Rule-based Expert Systems

- A rule-based expert system is an expert system which works as a production system in which rules encode expert knowledge.
- Most expert systems are rule-based.

- Alternatives are
  - frame-based - knowledge is associated with the objects of interest and reasoning consists of confirming expectations for slot values. Such systems often include rules too.
Production rules

- A production rule consists of two parts: condition (antecedent) part and conclusion (action, consequent) part,
- i.e.: IF (conditions) THEN (actions)

Example

IF Gauge is OK AND [TEMPERATURE] > 120
THEN Cooling system is in the state of overheating

Advantages of rule-based expert systems

- Natural knowledge representation.
  - An expert usually explains the problem-solving procedure with such expressions as this: “In such-and-such situation, I do so-and-so”.
  - These expressions can be represented quite naturally as IF-THEN production rules.

- Uniform structure.
  - Production rules have the uniform IF-THEN structure.
  - Each rule is an independent piece of knowledge.
  - Every syntax of production rules enables them to be self-documented.
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.
  - Thus 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.
  - Example: fuzzy rule based system

Disadvantages of rule-based expert systems

- **Opaque relations between rules**
  - Although individual 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**
  - Inference engine applies an exhaustive search through all the rules during each cycle.
  - Large set of rules (over 100 rules) can be slow, and large rule-based systems can be unsuitable for real-time applications.

- **Inability to learn**
  - Cannot automatically modify its existing rules or add new ones.
  - Knowledge engineer still responsible for revising / maintaining the system.
Rules Techniques

- Rule is a knowledge structure that relates some known information to other information that can be concluded or inferred to be known → Procedural Knowledge
- IF ... THEN ... ELSE
- Ex: IF the ball's color is blue
  THEN I like the ball

  IF Today's time is after 10 am
  AND Today is weekday
  AND I am at home
  OR My boss called and said that I am late for work
  THEN I am late for work
  ELSE I am not late for work

- Use rules to encode the knowledge in the system
- General rule syntax:
  - if <antecedent> then <consequent>
    - if I put my hand on a hot iron,
      then it will burn
    - if I want my car to stop,
      then apply pressure to the brake

- Fact (or assertion):
  - a statement that something is true
    - I put my hand on a hot iron
    - I want my car to stop

Rules Based System
Components of a Rule Based Expert System

- A rule-based expert system contains:
  - Set of rules - stored in knowledge base
  - Working memory or database of facts
  - Rule interpreter - or inference engine
  - User interface
  - Explanation module

- These five components are **essential** for any rule based expert system

Basic structure of a rule-based expert system
Basic Components of a Rule Based Expert System

- The knowledge base:
  - Contains the domain knowledge useful for problem solving.
  - In a rule-based expert system, 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 engine:
  - Carries out the reasoning.
  - 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.
  - An expert system must be able to explain its reasoning and justify its advice, analysis or conclusion.

- The user interface:
  - Means of communication between a user seeking a solution to the problem and an expert system.
Additional Components

- In addition to the essential components there can be:
  - External databases
    - Many Expert System shells have ODBC capabilities
  - External Programs
  - Developer Interface
    - Includes
      - Knowledge base editors
      - Debugging Aids

Additional structure of a rule-based expert system
Rule-based system operation

Knowledge Base
IF Ball’s Color is Red
THEN I Like the Ball
IF I Like the Ball
THEN I Will buy the Ball

Working Memory
Ball’s Color is Red
I Like the Ball
I Will Buy the Ball

Q: Ball’s color?
A: Red

Inferencing Strategies

Two strategies:

- Forward chaining → data driven
- Backward chaining → goal driven
Forward Chaining - Data Driven

- Forward chaining:
  - Is **data driven** reasoning - the reasoning starts from the known data and proceeds forward with that data
  - Start with a set of facts (i.e. assertions)
  - Match conditions of rules against items in database
  - When rule is fired, add consequent to database
  - Continue until no rules left to fire

Inference Engine cycles via a match-fire procedure

*Diagram showing the process: Match and Fire of facts and rules.*
Forward chaining

Cycle One

Database
A B C D E

Knowledge Base
Y & D → Z
X & B & E → Y
A → X
C → L
L & M → N

Match Fire

Cycle Two

Database
A B C D E X Y

Knowledge Base
Y & D → Z
X & B & E → Y
A → X
C → L
L & M → N

Match Fire

Cycle Three

Database
A B C D E

Knowledge Base
Y & D → Z
X & B & E → Y
A → X
C → L
L & M → N

Match Fire
**Forward chaining Example**

**Rule 1**
If patient has sore throat 
And suspect a bacterial infection 
Then patient has strep throat

**Rule 2**
If patient temperature > 100 
Then patient has a fever

**Rule 3**
If patient has been sick over one month 
And patient has a fever 
Then we suspect a bacterial infection

---

**Database**
- patient temperature = 102
- patient been sick for two months
- patient has sore throat

**patient has a fever**

**bacterial infection**

**patient has strep throat**
Backward Chaining – Goal Driven

- In contrast backward chaining:
  - goal driven, try to prove a specific goal
  - Work backwards from a conclusion and try to reach a set of conditions which establish that conclusion.
  - Start with a goal and use this to establish a set of sub-goals.
  - continue until goal is proved (or disproved), or no more matches

Backward chaining

- Backward chaining is the goal-driven reasoning.

- In backward chaining, an expert system has the goal (a hypothetical solution) and the inference engine attempts to find the evidence to prove it.

- First, the knowledge base is searched to find rules that might have the desired solution.

- Such rules must have the goal in their THEN (action) parts.

- If such a rule is found and its IF (condition) part matches data in the database, then the rule is fired and the goal is proved.

- However, this is rarely the case.
Backward chaining

- Thus the inference engine puts aside the rule it is working with (the rule is said to stack)

- And sets up a new goal, a subgoal, to prove the IF part of this rule

- The knowledge base is searched again for rules that can prove the subgoal

- The inference engine repeats the process of stacking the rules until no rules are found in the knowledge base to prove the current subgoal

---

Backward Chaining – Example 1

<table>
<thead>
<tr>
<th>Pass 1</th>
<th>Pass 2</th>
<th>Pass 3</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>A</strong></td>
<td><strong>A</strong></td>
<td><strong>A</strong></td>
</tr>
<tr>
<td><strong>B</strong></td>
<td><strong>B</strong></td>
<td><strong>B</strong></td>
</tr>
<tr>
<td><strong>C</strong></td>
<td><strong>C</strong></td>
<td><strong>C</strong></td>
</tr>
<tr>
<td><strong>D</strong></td>
<td><strong>D</strong></td>
<td><strong>D</strong></td>
</tr>
<tr>
<td><strong>E</strong></td>
<td><strong>E</strong></td>
<td><strong>E</strong></td>
</tr>
<tr>
<td><strong>X</strong></td>
<td><strong>X</strong></td>
<td><strong>X</strong></td>
</tr>
<tr>
<td><strong>Y</strong></td>
<td><strong>Y</strong></td>
<td><strong>Y</strong></td>
</tr>
<tr>
<td><strong>Z</strong></td>
<td><strong>Z</strong></td>
<td><strong>Z</strong></td>
</tr>
<tr>
<td><strong>L</strong></td>
<td><strong>L</strong></td>
<td><strong>L</strong></td>
</tr>
<tr>
<td><strong>M</strong></td>
<td><strong>M</strong></td>
<td><strong>M</strong></td>
</tr>
<tr>
<td><strong>N</strong></td>
<td><strong>N</strong></td>
<td><strong>N</strong></td>
</tr>
<tr>
<td><strong>Goal: Z</strong></td>
<td><strong>Sub-Goal: Y</strong></td>
<td><strong>Sub-Goal: X</strong></td>
</tr>
</tbody>
</table>

Knowledge Base:
- Y & D → Z
- X & B & E → Y
- A → X
- C → L
- L & M → N

Database:
- A
- B
- C
- D
- E

Sub-Goal: Y

Knowledge Base - Y & D → Z
- X & B & E → Y
- A → X
- C → L
- L & M → N

Sub-Goal: X

Knowledge Base - Y & D → Z
- X & B & E → Y
- A → X
- C → L
- L & M → N
Backward Chaining – Example 1

Pass 4

Database

Knowledge Base

\[ Y & D \rightarrow Z \]
\[ X & B & E \rightarrow Y \]
\[ A \rightarrow X \]
\[ C \rightarrow L \]
\[ L & M \rightarrow N \]

Sub-Goal: X

Match

Pass 5

Database

Knowledge Base

\[ Y & D \rightarrow Z \]
\[ X & B & E \rightarrow Y \]
\[ A \rightarrow X \]
\[ C \rightarrow L \]
\[ L & M \rightarrow N \]

Sub-Goal: Y

Match

Pass 6

Database

Knowledge Base

\[ Y & D \rightarrow Z \]
\[ X & B & E \rightarrow Y \]
\[ A \rightarrow X \]
\[ C \rightarrow L \]
\[ L & M \rightarrow N \]

Goal: Z

Match

Backward Chaining – Example 2

Rule 1
If patient has sore throat
And suspect a bacterial infection
Then patient has strep throat

Rule 2
If patient temperature > 100
Then patient has a fever

Rule 3
If patient has been sick over one month
And patient has a fever
Then we suspect a bacterial infection

Start with same set of facts:
- patient temperature = 102
- patient has been sick for two months
- patient has sore throat

But now start with goal
Patient has a strep throat
And try to prove this given the rules and the facts.
Example 2: Backward Chaining

Example 2: Backward Chaining

- Strep throat?
- Sore throat
- bacterial infection
- fever
- Temp>100
- Sick > One month

Choosing between forward and backward chaining?

- If an expert first needs to gather some information and then tries to infer from it whatever can be inferred, choose the forward chaining inference engine.

- However, if your expert begins with a hypothetical solution and then attempts to find facts to prove it, choose the backward chaining inference engine.
Forward Chaining - Evaluation

- **Advantages:**
  - Works well when problem naturally begins by gathering information
  - Planning, control, monitoring

- **Disadvantages:**
  - Difficult to recognise if some evidence is more important than others
  - May ask unrelated questions

Backward Chaining - Evaluation

- **Advantages:**
  - Remains focused on a goal
  - Produces a series of questions that are relevant
  - Good for diagnosis

- **Disadvantages:**
  - Will continue to follow a line of reasoning even when it should switch.