CASL Single Lecture Notes

SLN - AIB - 04

2006-12-05

Cognitive Autonomous Systems Laboratory
Department of Computer and Information Science
Linköping university, Sweden

Erik Sandewall

Logic of Actions and Change

Subject Area: Artificial Intelligence Basic

Date of lecture: 2006-11-22–23

"Single Lecture Notes" are notes corresponding to one lecture.

Related information can be obtained via the following WWW pages:

Linköping university:	http://www.liu.se/
This course:	http://www.ida.liu.se/~TDDA23/
CASL Group:	http://www.ida.liu.se/ext/casl/
The author:	http://www.ida.liu.se/~erisa/

1 Introduction

In lecture 1 of this sequence of lectures, we showed computational methods for planning based on a simple definition of actions as pairs of partial states. Then in lecture 2 we described the beginnings of an approach where more expressive and flexible views of actions could be accomodated by viewing available information as *restrictions* on possible *robotic histories*. One starts from the domain of all kinds of robotic histories for a given set of fluents, and restricts it in two ways. Each given piece of information, such as the effect laws for actions, is only compatible with some of those robotic histories and rejects the others. Secondly, the use of preference relations on robotic histories makes it possible to further reduce the number of accepted histories.

In lecture 3, then, we showed how formal logic can be viewed in two ways: in terms of proofs, and in terms of models. The model-oriented approach is basically the same as the restriction-oriented view of the information about robotic histories. In addition, the two views of formal logic can be proven to be equivalent for the case of first-order predicate logic (and therefore for propositional logic which is a special case).

The natural strategy, then, is to take the restriction-oriented view of actions and robotic histories, express it in terms of logic, apply the preference-relation idea to the models in logic, and thereby obtain a mechanism for drawing conclusions from information about actions and change in various applications. The present lecture centers around that strategy.

2 FOL for actions with explicit time

This section will introduce what is with minor variations the standard notation when phenomena of actions and change is expressed in first-order predicate calculus with explicit time. Another approach, the situation calculus, will be described in the next lecture.

2.1 An example

We begin with an example, namely a variant of the Yale shooting scenario from lecture 2, but now expressed in logic. To begin with we modify the example by assuming that 'load' takes place from time 0 to time 1, and 'fire' from time 1 to time 2, in order to avoid the need for imposing the preference relation on models. Each axiom will be preceded by a number that is later used for reference when conclusions are made.

```
1     H(0,alive)
2     -H(0,loaded)
3     D(t,t+1,load) -> H(t+1,loaded)
4     D(t,t+1,fire) & H(t,loaded) ->
          -H(t+1,loaded) & -H(t+1,alive)
5     D(0,1,load)
6     D(1,2,fire)
```

Axioms 1 and 2 describe the initial state, axioms 3 and 4 express the action laws, and axioms 5 and 6 specify when the actions are performed.

In order to draw conclusions, in particular to draw the conclusion that the turkey is dead at time 2, we use the resolution method. To that end we rewrite all axioms in clause form. Axiom 4 has to be split into two separate clauses. We obtain:

```
 1     H(0,alive)
 2     -H(0,loaded)
 3     -D(t,t+1,load) v H(t+1,loaded)
 4a    -D(t,t+1,fire) v -H(t,loaded) v -H(t+1,loaded)
 4b    -D(t,t+1,fire) v -H(t,loaded) v -H(t+1,alive)
 5     D(0,1,load)
 6     D(1,2,fire)
```

From these axioms we can start drawing conclusions as follows. Each conclusion is obtained by resolving two previous clauses, and their numbers are indicated in a third column to the right. Sometimes it is also necessary to perform unification on those clauses, that is, to impose instantiation on one or both of them so that a particular literal matches, and to simplify the resulting term.

```
 7     H(1,loaded)                          5,3
 8     -H(1,loaded) v -H(2,loaded)          4a,6
 9     -H(1,loaded) v -H(2,alive)           4b,6
10     -H(2,alive)                          7,9
```

Line 8 is unnecessary but was included here to show an additional possible conclusion. Line 7 is obtained by instantiating clause 3 with `t=0`, obtaining

```
       -D(0,0+1,load) v H(0+1,loaded)
```

and simplifying `0+1` to `1`. Technically speaking this is an extension to unification as such, but it is a natural and unproblematic extension.

2.2 The formal language

In general terms, the following formalism is used. We use the following types:

```
timepoints, represented as integers > 0
fluents, which are 'loaded' and 'alive' in this example
events, which are the actions 'load' and 'fire' in this example
```

We use a predicate `H` of two arguments, where the first argument is a timepoint and the second argument is a fluent. Fluents are supposed to have the value `T` or `F` at each timepoint. (In applications where fluents can have more than two values, one uses instead an `H` predicate with three arguments, and introduces additional types for each range of fluent values). The proposition `H(t,f)` expresses that the fluent `f` has the value `T` at time `t`.

In addition, we use a predicate `D` of three arguments, where the first two arguments are timepoints and the third argument is an event, including in partiuclar an action. The proposition `D(t,u,a)` expresses that the action `a` starts at time `t` and ends at time `u`. It can only be true if `u` is strictly greater than `t`.

The use of multiple types for the terms in predicate logic is a complication from a formal point of view, but we shall not delve into that for the present course. From an operational point of view it does not introduce any problems (in the particular case of the predicates and types being used here, at least).