Smodels^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
- Smodels^A System
- Evaluation
- Conclusion and Future work

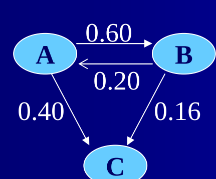
Motivation

- 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^{A}).

Motivation

- Example (Company Control)

```
control_shares(X,Y,N) :- owns(X,Y,N).
control_shares(X,Y,N) :- company(X), control(X,Z),
                         owns(Z,Y,N).
control(X,Y) :- company(X), company(Y),
                sum({{A, control_shares(X,Y,A)}})>50.
```



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:

```
p(1).      p(2).      p(3).
q  :- sum({{X, p(X)}}) > 10.
p(5) :- q.
```

$\mathbf{M_1}$ = {p(1), p(2), p(3)} and $\mathbf{M_2}$ = {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^{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 SOLN(*c*) denotes the set of all solutions of *c* .
- Example:

  Let *c* be *sum({{X, p(X)}}) < 5* and let B_p = {p(1), p(2), p(3)}

  SOLN(*c*) = {

  $<\{p(1)\}, \{p(2)\}>$, $<\{p(1)\}, \{p(3)\}>$, $<\{p(1)\}, \{p(2), p(3)\}>$,
  $<\{p(1),p(2)\}, \{p(3)\}>$, $<\{p(1),p(3)\}, \{p(2)\}>$, $<\{p(2)\}, \{p(3)\}>$,
  $<\{p(2)\}, \{p(3),p(1)\}>$, $<\{p(3)\}, \{p(2)\}>$, $<\{p(3)\}, \{p(2)\}>$,
  $<\{p(3)\}, \{p(2),p(1)\}>$, $<\emptyset, \{p(2)\}>$, $<\emptyset, \{p(2),p(1)\}>$, $<\emptyset, \{p(3)\}>$,
  $<\emptyset, \{p(3),p(1)\}>$, $<\emptyset, \{p(3),p(2)\}>$, $<\emptyset, \{p(3),p(2),p(1)\}>$

  }

New Semantics

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

- 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 $S_1 \wedge \text{not } S_2$.

- Unfolding of a Rule:

  The unfolding of a rule *r* of the form:

  $$a \text{ :- } c_1, \ldots, c_k, a_1, \ldots, a_n, \text{not } b_1, \ldots, \text{not } b_m$$

  consists of rules of the form:

  $$a \text{ :- } c'_1, \ldots, c'_k, a_1, \ldots, a_n, \text{not } b_1, \ldots, \text{not } b_m$$

  where each c_i' is an unfolding of c_i w.r.t. some solution c_i.

Examples

- Let P_1 be the program

  p(1). p(2). p(3). p(5) :- q.

  q:- sum({{X, p(X)}}) > 10.

  The only solution of sum({{X, p(X)}}) > 10 is $\langle\{p(1), p(2), p(3), p(5)\}, \emptyset\rangle$

  and unfolding(P_1) contains:

  p(1). p(2). p(3). p(5) :- q. q:- p(1), p(2), p(3), p(5).

  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):- min({X, p(X)}) ≥ 2.

  The only solution of min({X, p(X)}) ≥ 2 is $\langle\{p(2)\}, \{p(1)\}\rangle$

  and unfolding(P_2) = {p(2). p(1) :- p(2), not p(1).}.

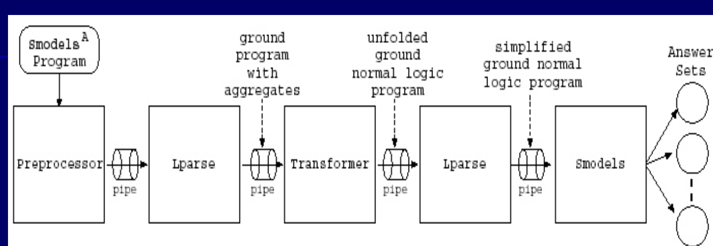
  unfolding(P_2) does not have answer sets.

Smodels^A System

- The implementation of the ***Smodels^A*** 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.

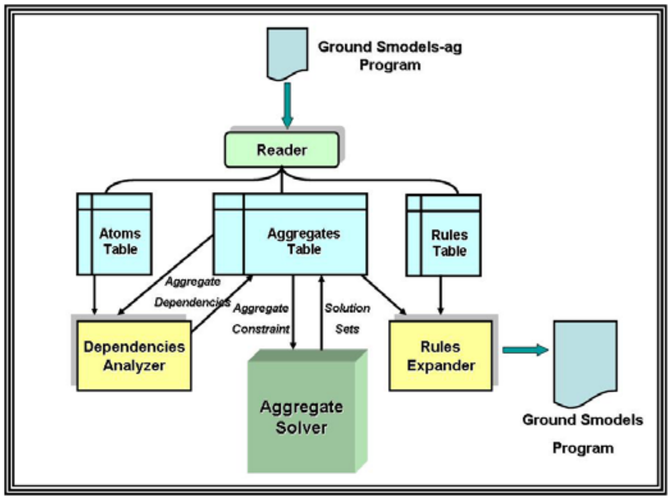
System is available at http://www.cs.nmsu.edu/~ielkaban/asp-aggr.html

Overall System Structure



- The overall structure of the system consists of five stages.
- The *Preprocessor Module*, in the 1st stage, is mainly used for rewriting the aggregate literals in a format acceptable by ***LPARSE***.
- In the 2nd and 4th stages, ***LPARSE*** is used. In the last stage, ***Smodels*** is used to compute the answer sets for the unfolded program.
- In the 3rd 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

Program	Instance	Smodels Time	Cmodels Time	Transformer Time	DLV^{A} Time
Company Control	20	0.010	0.00	0.080	N/A
Company Control	40	0.020	0.00	0.340	N/A
Company Control	80	0.030	0.00	2.850	N/A
Company Control	120	0.040	0.030	12.100	N/A
Shortest Path	20	0.220	0.05	0.740	N/A
Shortest Path	30	0.790	0.13	2.640	N/A
Shortest Path	50	3.510	0.51	13.400	N/A
Shortest Path (All Pairs)	20	6.020	1.15	35.400	N/A
Party Invitations	40	0.010	0.00	0.010	N/A
Party Invitations	80	0.020	0.01	0.030	N/A
Party Invitations	160	0.050	0.02	0.050	N/A
Seating	16/4/4	11.40	3.72	0.330	4.337
Employee Raise	15/5	0.57	0.87	0.140	2.750
Employee Raise	21/15	2.88	1.75	1.770	6.235
Employee Raise	24/20	3.13	25.03	2.420	26.50
Employee Raise	25/20	3.42	8.38	5.20	3.95
NM1	125	1.10	0.07	1.00	N/A
NM1	150	1.60	0.18	1.30	N/A
NM2	125	1.44	0.23	0.80	N/A
NM2	150	2.08	0.34	1.28	N/A

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.