Introduction

- Knowledge representation (KR) is as significant a factor in determining the success of a system as the software that uses the knowledge.

- KR is of major importance in expert systems for two reasons:
  - Expert systems are designed for a certain type of knowledge representation based on rules of logic called inferences.
  - KR is important as it affects the development, efficiency, speed, and maintenance of the system.
The meaning of knowledge

- Two special types of knowledge
  - a priori
  - a posteriori

- A priori knowledge
  - comes before and is independent of knowledge from the senses
  - is considered to be universally true and cannot be denied without contradiction
  - examples of a priori knowledge: logic statements, mathematical laws, and the knowledge possessed by teenagers

- The opposite of a priori knowledge is knowledge derived from the senses: a posteriori knowledge
  - since sensory experience may not always be reliable, a posteriori knowledge can be denied on the basis of new knowledge without the necessity of contradictions
The meaning of knowledge

- **Wisdom**: Using knowledge in a beneficial way
- **Metaknowledge**: Rules about knowledge
- **Knowledge**: Rules about using information
- **Information**: Potentially useful for knowledge
- **Data**: Potentially useful information
- **Noise**: No apparent information.
The meaning of knowledge

- As an example:
  137178766832525156430015
  - Without knowledge, this entire sequence appears to be noise
  - if it is known that this sequence is meaningful, then the sequence is data

- Certain knowledge may transform data into information
  - For example, the following algorithm processes the data to yield information:
    Group the numbers by twos.
    Ignore any two-digit numbers less than 32.
    Substitute the ASCII characters for the two-digit numbers.

- Application of this algorithm to the previous 24 numbers yields the information:
  GOLD 438+
The meaning of knowledge

- Now knowledge can be applied to this information:
  
  IF gold is less than 500
  and the price is rising (+)
  THEN
  buy gold
Knowledge Acquisition

- **Knowledge engineer**
  An AI specialist responsible for the technical side of developing an expert system. The knowledge engineer works closely with the domain expert to capture the expert’s knowledge in a knowledge base.

- **Knowledge engineering (KE)**
  The engineering discipline in which knowledge is integrated into computer systems to solve complex problems normally requiring a high level of human expertise.
Production Systems

- Represent knowledge in terms of multiple rules that specify what should be or should not be concluded in different situations.
- A rule-based system consists of IF-THEN rules, facts, and an interpreter.
- Rules are popular for a number of reasons:
  - Modular nature
    - easy to encapsulate knowledge and expand the expert system by incremental development
  - Explanation facilities
    - By keeping track of which rules have fired, an explanation facility can present the chain of reasoning that led to a certain conclusion.
  - Similarity to the human cognitive process
Post production systems

- Production systems were first used in symbolic logic by Post who originated the name.
- The basic idea of Post was that any mathematical or logic system is simply a set of rules specifying how to change one string of symbols into another set of symbols.
- Given an input string, the antecedent, a production rule could produce a new string, the consequent.
Post production systems

- A production rule could be:
  \[ \text{Antecedent} \rightarrow \text{Consequent} \]
  person has fever \rightarrow take aspirin

- In terms of the IF-THEN notation as:
  IF person has fever THEN take aspirin

- The production rules can also have multiple antecedents
  person has fever AND
  fever is greater than 102 \rightarrow see doctor
Post production systems

A Post production system consists of a group of production rules

1. car won't start $\rightarrow$ check battery
2. car won't start $\rightarrow$ check gas
3. check battery AND battery bad $\rightarrow$ replace battery
4. check gas AND no gas $\rightarrow$ fill gas tank
Post production systems

- The rules could have been written in any order
  (4) check gas AND no gas $\rightarrow$ fill gas tank
  (2) car won't start $\rightarrow$ check gas
  (1) car won't start $\rightarrow$ check battery
  (3) check battery AND battery bad $\rightarrow$ replace battery

- The basic limitation is lack of a control strategy to guide the application of the rules

- Therefore, although they were useful in laying part of the foundation of expert systems, they are not adequate.
Markov Algorithms

- A Markov algorithm is an ordered group of productions that are applied in order of priority to an input string.
- It terminates if either,
  - (1) the last production is not applicable to a string or,
  - (2) a production that ends with a period is applied,
- For example, suppose the system consists of the single rule:
  - AB → HIJ
  - input string GABKAB produces the new string GHIJKAB
  - the production now applies to the new string, the final result is GHIJKHIJ
Markov Algorithms

- ^ represents the null string
  \[ A \rightarrow ^ \]
  - deletes all occurrences of the character A

- Lowercase letters a, b, c, …x, y, z represent single-character variables
  \[ AxB \rightarrow BxA \]
  - Reverse the characters A and B, where x is any single character

- An example of a Markov algorithm that moves the first letter of an input string to the end
  - The rules are ordered in terms of highest priority (1), next highest (2), and so forth
Markov Algorithms

- The following rules can move the first letter to the end.
- The rules are prioritized in the order that they are entered:
  (x and y are single-character variable; α is a special punctuation)

1. \( \alpha xy \rightarrow yx \alpha \)
2. \( \alpha x \rightarrow x \alpha \)
3. \( \alpha \cdot \rightarrow ^\alpha \cdot \)
4. \( ^\alpha \rightarrow \alpha \)

---

**Execution Trace**

<table>
<thead>
<tr>
<th>Rule</th>
<th>Success / Failure</th>
<th>String</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>F</td>
<td>ABC ∙</td>
</tr>
<tr>
<td>2</td>
<td>F</td>
<td>ABC ∙</td>
</tr>
<tr>
<td>3</td>
<td>F</td>
<td>ABC ∙</td>
</tr>
<tr>
<td>4</td>
<td>S</td>
<td>( \alpha )ABC ∙</td>
</tr>
<tr>
<td>1</td>
<td>S</td>
<td>B( \alpha )AC ∙</td>
</tr>
<tr>
<td>1</td>
<td>S</td>
<td>BC( \alpha )A ∙</td>
</tr>
<tr>
<td>1</td>
<td>F</td>
<td>BC( \alpha )A ∙</td>
</tr>
<tr>
<td>2</td>
<td>S</td>
<td>BCA( \alpha ) ∙</td>
</tr>
<tr>
<td>1</td>
<td>F</td>
<td>BCA( \alpha ) ∙</td>
</tr>
<tr>
<td>2</td>
<td>F</td>
<td>BCA( \alpha ) ∙</td>
</tr>
<tr>
<td>3</td>
<td>S</td>
<td>BCA ∙</td>
</tr>
</tbody>
</table>
The Rete Algorithm

- Markov algorithm is very inefficient for systems with many rules
- A solution is the Rete Algorithm
- The Rete Algorithm functions like a net in holding a lot of information
  - much faster response times and rule firings can occur compared to a large group of IF-THEN rules which must be checked one by one in a conventional program
- The Rete Algorithm is a very fast pattern-matcher that obtains its speed by storing information about the rules in a network in memory
Rule-based Systems

- Consider the problem of deciding to cross a street
- The productions for the two rules are
  - the light is red -> stop
  - the light is green -> go
- An equivalent pseudo code IF-THEN format as:
  Rule: Red_light IF
  IF
  the light is red
  THEN
  stop

  Rule: Green_light IF
  IF
  The light is green
  THEN
  go
Rule-based Systems

- Each rule is identified by a name. Following the name is the IF part of the rule.
- Between the IF and THEN part of the rule is called by various names such as the antecedent, conditional part, pattern part, or left-hand-side (LHS).
- The individual condition:
  
   the light is green

is called a conditional element or a pattern.
Rule-based Systems

- The following are some examples of rules from the classic systems:

  MYCIN system for diagnosis of meningitis and bacteremia (bacterial infections)

  IF
  
  The site of the culture is blood, and
  The identity of the organism is not known with certainty, and
  The stain of the organism is gramneg, and
  The morphology of the organism is rod, and The patient has been seriously burned

  THEN
  
  There is weakly suggestive evidence (.4) that
  the identity of the organism is pseudomonas
Rule-based Systems

XCON/Rl for configuring DEC VAX computer systems

IF

The current context is assigning devices to Unibus modules and
There is an unassigned dual-port disk drive and
The type of controller it requires is known and
There are two such controllers, neither of which has any devices assigned to it, and
The number of devices that these controllers can support is known

THEN

Assign the disk drive to each of the controllers, and
Note that the two controllers have been associated and each supports one drive
Rule Inference

- Two general methods of inferencing used
  - forward chaining
  - backward chaining

- Other methods used
  - means-ends analysis, problem reduction, backtracking, plan-generate-test, hierarchical planning and the least commitment principle, and constraint handling

- Forward chaining is reasoning from facts to the conclusions resulting from those facts

- Backward chaining involves reasoning in reverse from a hypothesis, a potential conclusion to be proved, to the facts that support the hypothesis
Rule Inference

- CLIPS is designed for forward chaining, PROLOG performs backward chaining
- The choice of inference engine depends on the type of problem
  - Diagnostic problems are better solved with backward chaining
  - Prognosis, monitoring, and control are better done by forward chaining
- A rule whose patterns are all satisfied is said to be activated or instantiated.
- Multiple activated rules may be on the agenda at the same time, the inference engine must then select one rule for firing
Rule Inference

- Following the THEN part of a rule is a list of actions to be executed when the rule fires
  - the consequent or right-hand side (RHS)
- The inference engine operates in recognize-act cycles will repeatedly execute a group of tasks until certain criteria cause execution to cease
Rule Inference

- Conflict resolution, act, match, and check for halt:
  
  WHILE not done

  **Conflict Resolution**: If there are activations, then select the one with highest priority, else done.

  **Act**: Sequentially perform the actions on the RHS of the selected activation. Remove the activation that has just fired from the agenda.

  **Match**: Update the agenda by checking if the LHS of any rules are satisfied. If so, activate them. Remove activations if the LHS of their rules are no longer satisfied.

  **Check for Halt**: If a halt action is performed or break command given, then done.

  END-WHILE

  Accept a new user command