Foundations of Artificial Intelligence

Knowledge-Based Systems

Knowledge Representation and Reasoning
The field of knowledge engineering can be defined as the process of assessing problems, acquiring knowledge and building knowledge based systems.

An artificial intelligence system is capable of not only storing and manipulating data, but also of acquiring, representing, and manipulating knowledge.
Knowledge Based Systems

http://coventry.bcs.org/resources/artil.htm
Knowledge Based Systems

In automated AI systems, key issues:

- **knowledge acquisition**
  The transformation of potential problem-solving expertise from some knowledge source to a program

- **knowledge representation**
  as a set of sentences of first order logic symbolic encoding of propositions

- **knowledge reasoning**
  deducing logical consequences
  manipulation of symbols encoding propositions to produce representations of new propositions
Knowledge Representation
Knowledge Representation

When we try to build the **knowledge base** we have the problem of how to represent it.

We could just write down what we are told but, as the information grows, it becomes more and more difficult to keep track of the relationships between the items.
To build the **knowledge base** we have the problem of how to represent it.

**Knowledge representation** concerns the mismatch between human and computer 'memory'.

We call these representations **knowledge bases**, and the operations on these knowledge bases, **inference engine**.
Knowledge Representation

What to represent?

**Facts:** truths about the real world and what we represent.

**Representation of the facts:** which we manipulate. We define the representation in terms of *symbols* that can be manipulated by programs.
**Knowledge Representation**

Simple representation

<table>
<thead>
<tr>
<th>Musician</th>
<th>Style</th>
<th>Instrument</th>
<th>Age</th>
</tr>
</thead>
<tbody>
<tr>
<td>Miles Davis</td>
<td>Jazz</td>
<td>Trumpet</td>
<td>deceased</td>
</tr>
<tr>
<td>John Zorn</td>
<td>Avant Garde</td>
<td>Saxophone</td>
<td>35</td>
</tr>
<tr>
<td>Frank Zappa</td>
<td>Rock</td>
<td>Guitar</td>
<td>deceased</td>
</tr>
<tr>
<td>John McLaughlin</td>
<td>Jazz</td>
<td>Guitar</td>
<td>47</td>
</tr>
</tbody>
</table>

Simple way to store facts. Each fact about a set of objects is set out systematically in columns. Little opportunity for inference.
Natural language is an obvious way of representing and handling facts. However:

- Natural language is often ambiguous
- Syntax and semantics are not fully understood
- There is little uniformity in the structure of sentences

spot is a dog
all dogs have tails
We can then deduce:
spot has a tail
Knowledge Representation

Four General Representation Types

Logical Representations

Semantic Networks

Production Rules

Frames
Knowledge Representation

Predicate Logic – Set Theory

Use a formal language which is capable of representing and mathematically manipulating logical thought.

It suggests a powerful way of deriving new knowledge from old, through mathematical deduction.

- Blue Cars
- Cars with Automatic Gears
- Blue Cars with Automatic Gears
Knowledge Representation

Predicate Logic – Vocabulary

- The logical **and** called the *conjunction* of two logical propositions
- The logical **or** called the *disjunction* of two logical propositions
- The negation, or **not** connective
- The **implies** connective

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot; → &quot;</td>
<td>implication (implies)</td>
</tr>
<tr>
<td>&quot; ¬ &quot;</td>
<td>not</td>
</tr>
<tr>
<td>&quot; ∨ &quot;</td>
<td>or</td>
</tr>
<tr>
<td>&quot; ∧ &quot;</td>
<td>and</td>
</tr>
<tr>
<td>&quot; ∀ &quot;</td>
<td>for all</td>
</tr>
<tr>
<td>&quot; ∃ &quot;</td>
<td>there exists</td>
</tr>
</tbody>
</table>
Knowledge Representation

Predicate Logic - Examples

It is raining.

RAINING

It is sunny.

SUNNY

It is windy.

WINDY

If it is raining, then it is not sunny.

RAINING → ¬ SUNNY
Consider the following fact as: 
spot is a dog as \(\text{dog}(\text{spot})\)

We could then represent that all dogs have tails with 
\(\text{dog}(x) \rightarrow \text{hasatail}(x)\)

We can then deduce: 
\(\text{hasatail}(\text{spot})\)

Using an appropriate mapping function the sentence spot has a tail can be generated
1. A is an elephant
   • Elephant (A)

2. Elephants are animals
   • Animals (elephants)

3. A likes fruit
   • Like (A, fruit)

4. All animals eat fruit or meat
   • ∀ (x) : animal (x) → eat (x, fruit) ∨ eat (x, meat)
5. Elephants are grey
   • Grey (elephant)

6. All animals eat the food they like
   • $\forall(x)\exists(y):$ animal($x$) $\land$ food($y$) $\land$ like($x$, $y$) $\rightarrow$ eat($x$, $y$)

7. Elephants are not carnivores
   • Elephant ($x$) $\rightarrow$ $\neg$ carnivore ($x$)

8. Fruit is food
   • Food (Fruit)
The **Semantic Net** is a formal graphic language representing facts about entities about which we could reason.
Knowledge Representation

Semantic Nets

Diagram:
- Two nodes: Dave and Steve
- Connections:
  - Dave to height to $H_1$
  - Steve to height to $H_2$
  - $H_1$ to greater-than to $H_2$
  - Value from $H_1$ to 160
Knowledge Representation

Frames

Frames represent an alternative way to structure and organise knowledge. A frame system is a hierarchy of frames.

Each frame has:

- a **name**;
- **slots**: properties of the entity that has the name, and their values.
Knowledge Representation

Frames – An Example

- there is a category of things called cars

- a car has 4 wheels, is moved by an engine, and runs on petrol or diesel
Frames – An Example

-there is a particular type of car called a VW, manufactured in Germany

- slots and values in the previous frame will be inherited
- there is a particular type of VW called a Golf, which has a sunroof
- there is a particular type of Golf called a TDi, which runs on diesel. A TDi has 4 cylinders, and an engine capacity of 1.8 litres
- my car, called C637SRK, is a Golf Tdi. It hasn’t got a sun-roof.

- fifth frame: an instance frame.

- The top slot appears here as the value contradicts (overwrites) the value which would otherwise be inherited.
Production rule systems: based on the general underlying idea of **condition-action** pairs (also called **if-then** pairs, **production rules**, or just plain **productions**).

A production rule is written in the form

“if this condition holds, **then** this action is appropriate”.

They are capable of modelling any computable procedure.
(rule (name)
   (if (trigger fact 1)
      (trigger fact 2)
      ...
      (trigger fact n))
   (then (conclusion fact 1, or action 1 )
      (conclusion fact 2, or action 2)
      ...
      (conclusion fact n, or action n)))
Knowledge Representation

Rules

if it is raining then the ground is wet

if height of X > height of Y then X is taller than Y
where X and Y are variables, and the knowledge base has the following items:

it is raining

the ground is wet

height of Tom = 6
height of Tim = 5

Tom is taller than Tim
Knowledge Acquisition
Human knowledge is complex, unstructured and usually ill formulated.

Often the expert is so close to the problem under consideration they have difficulty in seeing it objectively.

This situation is worse when the knowledge source comprises of several experts.
Knowledge Acquisition

Stages of Acquisition

1. Define task
2. Build-up Domain Vocabulary
   - Words, phrases, formulae that make up the natural language of the task.
3. Develop a Model of the Reasoning Involved and how it is applied.
   - Flowcharts and decision trees often used.
   - Paper exercise - no programming at this stage.
   - Iterative procedure with Experts
Knowledge Acquisition

Spider Diagrams

- Simple enough that the structure is self evident
- Powerful enough to express complex structures
- Flexible enough to accommodate the inevitable flow of changes and revisions
MILTON: “Well, the routers are usually OK, so always check the server first. If the traffic coming from the server is OK, then check the routers. Check the mixer temperature, if it’s above 20°C then the cooling fan has gone and needs replacing. If it’s not then check the connections they may be loose and need re-soldering. If they’re OK it has to be the cabling that is problem, so check that and replace if necessary. If the cabling is not damaged then it is beyond me, call in the manufacturer’s repair team.”

MILTON: “If the problem is in the server, check the disks first – repairing any bad sectors should fix it. If not check the power supply, if the supply is spiking, replace it. If none of this works then the problem must be with the processor.”
Create a domain dictionary for the above problem

<table>
<thead>
<tr>
<th>Traffic OK</th>
<th>Yes, No</th>
</tr>
</thead>
<tbody>
<tr>
<td>Problem Machine</td>
<td>Router, Server</td>
</tr>
<tr>
<td>Router Connection</td>
<td>Loose, Not Loose</td>
</tr>
<tr>
<td>Cabling</td>
<td>Damaged, Not Damaged</td>
</tr>
<tr>
<td>Temperature</td>
<td>Numeric value</td>
</tr>
<tr>
<td>Disks</td>
<td>Bad Sectors, No Bad Sectors</td>
</tr>
<tr>
<td>Power Supply</td>
<td>Spiking, Steady</td>
</tr>
<tr>
<td>Problem Solution</td>
<td>Change Cooling Fan, Re-solder Connections, Replace Cabling, Call Manufacturer’s Repair Team, Fix Bad Disk Sectors, Replace Power Supply, Processor Problem</td>
</tr>
</tbody>
</table>
Knowledge Acquisition

An Example

Build decision tree(s) for the above system

- Traffic OK
  - Problem: Router
  - Problem: Server
Reasoning
A good knowledge representation scheme has to allow easy, natural and plausible reasoning.

The reasoning part of a knowledge based system is the **inference engine**.
If I tell you:

"All boys have a mother"

and that:

"John is a boy"

What can you tell me about John?

You are able to figure out that:

"John has a mother"

without being explicitly told this fact.
"All boys have a mother"
"John is a boy"
John has a mother"

To know that John is a boy or that all boys have mothers merely requires retrieval of previously stored facts.

To know that John has a mother requires the ability to reason from the available facts.
Reasoning: to use some known knowledge to deduce logical consequences (new propositions). When we require any knowledge system to do something it has not been explicitly told how to do it, it must reason.

If we know:
- Robins are birds.
- All birds have wings.
Then if we ask:
- Do robins have wings?

Some reasoning (albeit very simple) is needed to answer the question.
Rule of **modus ponens**: the simplest and most fundamental form of **inference**; it was known and described by Aristotle over 2000 years ago.

The schema is simply:

```
from  if $p$ then $q$
and    $p$
infer $q$
```
Substitution

A formal way of reasoning
Allows us to replace universally quantified variables by any constants
Substitute a variable by a known constant, and to make two logical expressions look the same
Extremely important in reasoning
Substitution

\[ \forall y \text{ King}(y) \land \text{ Greedy}(y) \Rightarrow \text{ Evil}(y) \]
\[ \text{King}(\text{John}) \]
\[ \text{Greedy}(\text{John}) \]

We put ‘John’ where there is a variable ‘y’
The idea is to make two logical expressions look the same

\[ \forall y \text{ King}(\text{John}) \land \text{ Greedy}(\text{John}) \Rightarrow \text{ Evil}(\text{John}) \]
\[ \text{King}(\text{John}) \land \text{ Greedy}(\text{John}) \]