Notes on Logic Programming

Artificial Intelligence II — CS575

October 2, 2007

We started with a short discussion on the difficulty of using first order logic in knowledge representation. The example we were looking at is

> Bird normally flies. Penguins do not. Tim is a bird. Tweety is a penguin.

We attempt to formalize the information as a first order logic theory in which the predicates *bird*, *penguin*, and *fly* are introduced. Our first attempt

$$\begin{array}{l} bird(X) \Rightarrow fly(X) \\ penguin(X) \Rightarrow \neg fly(X) \\ penguin(X) \Rightarrow bird(X) \\ bird(tim) \\ penguin(tweety) \end{array}$$

leads to an inconsistent theory.

We realized that we need an extra predicate *ab* (with the same arity as bird) to represent the fact that the bird is abnormal to the rule "bird normally flies." This allows us to represent the information as the theory

$$\begin{array}{l} bird(X) \wedge \neg ab(X) \Rightarrow fly(X) \\ penguin(X) \Rightarrow \neg fly(X) \\ penguin(X) \Rightarrow bird(X) \\ bird(tim) \\ penguin(tweety) \end{array}$$

This helps us to avoid the inconsistence problem (i.e., the theory is now satisfiable. Moreover, it entails $\neg fly(tweety)$. Yet, it does not allow us to conclude that $fly(tim)$ holds.

Logic programming is introduced with the promise that everything is going to be fine :-) The basic definitions and notations in LP are discussed. Among them are

- a logic programming rule with its intuition
- Herbrand universe
- Herbrand base
- Herbrand models of a logic program
- minimal Herbrand model M_P of a ground and positive logic program P and the immediate consequence operator T_P
- minimal Herbrand model M_P of a positive logic program P with variables
- stable models of a logic program, the Gelfond-Lifschitz's reduct

- entailments of a logic program

These definitions and notations can be found in the tutorial on answer set programming on the class website or directly from http://www.cs.nmsu.edu/~tson/classes/fall07-575/classnotes.html. (the first 26 slides).

The subsequent discussion will take a few classes for knowledge representation using logic programming. We started with the "RoundUp" story:

> The Round Up reported that two football players, Derek Dubois and Jeremiah Williams, have been suspended for this Saturday's game. It is mentioned that William is the best wide receiver of the team. Also of note is that William's performance against the Miners was instrumental to the team close win against the Miners.Aggies' fans are worry someas this Saturday's opponent, the Broncos from Boise State will come with no injuries or suspensions,has a much better record thus far into theseason comparing to theMiners as well as the Aggies.

We were attempting to represent the story as a logic program which allows us to conclude that "normally, Aggies will lose against Boise".

This story presents us with many facts:

$$\begin{array}{l} win(aggies, miners). \\ key\_member(william, aggies). \\ member(dubois, aggies). \\ suspended(william). \\ suspended(dubois). \\ \\ better\_record(boise, aggies). \\ better\_record(boise, miners). \\ \\ team(boise). \\ team(aggies). \\ team(miners). \end{array}$$

The fact that a team has better record often receives a better ranking is stated in the following rule

$$rank\_better(X,Y) \leftarrow better\_record(X,Y), not\ ab\_ranking(X,Y).$$

To predict that a team would lose, we rely on the following piece of commonsense knowledge

> A better ranked team would normally win against a lower ranked one.

A stronger version of this would be

> A better ranked in full strength team would normally win against those lower ranked ones which are not in full strength.

Furthermore,

> A team with injuries or suspensions of key members will not be in full strength.

is some information that we can also used. We encode this as follows

$$\begin{array}{l} not\_full_strength(X) \leftarrow key\_member(Y,X), suspended(Y). \\ not\_full\_strength(X) \leftarrow key\_member(Y,X), injured(Y). \\ \\ full\_strength(X) \leftarrow team(X), not\ not\_full\_strength(X). \end{array}$$

Finally, we add the rule

$$win(X,Y) \leftarrow rank\_better(X,Y), full\_strength(X), not\_full\_strength(Y), not\ ab(X,Y).$$