Answer Set Programming via Examples

Yuliya Lierler

University of Nebraska at Omaha

January 24, 2013
What is Answer Set Programming (ASP)

- ASP is a declarative programming paradigm intended to solve difficult (NP-hard) combinatorial search problems (Marek and Truszczyński, 2000; Niemelä, 2000).
  - Declarative programming: Integer Linear Programming, SQL
- Combinatorial search problems consist of finding the combinations of a discrete set of items that satisfy specified requirements (constraints).
- Such problems are often NP-hard and occur in various areas in engineering and science applications.
- ASP origins go back to semantic foundations of Prolog: answer sets (Gelfond and Lifschitz, 1988)
ASP applications

- planning
- model checking
- logical cryptanalysis
- computational biology
- computational linguistics
- space shuttle control
- robotics
- machine code optimization
- automatic music composition
- ...

(Gelfond and Lifschitz, 1988): answer sets

Grounders: GRINGO, DLV, LPARSE
ASP solvers: SMODELS, DLV, CMODELS, CLASP,…
Under one roof: CLINGO (GRINGO+CLASP), DLV
A program consists of rules

\[ a_0 \leftarrow a_1, \ldots, a_m, \text{not } a_{m+1}, \ldots, \text{not } a_n \]

Useful generalizations:

- atoms with variables
- special constructs: choice rules and constraints
Answer Sets

Input: 

\[ p \leftarrow q. \] 
\[ p \leftarrow q. \]
\[ q \leftarrow \text{not } r. \] 
\[ p \leftarrow \text{not } q. \]
\[ q \leftarrow \text{not } p. \]

Output: 

Answer: 1
Answer set: 
Answer: 1
Answer set: q p
Answer: 1
Answer set: p
Answer: 2
Answer set: q

The idea of ASP is to represent the given search problem by a logic program so that the solutions correspond to its answer sets.
Choice Rules and Constraints

Input:

\[ p :- q. \]
\[ q :- not r. \]

Output:

Answer: 1
Answer set: q p

Answer: 1
Answer set: q p

\[ \{s, t\} :- p. \]

Answer: 2
Answer set: q p s t

Answer: 3
Answer set: q p t

Answer: 4
Answer set: q p s

Yuliya Lierler

Answer Set Programming via Examples
Input:

\[ p \leftarrow q. \]
\[ q \leftarrow \neg r. \]
\[ \{s, t\} \leftarrow p. \]

\[ \leftarrow s, \neg t. \]

Output:

Answer: 1
Answer set: \( q \ p \)

Answer: 2
Answer set: \( q \ p \ s \ t \)

Answer: 3
Answer set: \( q \ p \ t \)

NO MORE Answer: 4
Answer set: \( q \ p \ s \)
Input:  

\begin{align*} 
p(a) . & \quad \text{Answer: 1} 
p(b) . & \quad \text{Answer set: } p(b) \ p(a) 
\{q(X) : p(X)\} . & \quad \text{Answer: 2} 
\text{Answer set: } q(a) \ p(b) \ p(a) 
\% \ \{q(a), \ q(b)\} . & \quad \text{Answer: 3} 
\text{Answer set: } q(b) \ p(b) \ p(a) 
\end{align*}  

Output:  

\begin{align*} 
\text{Answer set: } q(b) \ q(a) \ p(b) \ p(a) 
\end{align*}
A clique of 4 in a graph is a set of 4 pairwise adjacent vertexes.

Given:

```
node(1). node(2). node(3). node(4). node(5).
edge(1,2). edge(1,3). edge(1,4). edge(1,5).
edge(2,3). edge(2,4). edge(2,5).
edge(3,4). edge(3,5).
```
% GENERATE
% at least 4 inCliq nodes form a potential solution
  4{inCliq(X) : node(X)}.

% DEFINE
  adjacent(X,Y) :- edge(X,Y).
  adjacent(X,Y) :- edge(Y,X).

% TEST
% Forbidden: different X and Y are inCliq, but not adjacent
false :- X!=Y, inCliq(X), inCliq(Y), not adjacent(X,Y).

Answer set 1: inCliq(1), inCliq(2), inCliq(3), inCliq(4)
Answer set 2: inCliq(1), inCliq(2), inCliq(3), inCliq(5)
Mr and Mrs Astor, Mr and Mrs Blake, Mr and Mrs Crane, and Mr and Mrs Davis were seated around a circular table. Mrs Astor was insulted by Mr Blake, who sat next to her on her left. Mr Blake was insulted by Mrs Crane, who sat opposite him across the center of the table. Mrs Crane was insulted by the hostess, who was the only person to sit next to each one of a married couple. The hostess was insulted by the only person to sit next to each one of two men. Who insulted the hostess? Mrs. Davis is the hostess and she is seated at place 0.

⇒ Constraint Satisfaction Problem
Mr and Mrs Astor, Mr and Mrs Blake, Mr and Mrs Crane, and Mr and Mrs Davis were seated around a circular table. Mrs Astor was insulted by Mr Blake, who sat next to her on her left. Mr Blake was insulted by Mrs Crane, who sat opposite him across the center of the table. Mrs Crane was insulted by the hostess, who was the only person to sit next to each one of a married couple. The hostess was insulted by the only person to sit next to each one of two men. Who insulted the hostess? Mrs. Davis is the hostess and she is seated at place 0.

⇒ Constraint Satisfaction Problem
Given

spot(0..7).

%male(mrAstor). male(mrBlake). ...
male(mrAstor;mrBlake;mrCrane;mrDavis).

person(mrAstor;mrsAstor;mrBlake;mrsBlake;
    mrCrane;mrsCrane;mrDavis;mrsDavis).

married(mrAstor,mrsAstor).
married(mrBlake,mrsBlake).
married(mrCrane,mrsCrane).
married(mrDavis,mrsDavis).

%married is symmetric
married(P,P1) :- married(P1,P), person(P;P1).
%% GENERATE
%% Mr and Mrs Astor, Mr and Mrs Blake, Mr and Mrs Crane, Mr and Mrs Davis were seated around a circular table.

% every person is assigned a spot
1{place(P,S): spot(S)}1:-person(P).

%% DEFINE
% two places at a table are opposite
opposite(S,S+4) :- spot(S;S+4).

% opposite is symmetric
opposite(S1,S2) :- opposite(S2,S1), spot(S1;S2).
% two people cannot occupy the same spot
:- place(P1,S), place(P2,S), P1!=P2, spot(S), person(P1;P2).

%%% Mrs Astor was insulted by Mr Blake, who sat
%%% next to her on her left.
%%% Mr Blake sat next to Mrs Astor on her left.
:- place(mrsAstor,S), not place(mrBlake,S+1), spot(S).

%%% Mr Blake was insulted by Mrs Crane, who sat opposite
%%% him across the center of the table.
%%% Mr Blake sat next to Mrs Astor on her left.
:- place(mrBlake,S1), not place(mrsCrane,S2), opposite(S1,S2), spot(S1;S2).

...
/u/yuliya % lparse diners | cmodels 0
cmodels version 3.79 Reading...done
Program is tight.
Calling SAT solver zChaff 2007.3.12 ...
Answer: 1

Answer set: place(mrsDavis,0) place(mrDavis,4)

place(mrsCrane,6) place(mrCrane,3)

place(mrsBlake,5) place(mrBlake,2)

place(mrsAstor,1) place(mrAstor,7)

insult(mrCrane)
Roughly 100 people live in the 21st Street Co-op at any given time. There is a core subset of labor that must be done or the house falls apart (kitchen, maintenance, groundskeeping, etc).

Most people are required to do 4 hours of work and some 2 hours. Each labor has their time and duration (eg, Monday Lunch cleanup 1, 2 hours). All labor is required and nobody has to do more than 4 hours.
Roughly 100 people live in the 21st Street Co-op at any given time. There is a core subset of labor that must be done or the house falls apart (kitchen, maintenance, groundskeeping, etc).

Most people are required to do 4 hours of work and some 2 hours. Each labor has their time and duration (eg, Monday Lunch cleanup 1, 2 hours). All labor is required and nobody has to do more than 4 hours.

⇒ Scheduling
Given

ASP-language **CLINGO** (**GRINGO**):

Sample instance of 20 people and 50 jobs:

```
person_hours(1,2).
person_hours(2,2).
...
person_hours(6,4).
person_hours(7,4).
...
wid_wtype_day_time_duration(1, kitchen, mon, 12, 1).
wid_wtype_day_time_duration(2, kitchen, mon, 12, 1).
...```
%Definitions from the input instance

%work id definition
wid(X):-wid_wtype_day_time_duration(X,_,_,_,_,_).

%work id - duration relation definition
wid_duration(X,D):-wid_wtype_day_time_duration(X,_,_,_,_,D).

%person id definition
pid(X):-person_hours(X,_).
% each "job" has to be assigned to exactly one person
1{assign(P,W):pid(P)}1:-wid(W).
% a person may not be assigned multiple jobs
% such that their sum duration is greater than
% the hours he has to work

:- pid(P), person_hours(P,H),
    S = #sum[assign(P,W):wid_duration(W,D)=D],
    S>H.

% Here we ASSUME that the DURATION of each JOB is
% at least ONE HOUR

:-N=#count{assign(P,W)}, pid(P), N>4.
%output predicate
assigned(P,W,Ty,D,Ti,Du):-assign(P,W),
    wid_wtype_day_time_duration(W,Ty,D,Ti,Du).

%output only assigned relation
#show assigned(P,W,Ty,D,Ti,D).
#hide.
CLINGO available at http://potassco.sourceforge.net/

/u/yuliya % clingo schedInstance sched.cl
Answer: 1
assigned(6,50,maintenance,fr,18,1)
assigned(6,49,maintenance,fr,18,1)
assigned(7,48,maintenance,fr,10,2)
assigned(7,47,maintenance,fr,10,1)
assigned(7,46,maintenance,fr,10,1)
...

SATISFIABLE

Models : 1+
Time : 3.670
Propositional Satisfiability (SAT) is one of the most studied problems in computational logic.

SAT is the problem of determining if the atoms of a given propositional formula can be assigned truth values in such a way that the formula is evaluated to True.

\( a \lor b \) is satisfiable, it evaluates to True if \( a \) or \( b \) are assigned True.

\( a \land \neg a \) is unsatisfiable.

Modern SAT solvers zchaff, minisat, plingeling,... find satisfying assignments, models, for problems with millions of clauses and hundreds of thousands of atoms.
System **cmODELS** implements SAT-based methods for generating answer sets by incorporating SAT solvers for its computation.

SAT solvers:

1. RELSAT
2. ZCHAFF
3. MINISAT
4. SIMO

http://www.cs.utexas.edu/users/tag/cmodels/
**1st ASP System Competition (LPNMR 2007):** 10 systems

<table>
<thead>
<tr>
<th>Place</th>
<th>MGS</th>
<th>SCore</th>
<th>SLparse</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>dlv</td>
<td>CLASP</td>
<td>CLASP</td>
</tr>
<tr>
<td>2</td>
<td>pbmodels</td>
<td>smodels</td>
<td>pbmodels</td>
</tr>
<tr>
<td>3</td>
<td>clasp</td>
<td>cmodesl</td>
<td>smodels</td>
</tr>
</tbody>
</table>

**2d ASP System Competition (LPNMR 2009):** 16 systems

<table>
<thead>
<tr>
<th>Place</th>
<th>Decision Problem</th>
<th>Decision Problem in NP</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>claspfolio</td>
<td>claspfolio</td>
</tr>
<tr>
<td>2</td>
<td>cmodesl</td>
<td>cmodesl</td>
</tr>
<tr>
<td>3</td>
<td>dlv</td>
<td>idp</td>
</tr>
</tbody>
</table>
Challenges in ASP

- Grounding
- Solving
- Synergy of grounding and Solving
- Multi-logics: constraint answer set programming and beyond
- Modeling: is there declarative programming?
- Applying ASP in Real Life