td>
</tr>
<tr>
<td>Employee Raise</td>
<td>25/20</td>
<td>3.42</td>
<td>8.38</td>
<td>5.20</td>
<td>3.95</td>
</tr>
<tr>
<td>NM1</td>
<td>125</td>
<td>1.19</td>
<td>0.07</td>
<td>1.69</td>
<td>N/A</td>
</tr>
<tr>
<td>NM1</td>
<td>150</td>
<td>1.60</td>
<td>0.18</td>
<td>1.30</td>
<td>N/A</td>
</tr>
<tr>
<td>NM2</td>
<td>125</td>
<td>1.44</td>
<td>0.23</td>
<td>0.80</td>
<td>N/A</td>
</tr>
<tr>
<td>NM2</td>
<td>150</td>
<td>2.08</td>
<td>0.34</td>
<td>1.28</td>
<td>N/A</td>
</tr>
</tbody>
</table>
Conclusions and Future Work

This system differs from our previous system in two ways:
- It implements a different intuitive semantics which leads only to minimal models.
- It does not modify LPARSE and Smodels.

The result of our initial experiments shows that this direction is promising.

Our focus in the near future is to optimize the performance of the system by:
- Improving the rule expander to reduce the size of the unfolding program.
- Improving the aggregate solver to allow more than one grouping variable.