Answer Set Programming via Examples

Yuliya Lierler

University of Kentucky

What is Answer Set Programming (ASP)

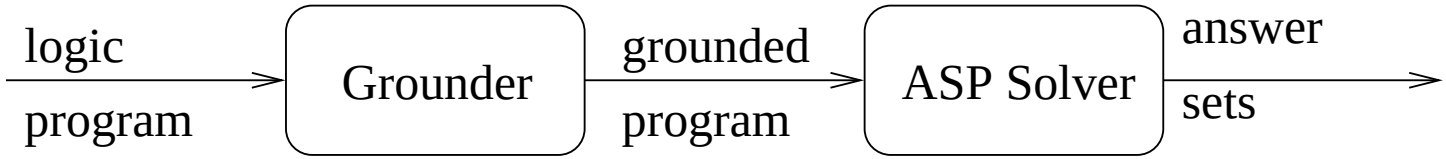
ASP is a declarative programming paradigm intended to solve difficult (NP-hard) combinatorial search problems (Marek and Truszczyński, 2000; Niemelä, 2000).

ASP applications:

- planning
- model checking
- logical cryptanalysis
- computational biology
- ...

ASP Solvers

(Gelfond and Lifschitz, 1988): answer sets



Grounders: GRINGO, DLV, LPARSE

ASP solvers: SMODELS, DLV, CMODELS, CLASP,...

Under one roof: CLINGO (GRINGO+CLASP), DLV

ASP Programs

A program consists of rules

$$\underbrace{a_0}_{\text{Head}} \leftarrow \underbrace{a_1, \ldots, a_m,\ not\ a_{m+1}, \ldots,\ not\ a_n}_{\text{Body}}$$

Useful generalizations:

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

Answer Sets

Input:	Output:
`p :- q.`	`Answer: 1` `Answer set:`
`p :- q.` `q :- not r.`	`Answer: 1` `Answer set: q p`

`p :- not q.`
`q :- not 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.
{s,t} :- p.
:- s, not t.
```

Output:

```
Answer: 1
Answer set: t s p q
Answer: 2
Answer set: p q
Answer: 3
Answer set: t p q
```

ASP Programs with Variables

Input:

```
p(a).
p(b).
{q(X) : p(X)}.

% {q(a), q(b)}.
```

Output:

```
Answer: 1
Answer set: p(b) p(a)
Answer: 2
Answer set: q(a) p(b) p(a)
Answer: 3
Answer set: q(b) p(b) p(a)
Answer: 4
Answer set: q(b) q(a) p(b) p(a)
```

Using ASP to Find a Large Clique of 5

A *clique* in a graph is a set of pairwise adjacent vertices.

```
vertex(1..99).  % 1,...,99 are vertices
edge(3,7).      % 3 is adjacent to 7
. . .
edge(X,Y) :- edge(Y,X), vertex(X;Y).

%% GENERATE
5 {in(X) : vertex(X)}.

%% TEST
:- in(X), in(Y), vertex(X), vertex(Y),
   X!=Y, not edge(X,Y).
```

Diners

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.

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 and DEFINE

```
%% 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).
```

TEST

```
% 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.
%%
:- place(mrBlake,S1), not place(mrsCrane,S2),
   opposite(S1,S2), spot(S1;S2).
...
```

Output

```
/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)
```

Gatlin Johnson: Co-op Scheduling

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.

Given

ASP-language CLINGO (GRINGO):

`http://ai.ustc.edu.cn/cn/seminar/files/guide.pdf`

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).
...
```

DEFINE

```
%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,_).
```

GENERATE

```
%each "job" has to be assigned to exactly one person
1{assign(P,W):pid(P)}1:-wid(W).
```

TEST

```
% 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

```
%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.
```

Solver Output

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
```

Behind ASP Solvers

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*.

$a \vee b$ is satisfiable, it evaluates to *True* if a or b are assigned *True*.

$a \wedge \neg a$ is unsatisfiable.

Modern SAT solvers ZCHAFF, MINISAT,... find satisfying assignments, models, for problems with millions of clauses and hundreds of thousands of atoms.

Cmodels

System CMODELS implements SAT-based methods for generating answer sets.

SAT solvers:

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

http://www.cs.utexas.edu/users/tag/cmodels/

ASP Competitions

1st ASP System Competition (LPNMR 2007): 10 systems

Place	MGS	SCore	SLparse
1	DLV	CLASP	CLASP
2	PBMODELS	SMODELS	PBMODELS
3	CLASP	CMODELS	SMODELS

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

Place	Decision Problem	Decision Problem in NP
1	CLASPFOLIO	CLASPFOLIO
2	CMODELS	CMODELS
3	DLV	IDP

Cmodels in Use

- reconstruction of phylogenies in historical linguistics and biology; robot control (Sabancı University, Turkey)
- machine code optimization, automatic music composition (University of Bath, UK)
- model checking of abstract state machines (Simon Fraser University, Canada)