

# Expert System

Prof. Dr. A K M Akhtar Hossain  
Dept. of CSE, RU

# What is Expert System?

---

- **Expert System** is an interactive and reliable computer-based **decision-making system** which uses both facts and heuristics to solve complex decision-making problems.
- It is considered at the highest level of human intelligence and expertise.
- The purpose of an expert system is to solve the most complex issues in a **specific domain**.

# Expert Systems (ES)

---

- Expert systems are knowledge based programs which provide expert quality solutions to the problems in specific domain of applications.
- The core components of expert system are
  - **Knowledge Base** and
  - **Navigational Capability (Inference Engine)**

[Later on we discuss on these topics.]

# Expert Systems (ES)

---

- Generally its knowledge is extracted from human experts in the domain of application by knowledge Engineer.
- **A process of gathering knowledge from domain expert and codifying it according to the formalism is called knowledge engineering.**

# Applications of ES

---

- Stock market trading
- Airline scheduling & Cargo scheduling
- Information management
- Hospitals and medical facilities
- Design and Manufacturing
- Computer maintenances & Virus detection
- Planning and scheduling
- Warehouse optimization
- Financial decision making Knowledge publishing
- Monitoring & control Office or Factory
- Agricultural Plants Disease Management System
- Satellite control missile system

# Phases in building Expert System

---

There are five basic **steps/phases** to build an expert system. These are as follows:

- **Identification Phase:**
- **Conceptualization Phase:**
- **Formalization Phase:**
- **Implementation Phase:**
- **Testing Phase:**

# Phases in building Expert System

---

- **Identification Phase:**

- Knowledge engineer finds out important features of the problem with the help of domain expert (human).
- He tries to determine the type and scope of the problem, the kind of resources required, goal and objective of the ES.

- **Conceptualization Phase:**

- In this phase, knowledge engineer and domain expert decide the concepts, relations and control mechanism needed to describe a problem solving.

# Phases in building Expert System

---

- **Formalization Phase:**

- It involves expressing the key concepts and relations in some framework supported by ES building tools.
- Formalized knowledge consists of data structures, inference rules, control strategies and languages for implementation.

- **Implementation Phase:**

- During this phase, formalized knowledge is converted to working computer program initially called prototype of the whole system.

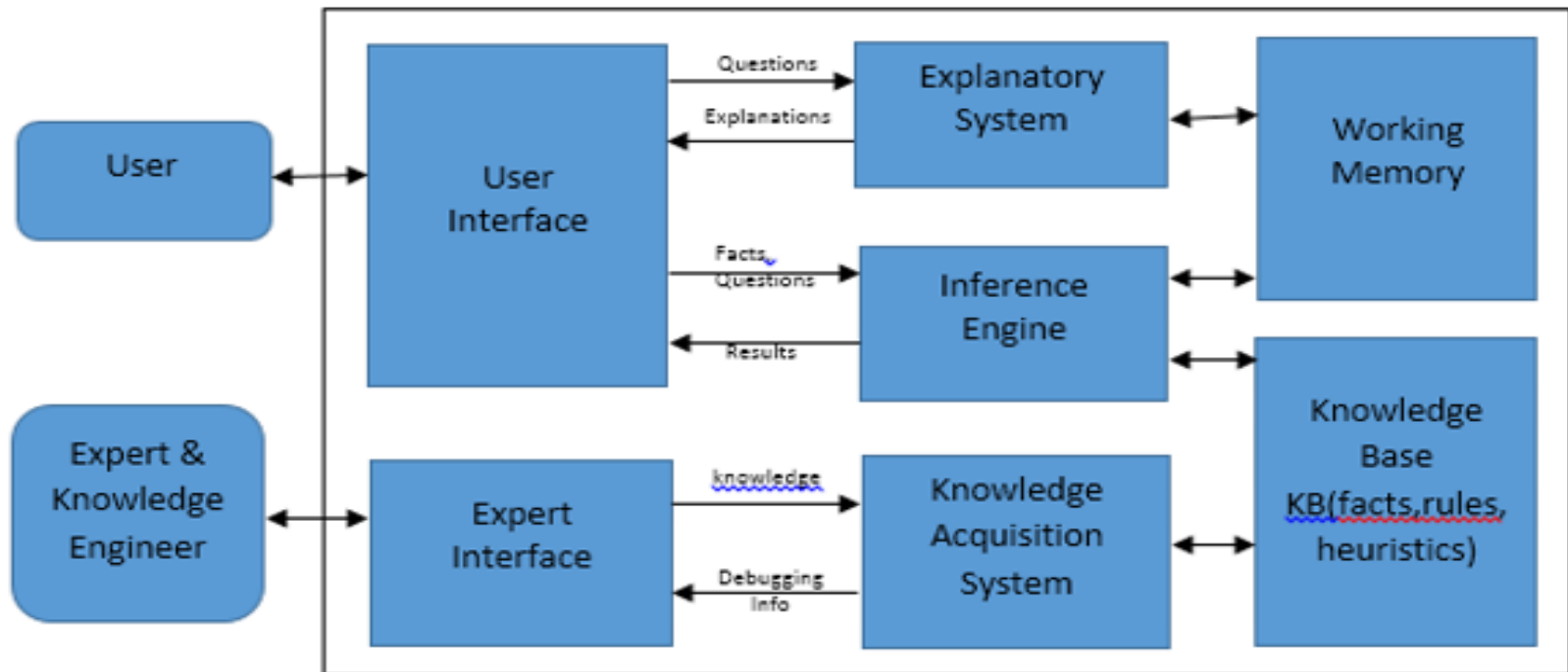
- **Testing Phase:**

- It involves evaluating the performance and utility of prototype systems and revising it, if it is needed.
- Domain expert evaluates the prototype system and his feedback help knowledge engineer to revise it.



# Expert System Architecture

Architecture of Expert System



**The Architecture of an Expert System (ES) consists of the following major components:**

# Knowledge Base (KB)

---

- **KB consists of knowledge about problem domain in the form of static and dynamic databases.**
- Static knowledge consists of
  - rules and facts which is compiled as a part of the system and does not change during execution of the system.
- **Dynamic knowledge consists of facts related to a particular consultation of the system.**
  - At the beginning of the consultation, the dynamic knowledge base often called working memory is empty.
  - **As a consultation progresses, dynamic knowledge base grows and is used along with static knowledge in decision making.**
- **Working memory** is deleted at the end of consultation of the system.

# Navigational Capability /Inference Engine

- Inference Engine consists of inference mechanism and control strategy.
- Inference means search through knowledge base and derive new knowledge.
- It involve formal reasoning involving matching and unification similar to the one performed by human expert to solve problems in a specific area of knowledge.
- **Inference operates by using modus ponens rule.**
- Control strategy determines the order in which rules are applied.
- There are mainly two types of control mechanism. These are **forward chaining** and **backward chaining**.

# Navigational Capability /Inference Engine

---

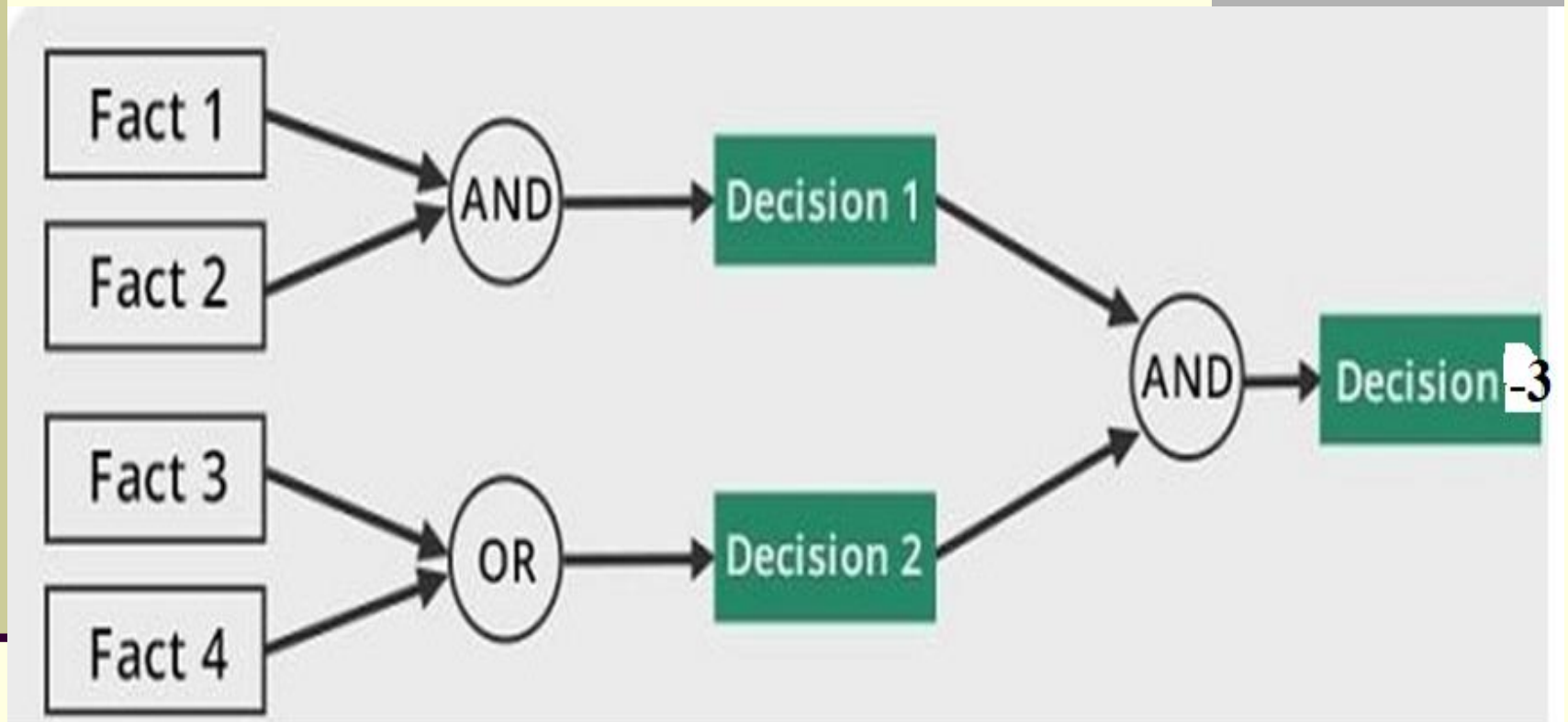
- **Forward Chaining:** A reasoning process that begins with the known facts and works forward, trying to find a successful goal from the known facts.
- **Backward Chaining:** A type of reasoning process that begins at a specified goal and works backward in an attempt to prove that the goal is true.

# Forward Chaining

---

- Forward chaining is a method of reasoning in artificial intelligence in which inference rules are applied to existing data to extract additional data until an endpoint (goal) is achieved.
- In this type of chaining, the inference engine starts by evaluating existing facts, derivations, and conditions before deducing new information. An endpoint (goal) is achieved through the manipulation of knowledge that exists in the knowledge base.

# Forward Chaining



# Applications of Forward Chaining

---

- Forward chaining can be used in planning, monitoring, controlling, and interpreting applications.
- **Properties of forward chaining:**
- The process uses a down-up approach (bottom to top).
- It starts from an initial state and uses facts to make a conclusion.
- This approach is data-driven.
- It's employed in expert systems and production rule system.

# Forward Chaining

---

## ■ Advantages

- It can be used to draw multiple conclusions.
- It provides a good basis for arriving at conclusions.
- It's more flexible than backward chaining because it does not have a limitation on the data derived from it.



# Forward Chaining

---

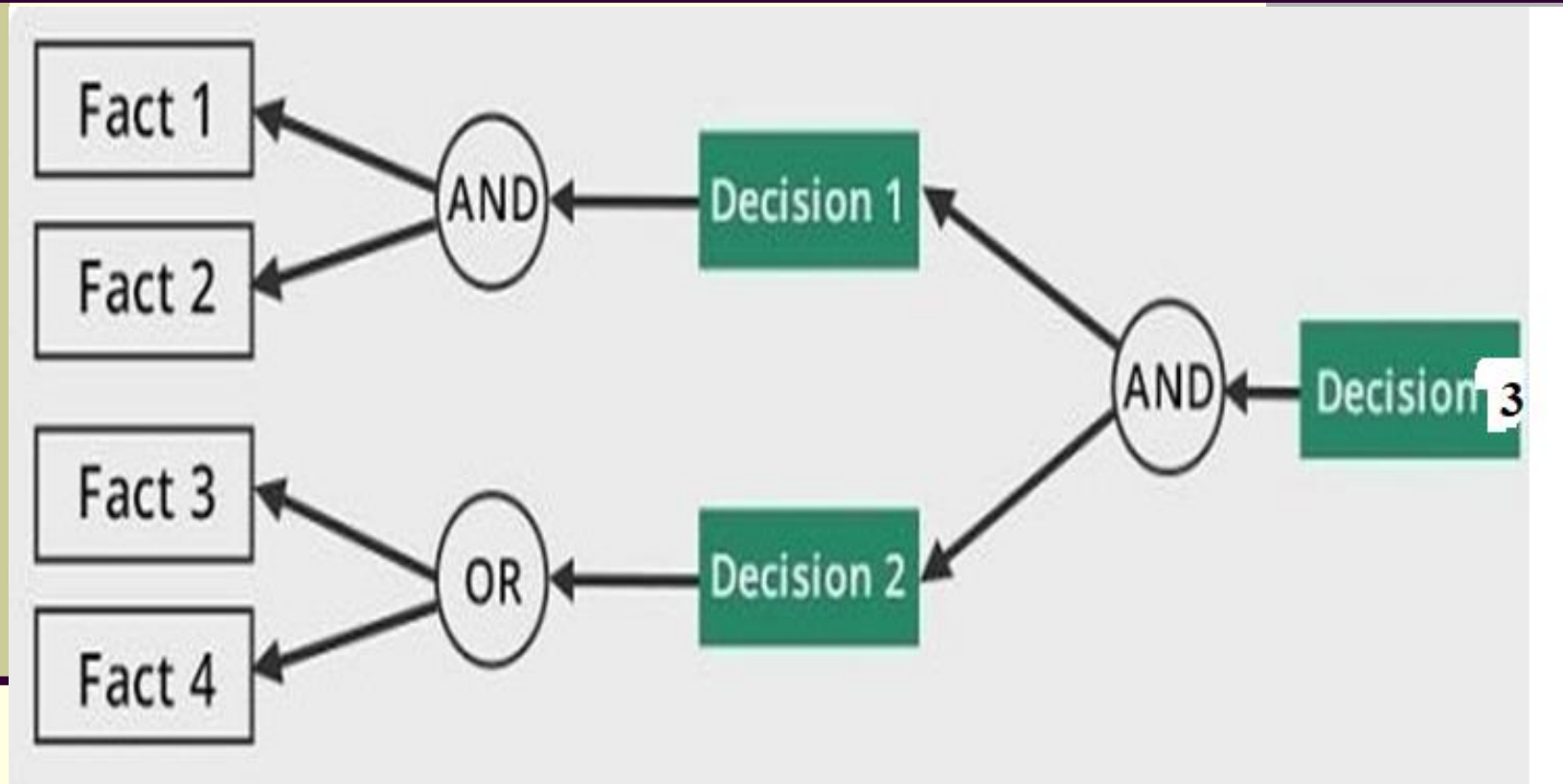
- **Disadvantages**
- The process of forward chaining may be time-consuming. It may take a lot of time to eliminate and synchronize available data.
- Unlike backward chaining, the explanation of facts or observations for this type of chaining is not very clear.
- The former uses a goal-driven method that arrives at conclusions efficiently.

# Backward Chaining

---

- Backward chaining is a concept in artificial intelligence that involves backtracking from the endpoint or goal to steps that led to the endpoint.
- This type of chaining starts from the goal and moves backward to comprehend the steps that were taken to attain this goal.
- The backtracking process can also enable a person establish logical steps that can be used to find other important solutions.

# Backward Chaining



# Applications of Backward Chaining

---

- Backward chaining can be used in debugging, diagnostics, and prescription applications.
- **Properties of backward chaining:**
- The process uses an up-down approach (top to bottom).
- It's a goal-driven method of reasoning.
- The endpoint (goal) is subdivided into sub-goals to prove the truth of facts.
- A backward chaining algorithm is employed in inference engines, game theories, and complex database systems.

# Backward Chaining

## ■ Advantages

- The result is already known, which makes it easy to deduce inferences.
- It's a quicker method of reasoning than forward chaining because the endpoint is available.
- In this type of chaining, correct solutions can be derived effectively if pre-determined rules are met by the inference engine.

## ■ Disadvantages

- The process of reasoning can only start if the endpoint is known.
- It doesn't deduce multiple solutions or answers.
- It only derives data that is needed, which makes it less flexible than forward chaining.

# Knowledge Acquisition

---

- **Knowledge acquisition module allows system to acquire knowledge about the problem domain.**
- Sources of Knowledge for ES
  - text books, reports, research papers, case studies,
  - empirical (experimental) data and
  - domain expert experience.
- **Updation of Knowledge can be done using knowledge acquisition module of the system.**
  - insertion,
  - deletion and
  - updation of existing knowledge

# Explanation Module

---

- Most expert systems have explanation facilities that allow the user to ask the system **why** it asked some question, and **how** it reached to conclusion.
- It contains 'How' and 'Why' modules attached to it.
  - **The sub-module 'How' tells the user about the process through which system has reached to a particular solution.**
  - **'Why' sub-module tells that why is that particular solution offered.**
- It explains user about the reasoning behind any particular problem solution.
- Questions are answered by referring to the system goals, the rules being used, and any existing problem data.

# Explanation Modules Responses

User	Why module
System	Do you have running nose?
User	Yes.
System	I am trying to prove that you have cold. Do you have headache?
User	Yes
System	Do you have sore throat?
User	No
System	Then you do not have cold. Do you have fever?
User	Yes
System	Now I am looking at the possibility of measles. Do you have cough?
User	Yes
System	I can infer measles using rule “If symptoms are fever, cough, running_nose, then patient has measles” measles is concluded.
User	How Module
System	Since you have fever, running_nose and cough and there is a rule “If symptoms are fever, cough, running_nose, then patient has measles”. So measles is concluded for you.



# User Interfaces

- Allows user to communicate with system in interactive mode and helps system to create working knowledge for the problem to be solved.

Dialogue Module (User Interface)	
<b>System</b>	Do you have fever?
<b>User</b>	Yes
<b>System</b>	Do you have bad throat?
<b>User</b>	No
<b>System</b>	Do you have cough?
<b>User</b>	Yes
<b>System</b>	Are you suffering from running nose?
<b>User</b>	Yes
<b>System</b>	Are you suffering from headache?
<b>User</b>	No

# Special interfaces

---

- It may be used for specialized activities such as handling **uncertainty in knowledge**.
- This is a major area of expert systems research that involves methods for reasoning with uncertain data and uncertain knowledge.
- Knowledge is generally incomplete and uncertain.
- To deal with uncertain knowledge, a rule may have associated with it a ***confidence factor or a weight***.
- The set of methods for using uncertain knowledge in combination with uncertain data in the reasoning process is called ***reasoning with uncertainty***.

# Rule Based Expert Systems

---

- **A rule based expert system is one in which knowledge base is in the form of rules and facts.**
  - Knowledge in the form of **rules and facts is most popular way** in designing expert systems.
- It is also called **production system**.

# Examples of Rule-based Systems

---

- ❑ **Medical Diagnosis:**
- Fraud Detection:**
- Quality Control:**
- Decision support systems:**

# Rule Based Expert Systems

- **Example:** Suppose doctor gives a rule for measles as follows:

**"If symptoms are fever, cough, running\_nose, rash and conjunctivitis then patient probably has measles".**

- Turbo Prolog (**/SWI Prolog**) is most suitable for implementing such systems.

**hypothesis(measles) :- symptom(fever), symptom(cough),  
symptom(running\_nose), symptom(conjunctivitis),  
symptom(rash).**

- **[SWI** is derived from Sociaal-Wetenschappelijke Informatica ("Social Science Informatics")]

# Example

- domains
  - disease, indication
  - patient = string
- predicates
  - hypothesis(patient,disease)
  - symptom(name,indication)
  - response(char)
- go
- clauses
  - go :-
    - write("What is the patient's name? "),
    - readln(Patient),
    - hypothesis(Patient,Disease),
    - write(Patient," probably has ",Disease,"/"),nl.

# Example Continue

```
■ go :-  
■         write("Sorry, I don't seem to be able to"),nl.  
■  
■         write("diagnose the disease."),nl.  
■ symptom(Patient,fever) :-  
■         write("Does ",Patient,"have a fever (y/n) ?");  
■         response(Reply),  
■         Reply='y'.  
■ symptom(Patient,rash) :-  
■         write("Does ",Patient,"have a rash (y/n) ?");  
■         response(Reply),  
■         Reply='y'.  
■ symptom(Patient,headache) :-  
■         write("Does ",Patient,"have a headache (y/n) ?");  
■         response(Reply),  
■         Reply='y'.
```

# Example Continue

- symptom(Patient,runny\_nose) :-
  - write("Does ",Patient,"have a runny\_nose (y/n) ?");
  - response(Reply),
  - Reply='y'.
  -
- symptom(Patient,conjunctivitis) :-
  - write("Does ",Patient,"have a conjunctivitis (y/n) ?");
  - response(Reply),
  - Reply='y'.
- symptom(Patient,cough) :-
  - write("Does ",Patient,"have a cough (y/n) ?");
  - response(Reply),
  - Reply='y'.
  -
- symptom(Patient,body\_ache) :-
  - write("Does ",Patient,"have a body\_ache (y/n) ?");
  - response(Reply),
  - Reply='y'.



# Example Continue

- symptom(Patient,chills) :-
  - write("Does ",Patient,"have a chills (y/n) ?");
  - response(Reply),
  - Reply='y'.
  -
- symptom(Patient,sore\_throat) :-
  - write("Does ",Patient,"have a sore\_throat (y/n) ?");
  - response(Reply),
  - Reply='y'.
  -
- symptom(Patient,sneezing) :-
  - write("Does ",Patient,"have a sneezing (y/n) ?");
  - response(Reply),
  - Reply='y'.
  -
- symptom(Patient,swollen\_glands) :-
  - write("Does ",Patient,"have a swollen\_glands (y/n) ?");
  - response(Reply),
  - Reply='y'.

# Example Continue

- hypothesis(Patient,measles) :-
  - symptom(Patient,fever),
  - symptom(Patient,cough),
  - symptom(Patient,conjunctivitis),
  - symptom(Patient,runny\_nose),
  - symptom(Patient,rash).
- hypothesis(Patient,german\_measles)
  - symptom(Patient,fever),
  - symptom(Patient,headache),
  - symptom(Patient,runny\_nose),
  - symptom(Patient,rash).
- hypothesis(Patient,flu) :-
  - symptom(Patient,fever),
  - symptom(Patient,headache),
  - symptom(Patient,body\_ache),
  - symptom(Patient,conjunctivitis),
  - symptom(Patient,chills),
  - symptom(Patient,sore\_throat),
  - symptom(Patient,cough),
  - symptom(Patient,runny\_nose).

# Example Continue


- hypothesis(Patient,common\_cold) :-
  - symptom(Patient,headache),
  - symptom(Patient,sneezing),
  - symptom(Patient,sore\_throat),
  - symptom(Patient,chills),
  - symptom(Patient,runny\_nose).
- hypothesis(Patient,mumps) :-
  - symptom(Patient,fever),
  - symptom(Patient,swollen\_glands),
- hypothesis(Patient,chicken\_pox) :-
  - symptom(Patient,fever),
  - symptom(Patient,rash),
  - symptom(Patient,body\_ache),
  - symptom(Patient,chills).


# Example Continue

- hypothesis(Patient,whooping\_cough) :-
- symptom(Patient,cough),
- symptom(Patient,sneezing),
- symptom(Patient,runny\_nose).
- 
- response(Reply) :-
- readchar(Reply),
- write(Reply),nl.

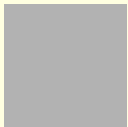
# Rule-based System vs. Learning-based System

<b>Rule-Based Systems</b>	<b>Learning-Based Systems</b>
<b>Uses a set of predetermined rules to make decisions</b>	Learns from data to make decisions
<b>Rules are created by human experts</b>	Knowledge is acquired through machine learning algorithms
<b>Limited ability to adapt to new situations</b>	Can adapt to new situations by learning from data
<b>Transparent decision-making process</b>	Decision-making process may not be transparent
<b>Scalable by adding or modifying rules</b>	Scalable by retraining the model
<b>Consistent in decision-making</b>	Decision-making may not always be consistent
<b>Fast decision-making</b>	May require significant time for training the model



 **THANKS**

 **&**

 **THE END**