32002: AI (Ontological engineering & Situation Calculus) **Spring 2014**

Lecture 19: February 19, 2014

Lecturer: K.R. Chowdhary *: Professor of CS*

19.1 Ontological Engineering

The *knowledge engineering* is process of knowledge representation; identification of task, assemble relevant knowledge, decide vocabulary of predicates, functions and constructs, encode knowledge about domain, encode description of specific problem instance, pose queries to procedures and get answers and debug knowledge-base.

The ontological engineering can be defined on the same line of knowledge engineering. It is related to the creating representation of general concepts - actions, time, physical objects,a and beliefs.

Ontological engineering comprise a set of activities conducted during conceptualization, design, implementation and deployment of ontologies. Ontological engineering covers topics including philosophy, metaphysics, knowledge representation formalisms, development methodology, knowledge sharing and reuse, knowledge management, business process modeling, common-sense knowledge, systematization of domain knowledge, information retrieval from the Internet, standardization, and evaluation. It also gives us design rationale of a knowledge base, helps define the essential concepts of the world of interest, allows for a more disciplined design of a knowledge base, and enables knowledge accumulation. In practice, knowledge of these disciplines helps:

- Organize the knowledge acquisition process;
- Specify the ontology's primary objective, purpose, granularity, and scope; and
- Build its initial vocabulary and organize taxonomy in an informal or semi-formal way, possibly using an intermediate representation

19.1.1 Categories and Objects

Organization of objects into categories is important in knowledge organization. Of course interactions take place as individuals but relations are at the level of categories. For example, in the sentence "Basketball team has won", does not refer to a single ball and not a single player, but a relation among team as a group.

The Category also helps in prediction of objects once they are classified. One refers to objects from perceptual inputs, infers category of membership from perceived properties of the objects, then use category information to predict about object. For example, from its yellow colour, size, shape, we identify that it is a 'Mango'.

In FOPL there are two choices for categories: *predicates* and *objects*. Following are examples:

$basketball(b);$

or $member(b, basketballs)$;

or $b \in basketballs$; (the object b is a member of category basketball)

$subset(basketballs, balls)$; (a category is a subclass of another category)

or $basketballs \subset balls$.

The category is defined by *members* and *subset* relation. We also note that categories organize the knowledge as well as they help to inherit. Thus, 'mango' $\in$ 'mangoes', and 'mangoes' $\subset$ 'fruits'. Thus, mango inherits the properties of 'taste' from fruits (fruits have 'tastes'). Similarly, we have categories: animals, birds, foods, institutions. A "set of institutions" $\subset$ "institutions", and MBM $\in$ 'institutions'. The subclass organize the categories into a *taxonomic* hierarchy. The taxonomical hierarchies are common in government, military, and other organizations.

Following are representations of taxonomies:

$x \in basketballs \Rightarrow round(x)$; (all members of of a category have same property)

$color(x, brown) \wedge round(x) \wedge dia(x) = 9" \wedge x \in balls \Rightarrow x \in basketballs$; (the member category can be recognized by some property)

$dogs \in domesticatedanimals$. (categories are members)

19.1.2 Physical Decomposition of Categories

An object can be part of another, for example, nose, eyes, hands, are part of body; steering part of car, wheels are part of wheel-assembly, and wheel-assembly is part of car. This can be represented by relations of physical decompositions as follows.

$partof(wheel, wheelassembly)$.

$partof(wheelassembly, car)$.

Thus taxonomies are *transitive relations*.

$$partof(x, y) \wedge partof(y, z) \Rightarrow partof(x, z).$$

Also, there is reflexive relation, $partof(x, x)$, as since, x has one part, and that is itself. The categories of composite objects is defined by structural relations between parts and assemblies.

19.1.3 Measurements

The object have height, weights, mass, length, and other physical measurements. The values need to be assigned to the objects in the form of their properties. Here are the examples.

$length(ipad) = inches(5) \Rightarrow centimeters(12.5)$.

$listprice(basketball) = rupee(500)$.

$height(x) > height(y) \Rightarrow taller(x, y)$.

19.2 Situation Calculus

The concept of action arises in at least two major subareas of artificial intelligence, namely, natural language processing and problem solving. For the most part, the formalisms that have been suggested in each sub-area are independent of each other and difficult to compare. However, there is presently no computational theory of action that is sufficiently powerful to capture the range of the meanings and distinctions expressible in English. The primary goal of situation calculus is to provide a formalism that is considerably more expressive than current theories of action and to explore its use in defining the meanings of English sentences that describe actions and events.

The requirement on the formalism is that it should be a useful representation for action reasoning (i.e., problem solving). It has application describing how the representation could be used by a planning or plan recognition system. This is essential to the natural language research as well, because problem-solving techniques are being used more and more in our models of language comprehension.

19.2.1 Action, Situation, and Objects

The situation calculus is a logic formalism designed for representing and reasoning about dynamical domains. The basic concepts in the situation calculus are *situations*, *actions* and *fluents*. 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. Actions, situations, and objects are elements of the domain, the fluents are modeled as either predicates or functions. A situation is a kind of state (ako), however a situation is a result of chain of earlier situations.

Briefly, 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 called *battery-level* whose value in a situation is an integer between 0 and 100 denoting the total battery power remaining on one's laptop computer.

The actions form a sort of the domain. Variables of sort action can be used. Actions can be quantified. A special predicate *Poss* is used to indicate when an action is executable. For example in a robot world, possible action terms would be $move(x, y)$ to model the robot moving to a new location (x, y), and '$pickup(o)$' to model the robot picking up an object o.

19.2.2 Formalism

In the situation calculus, a dynamic world is modeled as progressing through a series of situations as a result of various actions being performed within the world. A situation represents a history of action occurrences. The situation before any actions have been performed is typically denoted S_0 and called the initial situation. The 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 an action as arguments, and a situation as a result, the latter being the situation that results from performing the given action in the given situation.

Two function symbols of sort situation are:

1. A constant symbol S_0 , denoting the initial situation.
2. A binary function symbol

$$do : action \times situation \rightarrow situation \tag{19.1}$$

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

$$s' = do(e, s) \tag{19.2}$$

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

The basic ontology of situation calculus consists of 1) situations, which corresponds to snapshots of universe or an instant of time, and 2) actions or events, which change the world from one state to another. It is a sorted language with sorting (order) for situation and actions.

References

[1] Chowdhary K.R. (2020) Logic and Reasoning Patterns. In: Fundamentals of Artificial Intelligence. Springer, New Delhi. `https://doi.org/10.1007/978-81-322-3972-7_6`