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:

\[ a \leftarrow a_1, \ldots, a_n, \text{not } a_{n+1}, \ldots, \text{not } a_{n+k}. \]
Grounding

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

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

- After grounding:

  ```prolog
  isInterestedinASP(john):- attendsASP(john).
  isInterestedinASP(mary):- attendsASP(mary).
  ```
Gelfond-Lifschitz transformation

• A general program is a collection of rules of the form: 
  \( a \leftarrow a_1, \ldots, a_n, \text{not } a_{n+1}, \ldots, \text{not } a_{n+k} \).

• Let \( \Pi \) be a program and \( I \) be a set of atoms, by \( \Pi^I \) (Gelfond-Lifschitz transformation) we denote the positive program obtained from \( \text{ground}(\Pi) \) by:
  • Deleting from \( \text{ground}(\Pi) \) any rule for that 
    \( \{a_{n+1}, \ldots, a_{n+k}\} \cap I \neq \emptyset \), i.e., the body of the rule contains a naf-atom \textbf{not } a_1 \text{ and } a_1 \text{ belongs to } I; \text{ 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 $\Pi$ if $I$ is the **minimal** model of the program $\Pi$.

- Example: Consider $\Pi_2 = \{a \leftarrow \text{not } b. \ b \leftarrow \text{not } a.\}$. We will show that it has two answer sets $\{a\}$ and $\{b\}$.

<table>
<thead>
<tr>
<th>$S_1 = \emptyset$</th>
<th>$S_2 = {a}$</th>
<th>$S_3 = {b}$</th>
<th>$S_4 = {a, b}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\Pi_2^{S_1}$ :</td>
<td>$\Pi_2^{S_2}$ :</td>
<td>$\Pi_2^{S_3}$ :</td>
<td>$\Pi_2^{S_4}$ :</td>
</tr>
<tr>
<td>$a \leftarrow$</td>
<td>$a \leftarrow$</td>
<td>$b \leftarrow$</td>
<td></td>
</tr>
<tr>
<td>$b \leftarrow$</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$M_{\Pi_2^{S_1}} = {a, b}$</td>
<td>$M_{\Pi_2^{S_2}} = {a}$</td>
<td>$M_{\Pi_2^{S_3}} = {b}$</td>
<td>$M_{\Pi_2^{S_4}} = \emptyset$</td>
</tr>
<tr>
<td>$M_{\Pi_2^{S_1}} \neq S_1$</td>
<td>$M_{\Pi_2^{S_2}} = S_2$</td>
<td>$M_{\Pi_2^{S_3}} = S_3$</td>
<td>$M_{\Pi_2^{S_4}} \neq S_4$</td>
</tr>
<tr>
<td>$NO$</td>
<td>$YES$</td>
<td>$YES$</td>
<td>$NO$</td>
</tr>
</tbody>
</table>

- Theorem: For every positive program $\Pi$, the minimal model of $\Pi$, $M_\Pi$, is also the unique answer set of $\Pi$.  

(c) Paul Fodor (CS Stony Brook) and Elsevier
General Logic Programs

- $\Pi_5 = \{p \leftarrow \text{not } p.\}$ does not have an answer set.
  - $S_1 = \emptyset$, then $\Pi^{S_1} = \{p \leftarrow\}$ whose minimal model is $\{p\}$. $\{p\} \neq \emptyset$ implies that $S_1$ is not an answer set of $\Pi_5$.
  - $S_2 = \{p\}$, then $\Pi^{S_2} = \emptyset$ whose minimal model is $\emptyset$. $\{p\} \neq \emptyset$ implies that $S_2$ is not an answer set of $\Pi_5$.
  - 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 $\Pi$ and an atom $a$, $\Pi$ entails $a$, denoted by $\Pi \models a$, if $a \in S$ for every answer set $S$ of $\Pi$.
- For a program $\Pi$ and an atom $a$, $\Pi$ entails $\neg a$, denoted by $\Pi \models \neg a$, if $a \notin S$ for every answer set $S$ of $\Pi$.
- If neither $\Pi \models a$ nor $\Pi \models \neg a$, then we say that $a$ is unknown with respect to $\Pi$.

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. $\Pi_2$.
- $\Pi_3 = \{ p \leftarrow a. \ a \leftarrow \text{not } b. \ b \leftarrow \text{not } a. \}$ has two answer sets: $\{a, p\}$ and $\{b\}$. Everything is unknown.
- $\Pi_4 = \{ p \leftarrow \text{not } p. \}$ No answer set. $p$ is unknown.
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 $\Pi$ be a program with constraints.

Let $\Pi_O = \{r \mid r \in \Pi, r \text{ has non-empty head}\}$ ($\Pi_O$ is the set of normal logic program rules in $\Pi$)

Let $\Pi_C = \Pi \setminus \Pi_O$ ($\Pi_C$ is the set of constraints in $\Pi$)

A set of atoms $S$ is an answer sets of a program $\Pi$ if it is an answer set of $\Pi_O$ and satisfies all the constraints in ground ($\Pi_C$)
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, \( \Pi_2 = \{ \ a \leftarrow \text{not } b. \ b \leftarrow \text{not } a. \ \leftarrow \text{not } a. \ \} \) has only one answer set \( \{a\} \).
  - But, \( \Pi_3 = \{ \ a \leftarrow \text{not } b. \ b \leftarrow \text{not } a. \ \leftarrow \text{a.} \ \} \) 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
    - DLV [http://www.dbai.tuwien.ac.at/proj/dlv/](http://www.dbai.tuwien.ac.at/proj/dlv/)
    - deres [http://www.cs.engr.uky.edu/ai/deres.html](http://www.cs.engr.uky.edu/ai/deres.html)
  - **Using SAT solvers**: A program $\Pi$ is translated into a satisfiability problem $F\Pi$ and a call to a SAT solver is made to compute solution of $F\Pi$.
    - The main task of this approach is to write the program for the conversion from $\Pi$ to $F\Pi$
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

define the nodes and edges:

node(1). node(2). node(3).
edge(1,2). edge(2,3). edge(3,1).

define the color 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).

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