KNOWLEDGE REPRESENTATION, REASONING AND DECLARATIVE PROBLEM SOLVING

Jicheng Zhao

Department of Computer Science and Engg.
Arizona State University
Tempe, AZ 85287
chitta@asu.edu

http://www.public.asu.edu/~cbaral

Class 6: Splitting in Logic Programs and Declarative Problem Solving

Relating answer sets and the program rules

- **Forced atom proposition**: Let S be an answer set of an AnsProlog program Π. For any ground instance - of a rule in Π of the type $A_0 \leftarrow A_1, \cdots, A_m, not\ A_{m+1}, \cdots, not\ A_n$. If $\{A_1, \cdots, A_m\} \subseteq S$ and $\{A_{m+1}, \cdots, A_n\} \cap S = \emptyset$ then $A_0 \in S$.
- **Supporting rule proposition**: If S is an answer set of an AnsProlog program Π then S is supported by Π. That is, if $A_0 \in S$, then there exists a ground instance of a rule in Π of they type $A_0 \leftarrow A_1, \cdots, A_m, not\ A_{m+1}, \cdots, not\ A_n$ such that $\{A_1, \cdots, A_m\} \subseteq S$ and $\{A_{m+1}, \cdots, A_n\} \cap S = \emptyset$.

If A is an Answer Set$A = \{a_1, a_2, a_3\}$ there must be a rule r with $a_1 \in head(r)$. I.e. each answer a_1 must have justification. Also if the body of the program is true, then the head must be in the answer set.

Splitting

- **Example 1** *Given a program P. The top of P is:*

$$p \leftarrow not\ p$$
$$p \leftarrow a$$

The bottom of program P is:

$$a \leftarrow not\ b$$
$$b \leftarrow not\ a$$

There are two choices for models in the bottom, a or b. If you choose a, the second rule in the top of the program declares p true. If you proceed with the other choice, b, the top yields no answer set. As a result, the only answer set to the above program is $\{a, p\}$.

Note:

1. Some programs can be split into multiple layers.

2. When you split the program into layers, if an atom is in the body of a rule in one layer, the same atom cannot in the head of rules in top layers.

Example 2 *Exercise: Given program P:*

$$\begin{aligned} a &\leftarrow not\ b \\ b &\leftarrow not\ a, c \\ c &\leftarrow not\ d \\ d &\leftarrow not\ c \\ e &\leftarrow not\ f \\ f &\leftarrow not\ e \end{aligned}$$

spilt it into layers and find its stable models.

Declarative Problem Solving

- **At least one item is chosen**:

Example 3 *Suppose $p(x)$ means x is possible and $c(x)$ means x is chosen. Suppose we have $p(a)$, $p(b)$ and $p(c)$ and we want to enumerate choices such that at least one of $c(a)$, $c(b)$, and $c(c)$ is true in the answer sets of the program. That is, we want to have a program that has the following answer sets:*

$$\begin{aligned}
A_1 &= \{c(a), p(a), ...\} \\
A_2 &= \{c(b), p(a), ...\} \\
A_3 &= \{c(c), p(a), ...\} \\
A_4 &= \{c(a), c(b), p(a), ...\} \\
A_5 &= \{c(a), c(c), p(a), ...\} \\
A_6 &= \{c(b), c(c), p(a), ...\} \\
A_7 &= \{c(a), c(b), c(c), p(a), ...\}
\end{aligned}$$

The program can be:

$$\begin{aligned}
c(x) &\leftarrow p(x), not\ nc(x) \\
nc(x) &\leftarrow p(x), not\ c(x)
\end{aligned}$$

$$\begin{aligned} p(a) &\leftarrow \\ p(b) &\leftarrow \\ p(c) &\leftarrow \\ some &\leftarrow p(x), c(x) \\ &\leftarrow not\ some \end{aligned}$$

If we further requires that only one of a, b, c can be chosen. The program by adding the following rule satisfies this requirement:

$$\leftarrow c(x), c(y), x! = y.$$

- **To express "Exists" and "forall"**:

  Example 4 *p is true iff there is a block in the table;*

  q is true iff all blocks in the table are blue.

  Suppose the notation are:

  – *ontable(X) – block X is on the table*

  – *blue(X) – block X is blue*

The following example satisfies these requirements. In the sense that p is in the answer set of the program iff there is a block in the table. q in in the answer set of the program iff all blocks in the table are blue.

$$p \leftarrow ontable(X)$$
$$n\_q \leftarrow on\_table(X), not\ blue(X)$$
$$q \leftarrow not\ n\_q$$

- To ensure one atom must be true (or false) in the stable model, we may use the following rule.

$$p \leftarrow not\ p, cond. \qquad (1)$$

  where p is an atom that hasn't occurred in the program. *cond* is the condition that we may ensure to be false.

- We will discuss more declarative problem solving techniques in next class.