CS324: AI (Situation Calculus & Reasoning.) **Spring 2015**

Lecture 10: Feb. 11, 2015

Lecturer: K.R. Chowdhary *: Professor of CS (VF)*

Disclaimer: *These notes have not been subjected to the usual scrutiny reserved for formal publications. They may be distributed outside this class only with the permission of the Instructor.*

10.1 Situation Calculus

The concept of *action* arises in at least two major subareas of artificial intelligence, (1) natural language processing and (2) problem solving. The primary goal of situation calculus is to provide a formalism which is expressive enough for representation of actions and to explore its use in defining the meanings of English language sentences that describe actions and events.

The requirement on the formalism for representation of actions is that it should be a useful representation for *action reasoning* (i.e., problem solving). It has a very useful application, i.e., to describe, how this representation could be used for planning (of actions) or for plan recognition system. This is essential to the natural language processing as well, because the natural language understanding is aimed to ultimately result to problem-solving capability.

The situation calculus is also the language for choice for investigations of various technical problems that arise in theorizing about *actions* and their *effects*. It is being taken as a foundation for practical work in planning, control, simulation, database updates, agent programming and robotics. In parallel with these developments of its applications, there have emerged axiomatizations for the situation calculus, and explorations of some of their mathematical and computational properties.

10.1.1 Action, Situation, and Objects

The situation calculus is a logic formalism for representing and reasoning about dynamical domains. The concepts in the situation calculus are *situations*, *actions* and *fluents*. A number of objects are typically involved in the description of the world. The situation calculus is based on a *sorted domain* with three sorts: actions, situations, and objects, where the objects include everything that is not an action or a situation. Fluents are modeled as either predicates or functions.

A situation is a kind of state (*ako*), however it is a result of chain of earlier situations. Actions are what make the dynamic world change from one situation to another when performed by agents. A fluent is a condition that can change over time. Fluents are situation-dependent functions used to describe the effects of actions. There are two kinds of them: *relational fluents* and *functional fluents*. The former have only two values: *True* or *False*, while the latter can take a range of values. For instance, one may have a relational fluent called *handempty* which is true in a situation if the robot's hand is not holding anything. We may need a relation like this in a robot domain. One may also have a functional fluent, e.g., *battery-level*, whose value in a situation is an integer of value between 0 and 100 denoting the total battery power remaining in one's notebook computer.

The set of actions form sort of domain. The action can also use variables, and they can be quantified, for example in a robot world, possible action terms would be $move(x, y)$ to model a robot moving to a new

location (x, y), and $pickup(o)$ to model the robot picking up an object o, and so on.

10.1.2 Formalism

In the situation calculus, a dynamic world is modeled to progress through a series of situations as a result of various actions being performed within this world. A situation is a consequence of sequence of action's occurrences. The situation available before any actions are performed is generally denoted by S_0, called *initial situation*. A new situation resulting from the performance of an action is denoted using the function symbol *result* or *do*. This function symbol has a 'situation'and 'action'as arguments, and a new situation as a result due to performing the given action in the given situation. We shall make use of *do*, called binary function symbol *do*, expressed as,

$$do : action \times situation \rightarrow situation. \tag{10.1}$$

The intended interpretation is that $do(a, s)$ (or $result(a, s)$) denotes the successor situation resulting from performing action a in situation s. Accordingly, the basic formalism of the situation calculus is represented by:

$$s' = do(e, s) \tag{10.2}$$

which asserts that s' is the resulting situation when event e (action) occurs in situation s.

Example 10.1 *Situation-action.*

The fluent $is\_carrying(o, s)$ can be used to indicate that the robot is carrying a particular object 'o'in a particular situation 's'. If the robot initially carries nothing, $is\_carrying(Ball, S_0)$ is false while, the fluent

$$is\_carrying(Ball, do(pickup(Ball), S_0))$$

is true. The location of the robot can be modeled using a functional fluent location, which returns the location coordinates '(x, y)'of the robot in a particular situation. □

To describe a dynamic domain using the situation calculus, one has to decide on the set of actions available for the agents to perform, and the set of fluents needed to describe the changes these actions will have on the world. For example, consider the blocks world where some blocks of equal size can be arranged into some set of towers on a table. The set of actions in this domain depends on what the imaginary agent can do. If we imagine the agent to be a robot-arm that can be directed to grasp any block that is on the top of a tower, and either add it to the top of another tower or put it down on the table to make a new tower, then we can have the following actions:

- $stack(x, y)$ - put block x on block y, provided the robot is holding x, and y's top is clear. Being action, it is read "stack x on y", and shall not be confused with predicate.
- $unstack(x, y)$ - pick up block x from block y, provided the robot's hand is empty, x is on y, and x has top clear.
- $putdown(x)$ - put block x down on the table, provided the robot is holding x.

- *pickup*(x) - pick up block x from the table, provided the robot's hand is empty, x is on the table and top clear.
- *move*(x, y) - move block x to position y.

To describe the effects of these actions, we can use the following relational fluents:

- *handempty* - True in a situation if the robot's hand is empty.
- *holding*(x) - True in a situation if the robot's hand is holding the block x.
- *on*(x, y) - True in a situation if block x is on block y.
- *ontable*(x) - True in a situation if block x is on the table.
- *clear*(x) - True in a situation if block x has top clear.

For action *stack*(x, y) to be performed in a situation, the fluent,

- *holding*(x) must be true, and
- *clear*(y) must be true.

Also, after *stack*(x, y) is performed and it results to a new situation, the fluent,

- *on*(x, y) will be true,
- *handempty* will be true,
- *holding*(x) will be false, and
- *clear*(y) will be false.

The action *stack*(x, y) along with present value of fluents, and resulting action with new value of fluents can be formally expressed as,

$$\begin{aligned}\forall x \forall y[[((holding(x) \wedge clear(y)) \rightarrow (stack(x, y)] \rightarrow on(x, y) \\ \wedge handempty \wedge \neg holding(x) \wedge \neg clear(y))]\end{aligned} \tag{10.3}$$

Now, for example, we can say that for action *stack*(x, y) to be performed in a situation, *holding*(x) and *clear*(y) must be true, and that after *stack*(x, y) is performed, in the resulting new situation, *on*(x, y) and *handempty* both will be True, and *holding*(x) and *clear*(y) will no longer be True.

10.1.3 Formalizing the notions of context

Consider axiomatizing "on" so as to draw appropriate consequences from the information expressed in the sentence, in a world of space-craft on a moon mission:

"The book is on the table."

One may haggle about the precise meaning of 'on'causing difficulties about what can be between the book and the table or about how much gravity there has to be in a spacecraft in order to use the word "on" and whether centrifugal force counts.

There are a some predicates about contexts as well as dealing with time. One of the most important predicate is *holds*, which asserts that a property holds (i.e., is true) during a time interval. Thus, the sentence,

$holds(p, t)$

is true if and only if property p holds during time t. A subsequent axiom will state, this is intended to mean that p holds at every subinterval of t as well. Note that if we had introduced *holds* as a "modal operator" we would not need to introduce properties into our ontology.

Suppose that, whenever a sentence p is present in the memory of a computer, we consider it as in a particular context and as an abbreviation for the sentence,

$holds(p, C)$

where C is the name of a context.

A logical system using contexts might provide operations of *entering* and *leaving* a context yielding what we might call *unnatural deduction* allowing a sequence of reasoning as given in the followings.

$holds(p, C)$
$ENTER\ C$
p
...
...
q
$LEAVE\ C$

This resembles the usual logical natural deduction systems.

Example 10.2 *Represent the following sentence in the formalism of situation calculus.*

"A man called Rex is standing at the gate of Ghana Bird Sanctuary wearing a black T-shirt. Rex loads his toy gun, waits for few seconds, and shoots at a migratory crane."

Solution. The fluents, which are either true or false, in this sentence are:

standing(place): whether Rex is standing at a given place or not?
black: whether Rex is wearing black T-shirt or not?
loaded: whether the gun is loaded or not?

Following are the actions in above sentence:

load: Rex is loading the gun.
wait: Rex waits for few seconds.
shoot: Rex shoots his gun.

Now, let us find out what holds at initial situation S_0.

$holds(standing(gate), S_0)$

$holds(black, S_0)$

$holds(alive, S_0)$

$\neg holds(loaded, S_0)$

Now we try to relate actions with situations. In other words, which fluents will hold after performing a certain action in a given situation?

1. If Rex shoots the gun and gun is loaded, crane is not alive.
2. If Rex shoots the gun, gun will become unloaded.
3. If Rex loads the gun, gun will be loaded.
4. If Rex waits on an loaded gun, the gun remain loaded.
5. If Rex waits on an unloaded gun, the gun remain unloaded.

Our first method for writing above sentences in formal way is as follows. Consider a general sentence "f_2 (fluent) will be true by performing action a_2 in state s if (provided that) f_1 is true in s."

$$\forall s[holds(f_1, s) \rightarrow holds(f_2, do(a_2, s))]$$

Now consider the first sentence: "if Rex shoots the gun and the gun is loaded, crane is not alive," which is written as:

1. $\forall s[holds(loaded, s) \rightarrow \neg holds(alive, do(shoots, s))]$

   The remaining four sentences can be expressed in situation calculus as follows:

2. $\forall s[\neg holds(loaded, do(shoot, s))]$
3. $\forall s[holds(loaded, do(load, s))]$
4. $\forall s[holds(loaded, s) \rightarrow holds(loaded, do(wait, s))]$
5. $\forall s[\neg holds(loaded, s) \rightarrow \neg holds(loaded, do(wait, s))]$

Having represented in this form of FOPL, the reasoning is done using conventional methods used for predicate logic reasoning, e.g., resolution based theorem proving.