Stable Models Semantics and Answer Set Programming

CSE 505 – Computing with Logic

Stony Brook University

http://www.cs.stonybrook.edu/~cse505

General Logic Programs

- A general program is a collection of rules of the form:

$$\mathbf{a} \leftarrow \mathbf{a_1}, \ \ldots, \ \mathbf{a_n}, \ \textbf{not } \mathbf{a_{n+1}}, \ldots, \ \textbf{not } \mathbf{a_{n+k}}.$$

Grounding

- Variables are placeholders for constants.
- ***<u>Grounding</u>*** is the process to “replace variables by constants in all possible ways”
- Example:

```
isInterestedinASP(X) :- attendsASP(X).
attendsASP(john). attendsASP(mary).
```

- After grounding:

```
isInterestedinASP(john) :-
     attendsASP(john).
isInterestedinASP(mary) :-
     attendsASP(mary).
attendsASP(john). attendsASP(mary).
```

Gelfond-Lifschitz transformation

- A general program is a collection of rules of the form:

$$\mathtt{a} \leftarrow \mathtt{a_1,\ \ldots,\ a_n,\ not\ a_{n+1},\ldots,\ not\ a_{n+k}.}$$

- Let Π be a program and I be a set of atoms, by Π^I (***Gelfond-Lifschitz transformation***) we denote the positive program obtained from ground(Π) by:
  - Deleting from ground(Π) any rule for that $\{\mathtt{a_{n+1},\ldots,a_{n+k}}\} \cap \mathtt{I} \neq \emptyset$, i.e., the body of the rule contains a naf-atom $\mathtt{not\ a_l}$ and $\mathtt{a_l}$ belongs to $\mathtt{I}$; and
  - Removing all of the naf-atoms from the remaining rules

General Logic Programs

- A set of atoms I is called an ***answer set*** of a program Π if I is the **minimal** model of the program Π^I
- Example: Consider $\Pi_2 = \{$**`a ← not b. b ← not a.`**$\}$. We will show that it has two answer sets $\{$**`a`**$\}$ and $\{$**`b`**$\}$

$S_1 = \emptyset$	$S_2 = \{a\}$	$S_3 = \{b\}$	$S_4 = \{a, b\}$
$\Pi_2^{S_1}$: $a \leftarrow$ $b \leftarrow$	$\Pi_2^{S_2}$: $a \leftarrow$	$\Pi_2^{S_3}$: $b \leftarrow$	$\Pi_2^{S_4}$:
$M_{\Pi_2^{S_1}} = \{a, b\}$	$M_{\Pi_2^{S_2}} = \{a\}$	$M_{\Pi_2^{S_3}} = \{b\}$	$M_{P^{S_4}} = \emptyset$
$M_{\Pi_2^{S_1}} \neq S_1$	$M_{\Pi_2^{S_2}} = S_2$	$M_{\Pi_2^{S_3}} = S_3$	$M_{\Pi_2^{S_4}} \neq S_4$
NO	YES	YES	NO

- Theorem: For every positive program Π, the minimal model of Π, M_Π, is also the unique answer set of Π.

General Logic Programs

- $\Pi_5 = \{\mathbf{p} \leftarrow \mathbf{not\ p.}\}$ does not have an answer set.
  - $S_1 = \emptyset$, then $\Pi^{S1} = \{\mathbf{p} \leftarrow\}$ whose minimal model is $\{\mathbf{p}\}$. $\{\mathbf{p}\} \neq \emptyset$ implies that S_1 is not an answer set of Π_5.
  - $S_2 = \{\mathbf{p}\}$, then $\Pi^{S2} = \emptyset$ whose minimal model is $\emptyset$. $\{\mathbf{p}\} \neq \emptyset$ implies that S_2 is not an answer set of Π_5.
  - This shows that this program does not have an answer set.
- A program may have zero, one, or more than one answer sets:
  - $\Pi_1 = \{a \leftarrow \text{not } b.\}$ has a unique answer set $\{a\}$.
  - $\Pi_2 = \{a \leftarrow \text{not } b.\ b \leftarrow \text{not } a.\}$ has two answer sets: $\{a\}$ and $\{b\}$.
  - $\Pi_3 = \{p \leftarrow a.\ a \leftarrow \text{not } b.\ b \leftarrow \text{not } a.\}$ has two answer sets: $\{a, p\}$ and $\{b\}$
  - $\Pi_4 = \{a \leftarrow \text{not } b.\ b \leftarrow \text{not } c.\ d \leftarrow .\}$ has one answer set $\{d, b\}$.
  - $\Pi_5 = \{p \leftarrow \text{not } p.\}$ No answer set.
  - $\Pi_6 = \{p \leftarrow d, \text{not } p.\ r \leftarrow \text{not } d.\ d \leftarrow \text{not } r.\}$ has one answer set $\{r\}$.

Entailment w.r.t. Answer Set Semantics

- For a program Π and an atom a, Π ***entails*** a, denoted by $\Pi \models a$, if $a \in S$ for every answer set S of Π.
- For a program Π and an atom a, Π entails $\neg a$, denoted by $\Pi \models \neg a$, if $a \notin S$ for every answer set S of Π.
- If neither $\Pi \models a$ nor $\Pi \models \neg a$, then we say that a is ***unknown*** with respect to Π.
- Examples:
  - $\Pi_1 = \{a \leftarrow \text{not } b.\}$ has a unique answer set $\{a\}$. $\Pi_1 \models a$, $\Pi_1 \models \neg b$.
  - $\Pi_2 = \{a \leftarrow \text{not } b.\ b \leftarrow \text{not } a\}$ has two answer sets: $\{a\}$ and $\{b\}$. Both a and b are unknown w.r.t. Π_2.
  - $\Pi_3 = \{p \leftarrow a.\ a \leftarrow \text{not } b.\ b \leftarrow \text{not } a.\}$ has two answer sets: $\{a, p\}$ and $\{b\}$. Everything is unknown.
  - $\Pi_4 = \{p \leftarrow \text{not } p.\}$ No answer set. p is unknown.

Answer Sets of Programs with Constraints

- For a set of ground atoms S and a ***constraint*** c

$$\leftarrow a_1, \ldots, a_n, \text{not } a_{n+1}, \text{not } a_{n+k}.$$

we say that c is ***satisfied*** by S if $\{a_1, \ldots, a_n\} \setminus S \neq \emptyset$ or $\{a_{n+1},\ldots,a_{n+k}\} \cap S \neq \emptyset$.

- Let Π be a program with constraints.
- Let $\Pi_O = \{r \mid r \in \Pi$, r has non-empty head$\}$ (Π_O is the set of normal logic program rules in Π)
- Let $\Pi_C = \Pi \setminus \Pi_O$ (Π_C is the set of constraints in Π)
- A set of atoms S is an answer sets of a program Π if it is an answer set of Π_O and satisfies all the constraints in ground (Π_C)

Answer Sets of Programs with Constraints

- Example:
  - $\Pi_1 = \{a \leftarrow \text{not } b.\ b \leftarrow \text{not } a.\}$ has two answer sets $\{a\}$ and $\{b\}$
  - But, $\Pi_2 = \{$
    $$a \leftarrow \text{not } b.$$
    $$b \leftarrow \text{not } a.$$
    $$\leftarrow \text{not } a. \quad \}$$

  has only one answer set $\{a\}$.
  - But, $\Pi_3 = \{$
    $$a \leftarrow \text{not } b.$$
    $$b \leftarrow \text{not } a.$$
    $$\leftarrow a. \quad \}$$

  has only one answer set $\{b\}$.

Computing Answer Sets

- Complexity: The problem of determining the existence of an answer set for finite propositional programs (programs without function symbols) is NP-complete.
- For programs with disjunctions, function symbols, etc. it is much higher.
- A consequence of this property is that there exists no polynomial-time algorithm for computing answer sets.

Answer set solvers

- Programs that compute answer sets of (finite and grounded) logic programs.
- Two main approaches:
  - Direct implementation: Due to the complexity of the problem, most solvers implement a variation of the generate-and-test algorithm
    - Smodels http://www.tcs.hut.fi/Software/smodels/
    - DLV http://www.dbai.tuwien.ac.at/proj/dlv/
    - deres http://www.cs.engr.uky.edu/ai/deres.html
  - Using SAT solvers: A program Π is translated into a satisfiabilty problem FΠ and a call to a SAT solver is made to compute solution of FΠ.

  The main task of this approach is to write the program for the conversion from Π to FΠ

    - Potassco: http://potassco.sourceforge.net/ (clasp, gringo, …)
    - Cmodels http://www.cs.utexas.edu/users/tag/cmodels.html
    - ASSAT http://assat.cs.ust.hk/

Example: Graph Coloring

- Given a (bi-directed) graph and three colors red, green, and yellow. Find a color assignment for the nodes of the graph such that no edge of the graph connects two nodes of the same color.
  - Graph representation:
    - The nodes: **`node(1).`** … **`node(n).`**
    - The edges: **`edge(i, j).`**
  - Each node is assigned one color:
    - the three rules:

```
color(X, red) ← node(X), not color(X, green), not color(X, yellow).
color(X, green) ← node(X), not color(X, red), not color(X, yellow).
color(X, yellow) ← node(X), not color(X, green), not color(X, red).
```

  - No edge connects two nodes of the same color:

```
← edge(X, Y ), color(X, C), color(Y, C).
```

Example: Graph Coloring

```
node(1). node(2). node(3).
edge(1,2). edge(2,3). edge(3,1).
color(X,red):- node(X), not color(X,green), not color(X, yellow).
color(X,green):- node(X), not color(X,red), not color(X, yellow).
color(X,yellow):- node(X), not color(X,green), not color(X, red).
:- edge(X,Y), color(X,C), color(Y,C).
```

- Try with

```
clingo -n 0 color.lp
```

```
Answer: 1
node(1) node(2) node(3) edge(1,2) edge(2,3) edge(3,1) color(1,red) color(2,green) color(3,yellow)
Answer: 2
node(1) node(2) node(3) edge(1,2) edge(2,3) edge(3,1) color(1,red) color(2,yellow) color(3,green)
Answer: 3
node(1) node(2) node(3) edge(1,2) edge(2,3) edge(3,1) color(1,green) color(2,red) color(3,yellow)
Answer: 4
node(1) node(2) node(3) edge(1,2) edge(2,3) edge(3,1) color(1,yellow) color(2,red) color(3,green)
Answer: 5
node(1) node(2) node(3) edge(1,2) edge(2,3) edge(3,1) color(1,green) color(2,yellow) color(3,red)
Answer: 6
node(1) node(2) node(3) edge(1,2) edge(2,3) edge(3,1) color(1,yellow) color(2,green) color(3,red)
Models      : 6
```