CASL Single Lecture Notes

SLN - AIB - 02

2006-12-05

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

Erik Sandewall

Ontology of Actions

Subject Area: Artificial Intelligence Basic

Date of lecture: 2006-11-08

"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 Basic action-related concepts

A robotic system that uses artificial intelligence techniques needs to deal with descriptions of the robot's environment and its own behavior using both continuous and discrete state variables. Computational methods for classical action planning are restricted to simple forms of systems with discrete-valued state variables. It is important to relate them to more general ways of describing the robot and its 'world', for two reasons: in order to prepare the way for more general approaches to planning and similar computations, but also in order to understand precisely how the structures used in classical planning relate to the actual robotic world. This is the topic of the present memo.

1.1 State variables, or fluents

In the case of a robot UAV, qualitative state variables on the sensor and actuator level include e.g. whether the robot's engine is running or not, and whether it is standing on the ground or not. Quantitative state variables include its altitude, velocity, and direction of flight. Higher-level state variables which are important on the cognitive (AI) level may include the indication whether the present mission is operating correctly relative to the current plan, the number of ground vehicles that are within the field of vision of the UAV (by its video camera), and the type of each of them (car, bus, truck, etc).

Much of the work in AI, and in particular in knowledge recognition, has only taken qualitative state variables into account, and has assumed that all quantitative information can be handled on a 'lower' level in the architecture. However the validity of this separation of levels has also often been questioned, and in the present lectures we shall start with the 'hybrid' framework of both discrete and continuous state variables, and then consider the restriction to discrete-only as an important special case.

Consider then a time axis, represented by a time variable t, and a number of quantities that are defined as functions of t. A function of t is usually called a *signal* in control engineering. In AI it is often called a *fluent*, recycling a term that was originally used by Newton. Here I shall write 'state variable' and 'fluent' alternatingly.

Formally, then, a *robotic history* consists of the starting point (usually 0), then endpoint (T or infinity), and a finite number of fluents over that interval.

It is rarely meaningful to consider the history of the robot system for the entire time axis form minus infinity to infinity. In practice it is considered from time 0 to some upper bound T. The choice of T usually comes out of the description of the application in a natural way, for example, after the end of the last known action there.

In practical situations, 'continuous' state variables are usually not entirely continuous but only *piecewise continuous*, that is, at specific points in time they have discontinuities. Henceforth when we write 'continuous' we mean piecewise continuous unless 'truly continuous' is specified.

For example, if fuel supply to a helicopter's engine varies truly continuously while the motor is running, then it is fair to describe its vertical and horizontal velocities and accelerations are truly continuous as well during the flights. However, the points where the helicopter takes off from the ground and the point where it lands again are probably best modelled in terms of discontinuities for its vertical velocity or acceleration.

State variables that can only take a finite number of values, for example the engine being either 'running' or 'not running' have a simpler situation: they can only change their value at specific points in time but not vary continuously, so they are *piecewise constant.* It is natural to consider piecewise constant as a special case of piecewise continuous, and we shall adopt that convention here. It turns out that several of the methods and results that have first been developed for the discrete-only case can generalize in a nice way to the case of piecewise continuous state variables by replacing 'change of value' by 'has a discontinuity'.

[Ball in the shaft example was shown in the lecture, omitted here for the time being].

1.2 Situations

Suppose a set S of state variables for a robot is given, and S' is a subset of S. We define a *situation* for S' as a period of time during which none of the members of S' has a discontinuity or change of discrete value. To be precise, 'during which' is taken to mean that none of them has a discontinuity (etc) inside the interval, but they may have so at the interval's endpoints.

The entire history of the system from time 0 to time T can therefore be decomposed into a sequence of situations. It is convenient (but not necessary) to assume that situations are maximal, that is, that there is some discontinuity at the starting point and the ending point of each situation interval, besides when situation intervals are bounded by time 0 and time T.

For the mathematically inclined, we should add that we assume of the fluents that they have at most a finite number of discontinuities in the entire history of the robot. This, together with the requirement that the number of fluents is finite, means that the situation structure is Zenon-free.

It follows at once that one can work with different sets of state variables and obtain different partitioning of the time axis into situations. For example, if we start with all the state variables that are used directly by the robot's sensors and actuators, we obtain a certain sequence of situations. Let f be a continuous-valued state variable in that set. Then add one more state variable, g, that has only a few discrete values representing whether the value of f is 'big', 'small', 'very small', etc, and define the value of g in terms of some thresholds on f that it can pass continuously. Then the addition of g will add more discontinuities to the history of the system, so some situations will be broken down into several parts. On the other hand if some state variables are removed from the set then several successive situations may coalesce into a single one as a consequence.

1.3 Events and Actions

We define an *event recognizer* as a function that takes a robotic history over a time interval $[0, \mathrm{T}]$ and that has as value a set of *events* that are pairwise disjoint intervals within $[0, \mathrm{T}]$. Events being disjoint means that the endpoint of one event may be the starting-point of another one, but two events may not have more than one point (an endpoint) in common.

For example, one event recognizer may identify periods of time during which a helicopter first starts its engine, then takes off, flies, lands at another point, and turns off its engine. This recognizer will recognize events that we would characterize as 'the helicopter flies to a point P and lands there'. If there are several helicopters in the world then they must have one recognizer each for this type of event, since they may fly at the same time.

An *event* is a period that is recognized by a particular event recognizer.

An *activity* is an event that is performed under active control by an agent, where the agent may be either a robot, a person, or another animate (for example an animal). This means that the agent can decide when the activity is to start and when it is to end. For example, a person that jumps into a swimming-pool and stays there for as long as he pleases and then decides to get up, performs an activity in this sense.

An *action* is an activity where there is a well-defined condition for when the activity ends. For example, if an agent has an activity for 'drink a glass of beer' that is defined in terms of successive behaviors (drink one gulp, wait a little, drink one more, etc) and where the activity ends when the glass turns out to be empty, then that activity is in particular an action.

The distinction between actions and other activities is not absolute; it is rather a question of how the designer of a reasoning system wishes to model things. The point about actions is that they can be understood as a kind of program that has well-defined preconditions (state of the world when the action starts) and postconditions (state of the world when the action ends), and that one can write a program or set of rules that describes or even simulates how the agent will perform the action. Actions are therefore fundamental for systems that do reasoning in or about robots.

The starting times and ending times of activities, including actions, are considered as points of discontinuity so that it is necessary to have a change of situation at those timepoints.

2 Preconditions and postconditions

In the previous lecture (lecture number 1) we defined an action as a pair of partial states over the same state variables. That action concept is much more restrictive than the action concept from the previous section here. However, in order to relate the two perspectives, let us see what restrictions must be imposed on actions as defined here, or what generalizations are needed in the previous lecture, in order to obtain a direct correspondence between the two action concepts.

First of all we generalize the concept of 'state' and 'partial state' from last lecture so that state variables can range over more than two values, but still only a finite number of values. For example, one can have a state variable for the color of a building, with values such as 'white', 'brown', and 'red'. Combining it with a few other, similar state variables, a state can be for example:

```
{color:white, roofcolor:black, floors:4, busstop-outside:T}
```

Then we proceed to the restrictions on the action concept of section 1 here, as follows:

- Time is discrete. It may be represented by the non-negative integers, i.e. the integers from 0 and up.
- All state variables must be discrete-valued. Their values must be taken from a finite set of possible values.
- In the description of actions, only the state of the world at their starting-time and their ending-time is of interest. The value of state variables in the interior points of the action interval is not of interest.
- Only one action can be performed at a time. No concurrent actions.
- The only changes that occur in the world are those that are the results of actions. Apart from them, state variables retain constant value.
- An action is characterized by only one pair of precondition and effect, both of which are partial states. It is not allowed to have actions with conditional effects, where the postcondition depends on the precondition. (Such an action would have to be represented as a *set of* pairs of partial states).

The effects of actions is then defined as follows: if the action $\langle P, E\rangle$ is performed in a state S where all the conditions in P apply, i.e. a state that has P as a subset, then the result state shall be $(S - P) \cup E$. If it is performed in any other state then the state is unchanged and the action has no effect at all. Therefore there is a certain 'conditional effect' between the action being applicable or not, but that is all. There can not be several different outcomes each containing a change.

These are of course very strong restrictions, but one has to start somewhere, and it is only with these restrictions that one can use the methods from Lecture 1. Once one realizes the restrictions and makes them precise, one can also proceed to removing restrictions and generalizing the concepts.

3 Actions as restrictions on robotic histories

With the restrictions of section 2 here, a robotic history becomes a structure that assigns a value to each combination of timepoint and fluent symbol, among the permitted values for that fluent. We shall illustrate robotic histories as tables with one column for each fluent and one line for each timepoint for as far as is needed.

We wish to study robotic histories that correspond to a given, finite set of actions. Since there are no changes in the 'world' besides the effects of actions, it is guaranteed that the robotic history does not change at all after the end of the last action. This defines an appropriate ending-time T of the robotic history.

In order to illustrate some of the problems with reasoning about actions we shall use a number of illustrative examples that are often used in the literature. The first one is called the ’Yale Shooting Problem’.

3.1 The Yale Shooting Example

Scenario: There is a turkey and a person equipped with a gun. The person loads the gun, waits a while, and fires the gun in the direction of the turkey.

Problem: From a common-sense point of view and allowing for a number of additional conditions (aiming right, etc), the turkey dies from the shot. How can a reasoning system draw that conclusion?

Formalization: States with two state variables, namely ’alive’ for whether the turkey is alive or not, and ’loaded’ for whether the gun is loaded. Three actions namely ’load’, ’wait’, and ’fire’ which are defined as follows:

```
load = <{loaded:F}, {loaded:T}>
wait = {0, 0}                  (no effect)
fire = <{loaded:T, alive:T}, {loaded:F, alive:F}>
```

From a common-sense point of view one may complain that the gun shall become unloaded as a result of the ’fire’ action even if the turkey is not alive when the gun is fired, but we ignore that complaint since our concern here is with formal reasoning machinery given a certain definition of actions.

We also assume the following initial state:

```
{loaded:F, alive:T}
```

Finally we assume that the ’load’ action is performed between time 3 and 4, and the ’fire’ action between time 6 and 7. The ’wait’ action is between time 4 and 6, or if one prefers one can say there is no action at those times.

The following is obviously the intended robotic history:

time	alive	loaded
0	T	F
1	T	F
2	T	F
3	T	F
4	T	T
5	T	T
6	T	T
7	F	F
8	F	F

Now let us consider the given action definitions as *restrictions* on the robotic history, saying that we are interested in *all* robotic histories that satisfy the given information: the right initial state at time zero, and the right precondition-postcondition behavior at those times when the actions are known to occur.

The reason why one are interested in that way of looking at things, and not merely view the action rules as a way of ’simulating’ the world from state 0 and up, is that one wishes to gradually add *other kinds of information* about how things change in the world and what the state of the world is: observations at later points in time, concurrent actions (actions at the same time), cause-effect rules, actions of more than one agent, etc. This can best be done if each of those sources of information is allowed to impose its own constraints on what are admissible robotic histories. Before adding those other sources of information, we must make this work for the very simple case of one action at a time, no causality but only direct effects of actions, etc.

The view of given information as restrictions on a set of possible structures (in this case: possible robotic histories) is well in line with how logic works and operates. That will be the topic of the next lecture.

Now it is clear that the history shown above does satisfy the restrictions for initial state and effects of actions. The problem is that there are many other robotic histories that *also* satisfy those restrictions, since the restrictions do not say anything about what happens at intermediate points in time. We shall return to this question in the lecture (and lecture note) about "The Frame Problem".