Prof. S. C. Sakure

Rule-based Expert Systems

What is knowledge?

- is a theoretical or practical understanding of a subject or a domain.
- is also the sim of what is currently known, and apparently knowledge is power. Those who possess knowledge are called experts.
- Anyone can be considered a domain expert if he or she has deep knowledge and strong practical experience in a particular domain.
- The human mental process is internal, and it is too complex to be represented as an algorithm
- However, most experts are capable of expressing their knowledge in the form of rules for problem solving.

IF the 'traffic light' is 'green'
THEN the action is go

IF the 'traffic light' is 'red'
THEN the action is stop

Rules as a Knowledge Representation Technique

- The term rule in AI, which is the most commonly used type of knowledge representation, can be defined as an IF-THEN structure that relates given information or facts in the IF part to some action in the THEN part.

- A rule provides some description of how to solve a problem.

- Rule are relatively easy to create and understand

- Any rules consists of two parts: the IF part, called the *antecedent* (premise or condition) and the THEN part called the *consequent* (conclusion or action)

  IF <antecedent>
  THEN <consequent>

- A rule can have multiple antecedents joined by the keywords **AND (conjunction)**, **OR (disjunction)** or a combination of both.

IF	<antecedent 1>	IF	<antecedent 1>
AND	<antecedent 2>	OR	<antecedent 2>
	:		:
AND	<antecedent *n*>	OR	<antecedent *n*>
THEN	<consequent>	THEN	<consequent>

- The antecedent of a rule incorporates two parts: an *object (linguistic object)* and its *value*. The object and its value are linked by an *operator*.

- The operator identifies the object and assigns the value. Operators such as *is*, *are*, *is not*, *are not* are used to assign a symbolic value to a linguistic object.

- Expert systems can also used mathematical operators to define an object as numerical and assign it to the numerical value.

  IF ‘age of the customer’ < 18
  AND ‘cash withdrawal’ > 1000
  THEN ‘signature of the parent’ is required

- Rules can represent:

  - **Relation**: IF the ‘fuel tank’ is empty THEN the car is dead

  - **Recommendation**: IF the season is autumn AND the sky is cloudy AND the forecast is drizzle THEN the advice is ‘take an umbrella’

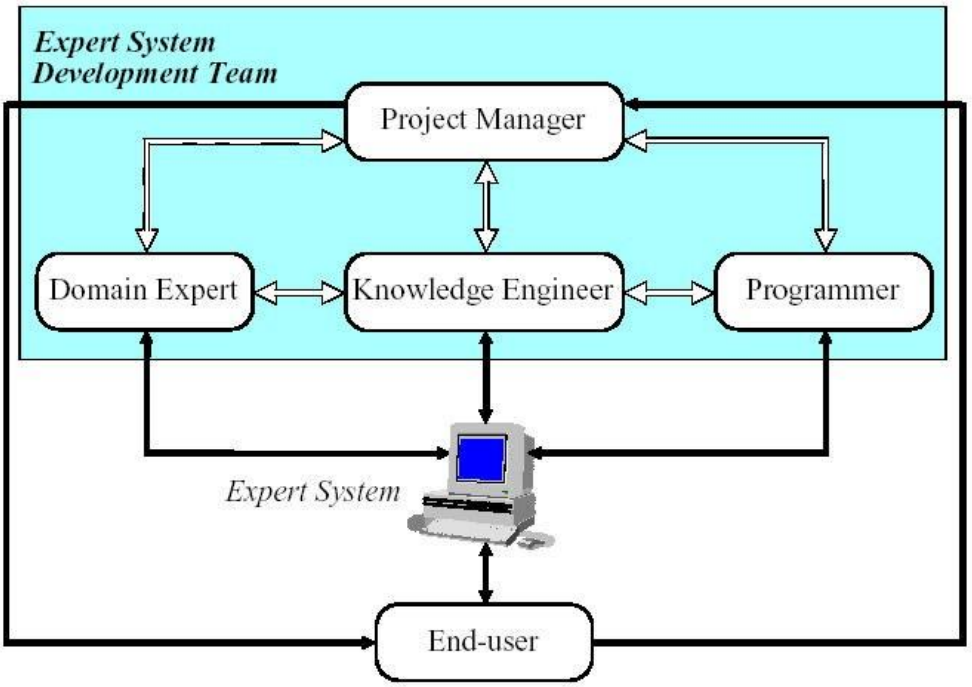
  - **Directive**: IF the car is dead AND the ‘fuel tank’ is empty THEN the action is ‘refuel the car’

- **Strategy**: IF the car is dead THEN the action is 'check the fuel tank'; step1 complete
  IF step1 is complete AND the 'fuel tank' is full THEN the action is 'check the battery'; step2 is complete

- **Heuristic**: IF the spill is liquid AND the 'spill pH' < 6 AND the 'spill smell' is vinegar THEN the 'spill material' is 'acetic acid'

The main players in the development team

- There are five members of the expert system development team:
  1. domain expert
  2. knowledge engineer
  3. programmer
  4. project manager
  5. end-user



- **Domain Expert**:
  - is a knowledgeable and skilled person capable of solving problems in a specific area or domain
  - the person's expertise is to be captured in the expert system
  - could be more than one expert that contribute to an expert system
  - the expert must be able to communicate his or her knowledge, be willing to participate in the expert system development and commit a substantial amount of time to the project
  - is the most important person in the expert system development team

- **Knowledge Engineer**:
  - is someone who is capable of designing, building and testing an expert system
  - interviews the domain expert to find out how a particular problem is solved
  - establishes what reasoning methods the expert uses to handle facts and rules and decides how to represent them in the expert system
  - choose some development software or an expert systems shell, or look at programming languages for encoding the knowledge
  - responsible for testing, revising and integrating the expert system into the workplace

- **Programmer**:
  - is the person responsible for the actual programming, describing the domain knowledge in terms that a computer can understand.
  - needs to have the skills in symbolic programming in such AI language such as Prolog.
  - should also know conventional programming language like C, Pascal, FORTRAN and Basic

- **Project Manager**:
  - is the leader of the expert system development team, responsible for keeping the project on track
  - makes sure that all deliverables and milestones are met, interacts with the expert, knowledge engineer, programmer and end-user

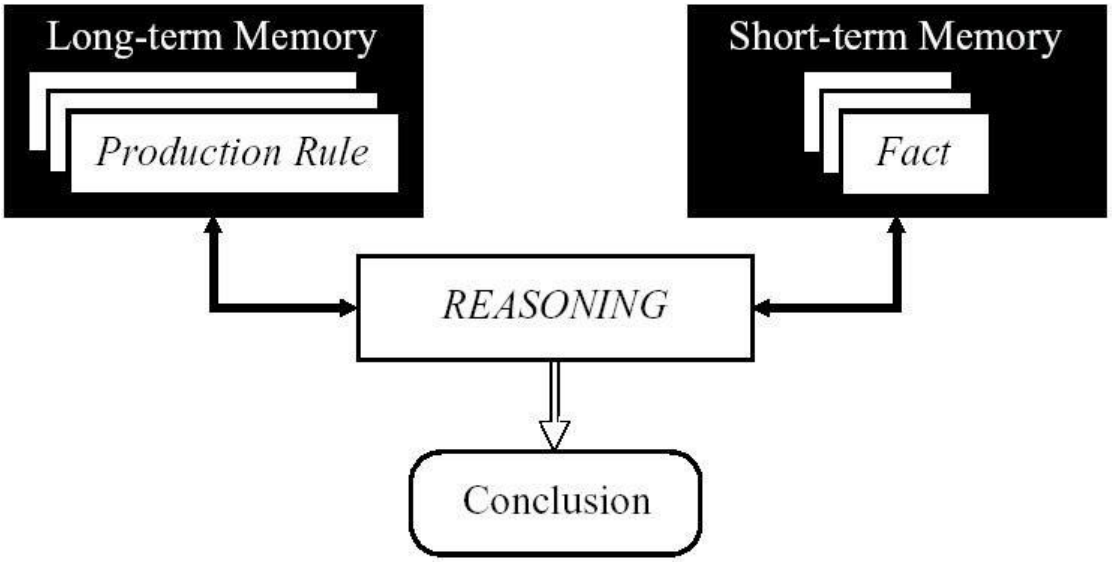
- **End-user**:
  - often called the user
  - is a person who uses the expert system when it is developed
  - must not only be confident in the expert system performance but also feel comfortable using it

Structure of a rule-based expert system

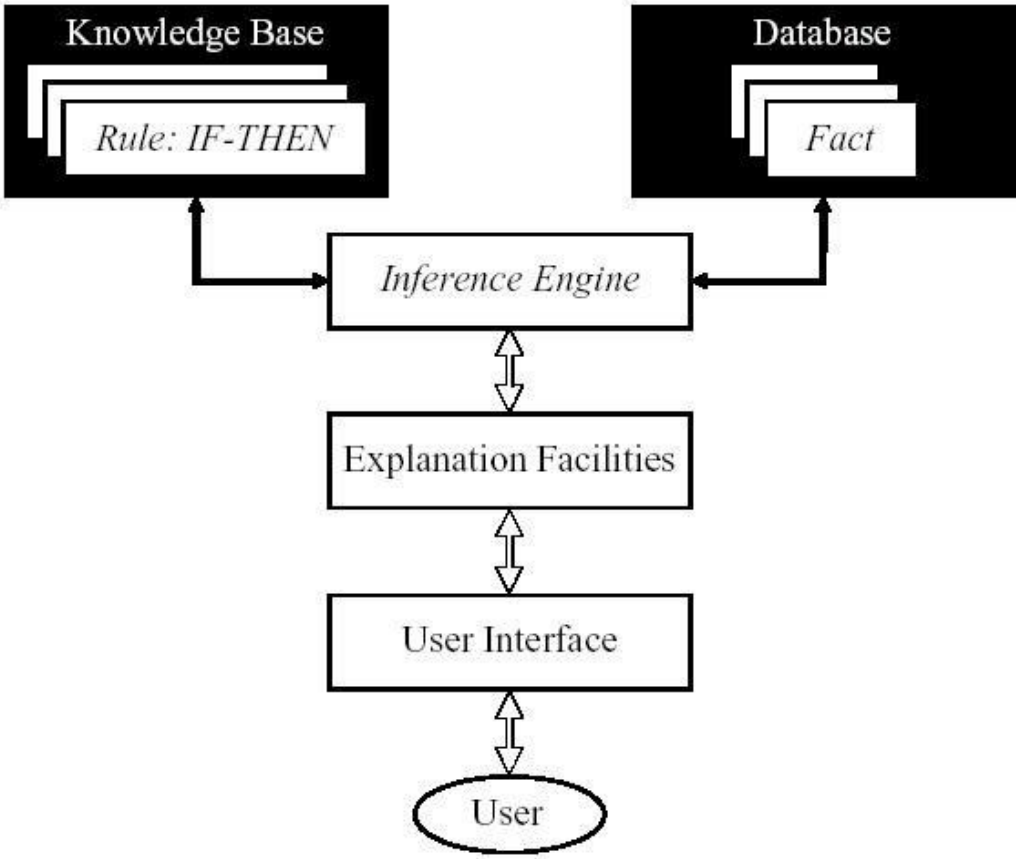
- In early 70s, Newell and Simon from Carnegie-Mellon University proposed a production system model, the foundation of the modern rule-based expert systems

- The production model is based on the idea that humans solve problems by applying their knowledge (expressed as production rules) to a given problem represented by problem-specific information

- The production rules are stored in the long-term memory and the problem-specific information or facts in the short-term memory



- Basic Structure of a rule-based expert system



- **Knowledge base** contains the domain knowledge useful for problem solving

- In rule-based expert system, the knowledge is represented as a set of rules. Each rule specifies a relation, recommendation, directive, strategy or heuristic and has the IF (condition) THEN (action) structure.

- When the condition part of a rule is satisfied, the rule is said to fire and the action part is executed

- The **database** includes a set of facts used to match against the IF (condition) parts of rules stored in the knowledge base.

- The **inference engineer** carries out the reasoning whereby the expert system reaches a solution. It links the rules given in the knowledge base with the facts provided in the database.

- The **explanation facilities** enable the user to ask the expert system how a particular conclusion is reached and why a specific fact is needed

- The **user interface** is the means of communication between a user seeking a solution to the problem and an expert system

Comparison of expert systems with conventional systems and human experts

Human Experts	*Expert Systems*	*Conventional Programs*
Use knowledge in the form of rules of thumb or heuristics to solve problems in a narrow domain.	Process knowledge expressed in the form of rules and use symbolic reasoning to solve problems in a *narrow domain*.	Process data and use algorithms, a series of well-defined operations, to solve general numerical problems.
In a human brain, knowledge exists in a compiled form.	Provide a *clear separation of knowledge from its processing*.	Do not separate knowledge from the control structure to process this knowledge.
Capable of explaining a line of reasoning and providing the details.	*Trace the rules fired* during a problem-solving session and *explain how* a particular conclusion was reached and *why* specific data was needed.	Do not explain how a particular result was obtained and why input data was needed.

Human Experts	*Expert Systems*	*Conventional Programs*
Use inexact reasoning and can deal with incomplete, uncertain and fuzzy information.	Permit *inexact reasoning* and can deal with incomplete, uncertain and fuzzy data.	Work only on problems where data is complete and exact.
Can make mistakes when information is incomplete or fuzzy.	*Can make mistakes* when data is incomplete or fuzzy.	Provide no solution at all, or a wrong one, when data is incomplete or fuzzy.
Enhance the quality of problem solving via years of learning and practical training. This process is slow, inefficient and expensive.	Enhance the quality of problem solving by adding new rules or adjusting old ones in the knowledge base. When new knowledge is acquired, *changes are easy* to accomplish.	Enhance the quality of problem solving by changing the program code, which affects both the knowledge and its processing, making changes difficult.

Conflict Resolution

- Let us consider two simple rules for crossing a road. And let us now add the third rule:

- Rule 1:
  IF the 'traffic light' is green
  THEN the action is go

- Rule 2:
  IF the 'traffic light' is red
  THEN the action is stop

- Rule 3:
  IF the 'traffic light' is red
  THEN the action is go

- We have two rules, Rule 2 and Rule 3, with the same IF part. Thus both of them can be set to fire when the condition part is satisfied.

- These rules represent a conflict set

- The inference engine must determine which rule to fire from such a set

- A method for choosing a rule to fire when more than one rule can be fired in a given cycle is called **conflict resolution**

Methods used for conflict resolution

- Fire the rule with the ***highest priority***. In simple applications, the priority can be established by placing the rules in an appropriate order in the knowledge base. Usually this strategy works well for expert systems with around 100 rules.

- Fire the ***most specific rule***. This method is also known as the ***longest matching strategy***. It is based on the assumption that a specific rule processes more information than a general one.

- Fire the rule that uses the ***data most recently entered*** in the database. This method relies on time tags attached to each fact in the database. In the conflict set, the expert system first fires the rule whose antecedent uses the data most recently added to the database.

Advantages of rule-based expert systems

- Separation of knowledge from its processing
  - The structure of a rule-based expert system provides an effective separation of the knowledge base from the inference engine.
  - This makes it possible to develop different applications using the same expert system shell.

- Dealing with incomplete and uncertain knowledge
  - Most rule-based expert systems are capable of representing and reasoning with incomplete and uncertain knowledge.

Disadvantages of rule-based expert systems

- Opaque relations between rules.
  - Although the individual production rules are relatively simple and self-documented, their logical interactions within the large set of rules may be opaque.
  - Rule-based systems make it difficult to observe how individual rules serve the overall strategy.

- Ineffective search strategy
  - The inference engine applies an exhaustive search through all the production rules during each cycle.
  - Expert systems with a large set of rules (over 100 rules) can be slow, and thus large rule-based systems can be unsuitable for real-time applications

- Inability to learn
  - In general, rule-based expert systems do not have an ability to learn from the experience.
  - Unlike human expert, who knows when to “break the rules”, an expert system cannot automatically modify its knowledge base, or adjust existing rules or add new ones.

- The knowledge engineer is still responsible for revising and maintaining the system

Uncertainty Management in Rule-based Expert Systems

- Information can be incomplete, inconsistent, uncertain, or all three.

- Uncertainty is defined as the lack of the exact knowledge that would enable us to reach a perfectly reliable conclusion.

- Classical logic permits only exact reasoning. Classical logic permits only exact reasoning. It assumes that perfect knowledge always exists and the law of the excluded middle can always be applied:

  IF A is true IF A is false
  THEN A is not false THEN A is not true

- Facts and inferences dealt with so far have been categorical - either true or false; real life facts and rules are often less than certain

- Uncertainty can be expressed numerically as certainty/confidence factor **(cf)** or measure of belief **(mb)**

- *cf* usually is a real number in a particular range, eg, 0 to 1 or -1 to 1
- Various schemes have been proposed to deal with uncertainty

Combining certainties of propositions and rules

Let **P1** and **P2** be two propositions and **cf(P1)** and **cf(P2)** denote their certainties

Then

cf(P1 and P2) = min(cf(P1), cf(P2))

cf(P1 or P2) = max(cf(P1), cf(P2))

Given the rule

if P1 then P2: cf = C

then certainty of **P2** is given by

cf(P2) = cf(P1) * C

Examples:

if it rains then John will catch a taxi: cf = 0.6

Suppose cf for 'it rains' is 0.5,

then cf of 'John catching a taxi'

= 0.5 * 0.6 = 0.3

if it rains and I forget the umbrella

then I'll get wet: cf = 0.9

cf of 'if it rains and I forget the umbrella'

= min(0.7, 0.6)

= 0.6

cf of 'I'll get wet'

= cf(if it rains and I forget the umbrella) * 0.9

= 0.6 * 0.9

= 0.54

- Uncertainty handling schemes such as the one above are not based on probability theory
- Difficulty with using probability theory:
  1. Difficult for human experts to express likelihood estimates in terms of probabilities
  2. Not all information are available for correct probabilistic treatment of uncertainties
- eg, to evaluate certainty of rule

  if A or B then C
- needs probabilities of both A and B, as well as correlation between occurrences of A and B