KNOWLEDGE REPRESENTATION, REASONING AND DECLARATIVE PROBLEM SOLVING

By Jicheng Zhao
based on the notes of others

Class 12: Reasoning about Actions

Introduction to Reasoning about Actions

Imagine we want to reason about a dynamic world where there are actions that change the state of the world.
Reasoning about this world will require us to use all of the reasoning and problem solving paradigms we have learned so far.
There are different kinds of reasoning:

- Prediction
- Hypothetical reasoning
- Counterfactual reasoning
- Explanationary reasoning
- Planning from the initial state
- Planning from the current state
- Diagnosis

In the following, we consider a domain that contains the following actions:

- action *load* causes *loaded*
- action *shoot* causes $\neg alive$ if *loaded*

Given different reasoning problems, we want to present them with AnsProlog programs.

Prediction

- Suppose we are given the initial state of the domain:
  - Initially loaded.
  - Initially alive.
- Now the question is: What happens to the turkey after we shoot?

- The desired conclusion is $\neg alive$ after $shoot$. How do we conclude it by reasoning with AnsProlog?

  To write rules for this we need a model to take into account time, or at least some concept of $before$ and $after$.

  We try the following program:

$$
\begin{aligned}
loaded(X+1) &\leftarrow load(X). \\
\neg alive(X+1) &\leftarrow shoot(X), loaded(X). \\
\neg loaded(X+1) &\leftarrow shoot(X). \\
loaded(0). & \\
alive(0). & \\
shoot(0). &
\end{aligned}
$$

- In this program, how do we ensure that things stay the same by default? This is the FRAME PROBLEM, which we will get to later.

- We could try by adding the following rules to the program:

$$\begin{aligned}
loaded(X+1) &\leftarrow loaded(X), \mathbf{not}\ \neg loaded(X+1).\\
\neg loaded(X+1) &\leftarrow \neg loaded(X), \mathbf{not}\ loaded(X+1).\\
alive(X+1) &\leftarrow alive(X), \mathbf{not}\ \neg alive(X+1).\\
\neg alive(X+1) &\leftarrow \neg alive(X), \mathbf{not}\ alive(X+1).
\end{aligned} \tag{1}$$

Hypothetical Reasoning

- Question: What happens to the turkey if we shoot?
- Naively, we could just add the fact $shoot(0)$ to our program, but this is an inadequate solution. We need to formulate some ways to reason about hypothetical situations in AnsProlog, without changing our existing model of the world.

Counterfactual Reasoning

- Question: What would have happened to the turkey if the gun had not been loaded when we shot?
- This is very similar to hypothetical reasoning, although the imperative of not just adding facts to the model here is clearer, as the necessary facts could (would) make the program inconsistent (by introducing contradictions).

  This problem can be regarded as the combination of hypothetical reasoning and prediction.

Explaining Observations

- Problem: We observe the turkey to be $\neg alive$ at time 1. Explain what happened.
- We can write an AnsProlog program to answer this question by defining a constraint (in this case, that the turkey is $\neg alive$ at time 1), the "world model", and an enumeration of the possible actions in the world.

  The enumerative aspect of this solution should remind us of the problem solving paradigm we have already seen.
- Program:
  - Initial State:
    $$loaded(0). \quad alive(0).$$
  - Constraint:
    $$\leftarrow \neg alive(1).$$

– Actions:

$$0\{load(0), shoot(0), pull(0)\}1 \leftarrow .$$

- This program, when added to our program modelling the results of actions in the dynamic world, will give us the result:

$$shoot(0).$$

- Another problem: We observe the turkey to be $\neg alive$ at time 1. What can be said about the gun being loaded at time zero?
- Similarly to the program above, we can answer this question via problem solving by combining enumeration and a constraint:
- Program:

– Initial State:

$$alive(0).$$

– Constraint:

$$\leftarrow \mathbf{not}\ \neg alive(1).$$

– Actions:

$$1\{loaded(0), \neg loaded(0)\}1 \leftarrow .$$

In the previous question we allowed zero or one actions to take place, in this case we specify that in any answer set, exactly one of either $loaded(0)$ or $\neg loaded(0)$ may be true.

- In this case, the result we get is:

$$loaded(0).$$

Planning from the Initial State

- Question: What do I need to do to make the turkey $\neg alive$ at time 1?
- In smodels, we are constrained to looking at plans of up to a certain finite length. Subject to that constraint, we can formulate the solution as follows:

  Program:

  - Initial State:

$$\neg loaded(0). \quad alive(0).$$

– Planning Constraints:(in this case, we are looking at plans of up to 5 steps in length)

$$\begin{aligned} goal &\leftarrow \neg alive(5). \\ \neg\textbf{not}\ goal. & \\ 0\{load(X), shoot(X)\}1 &\leftarrow time(X). \\ time(0). \cdots time(5). & \end{aligned} \tag{2}$$

- This program, when added to our program modelling the results of actions in the dynamic world, will give us answer sets which are sequences of actions that satisfy the constraints of the goal.

Planning from the Current State

This is significantly different and more complicated than planning from the initial state, as we must take into account the possibility that the plan we formulate initially will no longer be the best plan (or a possible plan) at some later point in time.

Diagnostics

Diagnostics involves reasoning about abnormal situations to find the cause(s) of a failure.
The typical implementation would be to specify the possibility of any given object or process being abnormal, for instance with the arity one predicate $ab(X)$. e.g:

$$ab(switch).$$
$$ab(light).$$

when we observed that the light in the room is not on.
Rules about the system modelled are then written contingent upon the involved processes and objects not being abnormal.
The idea then is to assume initially that nothing is abnormal, and in the case of failure to determine the minimal set(s) of abnormal things which could account for it.