Knowledge Representation

Knowledge Engineering Course

Andrea Bonarini

Department of Electronics - Politecnico di Milano

http://www.dei.polimi.it/people/bonarini

Academic Year 2010-2011

Let's start from history...

- 1956 - Birth of Artificial Intelligence at the Dartmouth College Meeting among phylosophers, psychologists, engineers, computer people
- Sixties - Big problems faced: General Problem Solver, Natural Language Interaction, ...
- Seventies - First expert systems and real applications
- Eighties - Success and delusions
- Nineties - Identification of application niches for AI techniques

- XXI Century - Agents. AI applications embedded in ICT

What is a Model?

What is a Model?

A representation of something, designed with a purpose

- it contains the information needed to achieve the goal
- it is usually compact
- it is different from the modeled "thing"

What is "Knowledge"?

What is "Knowledge"?

- data structures
- relationships
- possibility to reason to find new knowledge

What is “Knowledge Engineering?”

A set of techniques (methodology)
to design and implement Knowledge-Based Systems (KBS)

What is "What are KBSs for?"

Many applications:

- Games
- Diagnosis
- Planning
- Control
- Classification
- Machine Learning
- Perception
- Human-Machine Interaction

What will we do?

- Identification of knowledge characteristics and elements
- Symbolic knowledge representation
- Knowledge acquisition
- KBS design

And now, a game...

What is a G00237?

You are like a KBS that has to collect information from the external world to understand what is something that at the moment is just named by an internal name: G00237

I'm the external world and I can only answer with "Yes" or "No" or the only specific values that you might understand as a KBS

What are the knowledge elements?”

Here are some:

- Classes
- Instances
- Properties
- Relationships and hierarchies
- Inference
- Truth and uncertainty

Design levels

Epistemological aspects

Classes, Instances, Properties, Inference, Uncertanty, ...

Representation techniques

Objects, Logic, Frames, Semantic Networks, Rules, ...

Languages

Prolog, LISP, OPS5, CLIPS, ...

Let's start with the first one and try to frame knowledge so that we can represent it.

Terminology

Let's introduce a terminological framework to define knowledge

- **Entity**
  It is anything we would like to represent, either real or imagined, anything we can talk about.
  E.g.: Fred, an elephant, walking, Knowledge Representation, the great flight of April 30 2005
- **Unit**
  It is a conceptual data structure we use to collect and frame information about an entity.
  We put a # in front of the symbol we use to reference it
  E.g.: #Fred, #Elephant, #Walking, #KnowledgeRepresentation, #TheGreatFlightOfApril30-2005
- **Slot**
  It is a data structure that contains the value of a property.
  We put a $ in front of the symbol we use to reference it
  E.g.: $Color, $Age, $StartingTime, $ReproducesByEggs

The concept of "Class"

A class is a **set** of entities that share some characteristics

E.g.: the class of Ostriches, the class of Events, the class of Engineers

What is a class for?

- It defines the properties common to a set of entities
  E.g.: Ostriches are birds, none of them can fly, their height is on average 1.5 meters, ...
- It is used to classify entities
  E.g.: if we know that something is a bird, cannot fly and its height is more or less 1.50 meters than we can say that it might be an Ostrich
- It makes it possible to make default assumptions
  E.g.: If we know that Fred is an Ostrich, than we can assume by default that it does not fly although it is a bird, ...

How can we represent a class?

A class is an entity, so we can use a unit to represent it

```
Unit #Ostrich
    $IsA (#Collection)
    $GeneralizedBy (#Bird)
    $SpecializedBy (#AfricanOstrich)
    $Instances (#Fred #Mary)
    $InstProperties (($Height 1.50)($Flies False))
```

Problems about classes

- **Typical or constitutive properties**
  The only properties that have to be present in all the instances belonging to a class are those defined by conventions (e.g., for triangles NumberOfAngles is 3, Elephants are all mammals, ...)
  The other properties can be considered as common to most of the instances, and, in any case, characteristic (e.g. elephants have long noses, but an elephant who has lost the nose is still an elephant).
  For these properties we might assume that instances have *default* values, but if we observe a different value for a given instance, this has to be accepted (overriding).
- **Incompleteness of information**
  When is it possible to say that an instance belongs to a class?

The concept of "Instance"

An instance is an **entity** that belongs to a class

E.g.: Fred the Ostrich, the great flight of April 30 2005, ...

What is an instance for?

- It frames all the properties related to an entity
  E.g.: Fred is an Ostrich, its height is 1.48 cm, ...
  the great flight of April 30 2005, is an Event, its type is Flight, its date is "April, 30 2005".
- It belongs to a class

How can we represent an instance?

An instance is an entity, so we can use a unit to represent it

```
Unit #FredTheOstrich
     $InstanceOf (#Ostrich)
     $Height (1.57)
     $Name ("Fred")
```

Problems about instances

- **Identification**
  Instances are internally identified by their identifier (#Fred), but it might be described with the same values for their properties, this means that they are identical for the KBS. This might not be a problem, and it depends on applications.
- **Classification**
  When is it possible to say that an instance belongs to a class?

The concept of "Property"

A **property** is the minimal chunk of description for an entity

E.g.: the property "height" for ostriches, the property "is a" for the class of ostriches, ...

What is a property for?

- It identifies the basic knowledge chunk that describes an entity E.g.: Fred is described by its height the fact that is an instance of the class Ostrich, ...
- It is used in any inferential procedure
- It can, in turn, be described by properties E.g.: How much reliable are its values, what are the typical range or the default values, ...

How can we represent a property?

A property is an entity, so we can use a unit to represent it

```
Unit #Height
     $IsA (#Property)
     $MakesSenseFor (#PhysicalObjects)
     $EntryIsA (#Measure)
     $EntryFormat (#SingleEntry)
```

Problems about properties

How can we decide where to put properties?
For instance, let's try to code the knowledge stated as "the fire brigade vehicles are red"

- The property ($Colour #Red) in the class #FireBrigadeVehicle)
- The property ($Colour #Red) in the class #FireBrigadeVehicle)
- The class #RedThing of which #FireBrigadeVehicle is a specialization
- The class #RedCar of which #FireBrigadeVehicle is a specialization

The concept of "Hierarchy"

A **hierarchy** is an ordering relationship among classes according to some property

E.g.: the generalizazion hierarchy is defined on $GeneralizedBy, the part-of hierarchy on $Part-Of, ...

What is a hierarchy for?

It implements the inferential mechanism of **inheritance**
E.g.:
The #Ostrich class inherits from #Birds through the **$SpecializedBy** hierarchy the fact that their instances have feathers ($HaveFeathers True)
#Fred inherits from #Ostrich all the properties of ostriches through the **$InstanceOf** hierarchy
#Screw112 inherits from #MyComputer through the hierarchy **$PartOf** the fact that belongs to #Andrea, as well as #MyComputer

Something more about hierarchies

- It is possible to define multiple inheritance
  E.g.: Ostriches are both Birds and Runners
- The property values are collected from the bottom (Overriding) E.g.: If #Birds has among its $InstProperties that ($Flies True), and we define of the specializing #Ostrich class an $InstProperties as ($Flies False), then this last will be valid for all instances of# Ostrich (and eventually specializing classes) since they are below in the hierarchy and property values are collected from below

Test: declarative knowledge

Let's represent the knowledge contained in this text

- An engineer is a person
- Aristides is an engineer
- Engineers have a master degree
- Aristides got its degree on July, 25 2009
- Aristides likes dancing
- Engineers like computers and do not like dancing

Inference

What is inference? Obtain knowledge

from already represented knowledge

Inferential structures

We can represent inferential structures in the general form of **rules**.
$IS < string >< rule >$

$$
\begin{aligned}
&< rule > ::= \ If(< composite\_clause >) \\
&\qquad\qquad Then(< composite\_clause >) \\
&< clause > ::= < ref > ((< slot\_name >< slot\_value >)^*) \\
&< composite\_clause > ::= < clause > \mid NOT \ < composite\_clause > \\
&\qquad\qquad \mid < composite\_clause > \ AND \ < composite\_clause > \\
&\qquad\qquad \mid < composite\_clause > \ OR \ < composite\_clause > \\
&< slot\_value > ::= < ref > \mid < string > \mid < number > \\
&< ref > ::= \ \text{“}reference\ to\ a\ Unit\text{”} \mid < variable > \\
&< variable > ::= ? < string >
\end{aligned}
$$

How inference structures are used?

Clauses in the antecedent are matched with declarative knowledge and, if matching, the knowledge in the consequent is asserted in the knowledge model. For instance, we may define:

SI Inverse
If(?*X*($*HasFather*?*Y*))
Then(?*Y*($*FatherOf*?*X*))

Given this, if there is an instance:

Unit #*Marco*
$*InstanceOf*(#*Person*)
$*HasFather*(#*Bruno*)

we can re-assert Unit #Bruno as:

Unit #*Bruno*
$*InstanceOf*(#*Person*)
$*FatherOf*(#*Marco*)

Test: Inference

Let's represent the knowledge present in the following sentence, adding that needed to complete the diagnosis of a car that does not start when turned on with its key.

A car cannot start when the security system is active, when the battery is exhausted, or when the fuel tank is empty.