Part I
Knowledge engineering, Situation Calculus

1 Knowledge engineering

1.1 Ontology

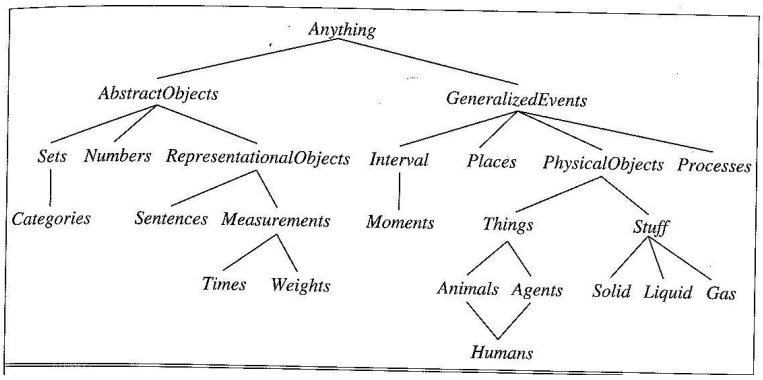
Ontological engineering

Definition 1. *Ontological engineering*: process for the representation of abstract concepts

Abstract concepts:

- Actions, Time, Objects, Knowledge (beliefs, possibilities...)

Upper Ontology



Categories and Objects

Definition 2. An *object* of the world always belongs to a *category*.

- organisation and simplification of a knowledge base
- use of *inheritance* (similar to object programming)
- *taxonomy*, *taxonomy hierarchy*

To make reasoning about the objects of the world, we generally make reasoning about the corresponding categories.

Categories and Objects: examples

Example 3.

- An object is a member of a category

$$BB_9 \in BasketBalls$$

- A category can be a subclass of another category

$$BasketBalls \subset Balls$$

- All members of a category have some properties

$$\forall x \; x \in BasketBalls \Rightarrow Round(x)$$

- Members of a category can be recognised by some properties

$$Orange(x) \wedge Round(x) \wedge Diameter(x) = 9.5'' \wedge x \in Balls \Rightarrow x \in BasketBalls$$

- A category as a whole has some properties

$$Dogs \in DomesticatedSpecies$$

Composite objects

To express that we can use terms like $PartOf$ to say that one thing is part of another

- $PartOf(Bucharest, Romania)$
- $PartOf(Romania, EasternEurope)$
- $PartOf(EasternEurope, Europe)$
- $PartOf(Europe, Earth)$

We only need to express the *transitivity* of the $PartOf$ relation:

$$PartOf(x, y) \wedge PartOf(y, z) \Rightarrow PartOf(x, z)$$

$$Part(x, x)$$

Composite objects

Example 4. What is a biped?

$Biped(a) \Rightarrow$

$$\exists l_1, l_2, b \; Leg(l_1) \wedge Leg(l_2) \wedge Body(b) \wedge$$
$$PartOf(l_1, a) \wedge PartOf(l_2, a) \wedge PartOf(b, a) \wedge$$
$$Attached(l_1, b) \wedge Attached(l_2, b) \wedge$$
$$l_1 \neq l_2 \wedge [\forall l_3 Leg(l_3) \wedge PartOf(l_3, a) \Rightarrow (l_3 = l_1 \vee l_3 = l_2)]$$

Organising and reasoning with categories

How to organise categories. We need a description language. There are two families:

1. *Semantic networks*
   - a graphical aids for visualising a KB
   - efficient algorithms for inferring properties of an object on the basis of its category membership
2. *Description logic*
   - Formal language for constructing and combining category definition
   - efficient algorithms for subset/uperset category relationships detection

Semantic network

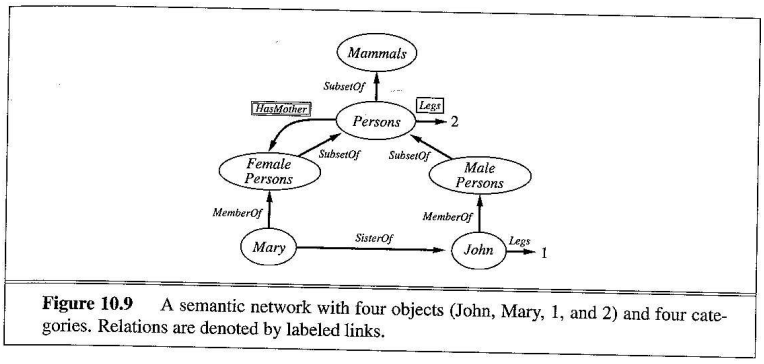


Figure 10.9 A semantic network with four objects (John, Mary, 1, and 2) and four categories. Relations are denoted by labeled links.

Example 5.
Links: $SisterOf(Mary, John)\ Mary \in FemalePersons\ FemalePersons \subset Persons...$
Single-boxed: $\forall x\ x \in Persons \Rightarrow Legs(x, 2)$
Double-boxed: $\forall x\ x \in Persons \Rightarrow [\forall y\ HasMother(x, y) \Rightarrow y \in FemalePersons]$

Description logics

Definition 6. *Description logics* are notations that are designed to make it easier to describe definitions and properties of categories.

Example 7. "Bachelors are unmarried adult males"

$$Bachelor = And(Unmarried, Adult, Male)$$

which is a "syntaxical sugar" for

$$\forall x\ Bachelor(x) \Leftrightarrow Unmarried(x) \wedge Adult(x) \wedge Male(x)$$

Summary

- To represent aspects of complex worlds, *ontology engineering*
- The world is a set of *objects* that belong to *categories*
- Categories are inherited from other categories: *Taxonomy*
- The description of the world is based on *compositionality* of objects

- The result of the description in a formal language (like FOL) is a *Knowledge base*
- The knowledge base contains
  - *Facts*
  - *Axioms* that allow to make reasoning about the facts in order to get other facts (by inference)

2 Situation Calculus

2.1 Situation calculus: an introduction

Situation calculus: an introduction

Previously, we saw FOL as a way to:

- Represent *static* worlds
- Prove *atemporal* sentences

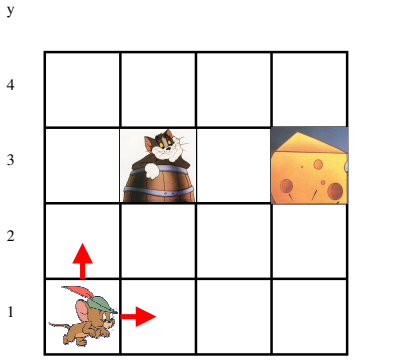
Nothing is *dynamical*

The language of *Situation calculus* is specifically designed for representing *dynamically* changing worlds.

- a dynamical world is a sequence of *situations*
- a situation is a static world
- all changes to the world are the result of *actions*

2.2 An example

Mouse, Cat and ... cheese



Example 8.

The figure presents the *initial situation*. If Jerry moves from room 1,1 to room 1,2 the world is changed and we have a new situation. How to express that? This is the purpose of Situation Calculus.

2.3 Ontology of situation calculus

Situation

Definition 9. A *Situation* is a logical term:

1. an initial term S_0 called the *initial situation*;
2. any term generated by applying an *action* to a situation:

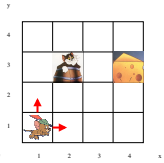
   $$Result(a, s)$$

   or

   $$Do(a, s)$$

   where a is an action and s is a situation.

Situation: examples

Example 10. S_0 represents 

A new situation could be:

$$Result(Go(Room(1,1), Room(1,2)), S_0)$$

If Jerry is in room (1,1) and goes to room (1,2) then we have a new situation. Then, we can another situation:

$$Result(Go(Room(1,2), Room(2,2)), Result(Go(Room(1,1), Room(1,2)), S_0))$$

Fluents

Definition 11. A *Fluent* is a function or a predicate that vary from one situation to the next. By convention, the situation is always the last argument of the fluent.

$$function(a, b, c, S)$$

or

$$predicate(a, b, c, S)$$

where a, b, c are terms and S is a situation.

Example 12.

- Jerry holds some cheese in the situation s (predicate)

$$Holding(Cheese(c), s)$$

- Here is the age of Tom in the situation s (function)

$$AgeOf(Tom, s)$$

Atemporal predicates/functions

Definition 13. An *atemporal predicate/function* is a predicate/function that do not vary from one situation to the next.

$$function(a, b, c)$$

or

$$predicate(a, b, c)$$

where a, b, c are terms (not situations).

Example 14.

- Tom is a cat (predicate)

$$Cat(Tom)$$

- The left leg of Jerry (function)

$$LeftlegOf(Jerry)$$

Actions in situation calculus

Definition 15. An *action* is described with two axioms:

1. *Possibility axiom*: when it is possible to execute the action
2. *Effect axiom*: what happens when the action is executed

Definition 16. *Possibility axiom*

$$Preconditions \Rightarrow Poss(a, s)$$

where a is the action and s the situation.

Definition 17. *Effect axiom*

$$Poss(a, s) \Rightarrow \text{Changes that result from taking action } a \text{ in situation } s$$

where a is the action and s the situation. The changes are expressed with fluents.

Possibility axioms: examples

Example 18. Our agent: Jerry.

- Jerry can go between adjacent locations

$$At(Jerry, x, s) \wedge Adjacent(x, y) \Rightarrow Poss(Go(x, y), s)$$

- Jerry can grab some cheese in the current location

$$Cheese(c) \wedge At(Jerry, x, s) \wedge At(c, x, s) \Rightarrow Poss(Grab(c), s)$$

- Jerry can release some cheese that it is holding

$$Holding(c, s) \Rightarrow Poss(Release(c), s)$$

Effect axioms: examples

Example 19.

- Jerry goes between adjacent locations

$$Poss(Go(x, y), s) \Rightarrow At(Jerry, y, Result(Go(x, y), s))$$

- Jerry grabs some cheese in the current location

$$Poss(Grab(c), s) \Rightarrow Holding(c, Result(Grab(c), s))$$

- Jerry releases some cheese that it is holding

$$Poss(Release(c), s) \Rightarrow \neg Holding(c, Result(Release(c), s))$$

And now? What happens?

We have an ontology (symbols, set of axioms) based on a formal system (FOL), let make some reasoning about situations!

Definition 20. *Projection/prediction* task: an agent (like Jerry) should be able to *deduce* the outcome of a given sequence of actions.

Example 21.

- "going to room (2,3) is not a good idea"
- "going to room(1,2) is not dangerous"
- "going to room (4,3) is an excellent idea"

And now? What happens?

Definition 22. *Planning* task: an agent should be able to *find* a sequence that achieves a desired effect

Example 23. **Desired effect**: "I want to get some cheese and Tom must not see me"

A solution: "Go to room (2,1) then Go to room (3,1) then Go to room (4,1) hen Go to room (4,2) then Go to room (4,3) then grab the cheese then Go to room (4,2) then Go to room (4,1) then Go to room (3,1) then Go to room (2,1) then Go to room (1,1) then release the cheese"

Planning: example

Example 24. **Initial situation** $S_0 =$  . **Goal**: "Jerry wants the cheese in 1,1" In the initial situation, we must have:

$At(Jerry, Room(1,1), S_0) \wedge At(Gruyere, Room(1,2), S_0)$ $Cheese(Gruyere) \wedge Adjacent(Room(1,1), Room(1,2)) \wedge Adjacent(Room(2,1), Room(1,1))$ (+ other things...)

The goal is also a sentence: $\exists seq\ At(Gruyere, Room(1,1), Result(seq, S_0))$

If the sentence is entailed by KB then we prove there is a plan for Jerry! If the inference algorithm is *constructive* then seq will be substituted with a term representing one plan!

Planning: example

Example 25. From S_0, we can get: (the action is $Go(Room(1,1), Room(1,2))$)

$$At(Jerry, Room(1,2), Result(Go(Room(1,1), Room(1,2)), S_0)$$

From this new sentence and KB we should be able to prove:

$$At(Gruyere, Room(1,2), Result(Go(Room(1,1), Room(1,2)), S_0)$$

... but we can't! (no axioms from KB can infer that)

Effect axioms say what changes but don't say what stays the same. The fact that Jerry moves from (1,1) to (1,2) does not change the fact that the cheese is *still* in (1.2). Something is missing: *frame problem*

2.4 Frame problem

Frame problem

Definition 26. Purpose: representing all the things that stay the same. We need *frame axioms* that do say what stay the same.

The solution for solving the frame problem must be *efficient*.

- Frame axioms are numerous (an action generally changes a few fluents)
- One solution is to use *successor-state axioms* (not complete)
  - It solves the *representational frame problem*

Representational frame problem

Definition 27. Instead of writing effect axioms,

$$Poss(a, s) \Rightarrow \text{Changes that result from taking action } a \text{ in } s$$

we use *successor-state axioms*:

$Poss(a, s) \Rightarrow$

Fluent is true in situation *s* $\Leftrightarrow$

[The effect of action *a* made it true $\vee$

It was true before and action *a* left it alone]

Representational frame problem: example

Example 28. $Poss(a, s) \Rightarrow$

$$[At(Jerry, y, Result(a, s)) \Leftrightarrow [a = Go(x, y) \vee (At(Jerry, y, s) \wedge a \neq Go(y, z))]]$$

Informally: if a is possible in situation s then Jerry is in y in the "next situation" if either the action a is "go to y" or Jerry was "already" in y and a is not a movement action from y.

$Poss(a, s) \Rightarrow$

$$[Holding(c, Result(a, s)) \Rightarrow [a = Grab(c) \vee (Holding(c, s) \wedge a \neq Release(c))]]$$

Informally: if a is possible in situation s then Jerry holds the cheese in the "next situation" if either the action a is "Grab the cheese" or Jerry was already holding the cheese and the current action a is not to release it.

Summary

- Situation calculus: a way to represent actions, events in the world and make reasoning
  - Initial situation + axioms for actions
- Use of inference algorithms to do *Deductive planning*
- *Frame Problem*:
  - Use of *frame axioms* to express what the actions don't change in the world.