TOPICS IN INFORMATION PROCESSING (1997-98)

EXPERT SYSTEMS

1. Overview
2. Representation
3. Development

Frans Coenen

Room G33, email `frans`

DEFINITION

- An **Expert System** (ES) is a computer system that behaves like a human expert in a particular (narrow) domain.
- Example application areas include (1) medicine, (2) geology, (3) computer configuration and (4) chemical analysis.
- **Knowledge Based Systems** (KBS) are almost synonymous with ES except that in the case of KBS the system is expected to behave in an expert manner rather than emulating a human expert (there is a distinction).
- **Rule Based systems**, are also like ES or KBS, but operate at a less expert level.

REPRESENTATION

- Knowledge in ES is typically expressed using **predicate calculus**.
- Predicate calculus is a powerful tool which underlies many aspects of computer science, e.g. database schema, machine learning, theorem proving, Etc.
- Predicates are of the form:

```
predicate_name(Arg1,Arg2 ... ArgN)
```

- Arguments can be constants or variables.
- Predicate calculus is more complex than is required for ES and therefore a simpler **clausal from** tends to be used sometimes referred to as **Horn clauses** or **Production Rules**.

PRODUCTION RULES

- Production rules are typically of the form:

```
if <ANTECEDENT> then <CONSEQUENT>.
```

- or in horn clause format:

```
<CONSEQUENT> :- <ANTECEDENT>.
```

where the consequent part comprises a single predicate and the antecedent part zero, one or more conjoined (‘‘anded’’) predicates.

- If a clause has no consequent part the antecedent part is considered to be ‘‘unconditionally’’ true. In such cases the clause is usually referred to as a **fact**.

EXAMPLE

```
/* facts */

parent(tom,mary).
parent(tom,susan).
parent(ann,tom).

/* Rules */

grandParent(ann,susan):-
        parent(ann,Y),
        parent(Y,susan).

sibling(mary,susan):-
        parent(mary,Z),
        parent(susan,Z).
```

RESOLUTION

- The principal advantage of clausal forms is that reasoning can be implemented using a technique referred to as **proof rule resolution**.
- Using this technique an **inference engine** attempts to satisfy the antecedent part of a rule by replacing it with facts and/or other rules.
- This process continues until only unconditionally true facts remain, it which case the antecedent part is said to "succeed" and the truth of the consequent part is established.
- If suitable replacements are not found the antecedent is said to "fail" and we conclude that the consequent is false.

UNIFICATION

- The process of finding a matching substitution which will make an antecedent part of a rule equal to a consequent part of another rule (or a fact) so that the resolution can be applied is called **unification**.
- Two predicates match if:
  - □ They have the same predicate name.
  - □ They have the same number of arguments.
  - □ Constants are either the same or can be paired with variables.
- (Unification is also sometimes referred to as **pattern matching**.)

CHAINING

- The process of moving from antecedent patterns to consequent patterns is referred to as **forward chaining**. (The reverse is **backward chaining**.)
- Most ES operate using forward chaining.
- A rule is said to be **triggered** if its consequent is matched with a predicate contained in the antecedent of some other rule.
- A rule is said to have **fired** if (once it has been triggered) its antecedents are satisfied.

BINDING

- During the matching process variables can be matched with constants, we say that the constant and variable are **bound** to each other.
- We also sometimes say that a variable is **instantiated** with a value.

EXAMPLE

```
/* facts */

parent(tom,mary).
parent(tom,susan).
parent(ann,tom).

female(mary).
female(susan).
female(ann).

male(tom).

/* Rules */

grandParent(X,Y):-
        parent(X,Y),
        parent(Y,Z).

grandMother(X,Y):-
        grandParent(X,Y),
        female(X).
```

DEVELOPING ES

- The engineering of ES systems requires understanding of the knowledge underpinning the application.
- Unlike more main stream software applications (e.g. data processing, number crunching etc.) the nature of this knowledge is often not obvious.
- Questions that must be answered include:
  - □ What kind of knowledge is involved?
  - □ How should knowledge be represented?
  - □ How much knowledge is required
  - □ What knowledge is needed?
- ES software engineering processes therefore require the inclusion of a **knowledge acquisition** and **knowledge analysis** phases.

COMPARISON OF LIFE-CYCLES

Traditional	ES
System analysis	Knowledge elicitation
Requirements analysis and definition	Knowledge analysis Domain conceptualisation
Software specification and design	Knowledge representation
Implementation	Implementation
Testing	Testing
Operation	Operation
Maintenance	Maintenance

INCREMENTAL ES LIFE-CYCLE

- ES are generally developed using the **incremental**, **evolutionary** or **experimental** life cycle.
- Based on ideas concerning rapid prototyping.
- Start by building a working system to address a sub-set of the domain.
- Then modify and expand this initial system.
- Continue until solution to problem domain is satisfactory.

KNOWLEDGE ELICITATION

- Knowledge elicitation, also sometimes referred to as knowledge acquisition, is the process of extracting, from expert sources, knowledge about a problem domain.
- In the case of ES this is done through interviews with experts.
- Aim is to elicit rules.
- Formal techniques exist.

KNOWLEDGE ANALYSIS

- During knowledge analysis relevant knowledge is identified and extracted from the raw knowledge collected during knowledge elicitation.

- Formal techniques exist, e.g. entity-attribute-value searches.

DOMAIN CONCEPTUALISATION

- In the domain conceptualisation stage the knowledge engineer attempts to impose a global structure onto the analysed knowledge to produce a model of the domain.
- Formal techniques exist, e.g. flow diagrams, networks, tables or taxonomic hierarchies.
- During this stage ‘‘holes’’ in the knowledge may be discovered necessitating a return to the knowledge elicitation stage.

IMPLEMENTATION

- ES can be implemented using any programming paradigm.
- However some paradigms are more suitable than others.
- The declarative paradigm offers many of the features which are desirable with respect to ES:

  - Symbolic manipulation
  - Non procedural processing
  - Recursion and non-destructive assignment to support back tracking

  E.G. PROLOG

SUGGESTED SEMINAR TITLES

1. ES Applications

2. Techniques to develop ES

3. Knowledge elicitation and analysis

4. Knowledge representation