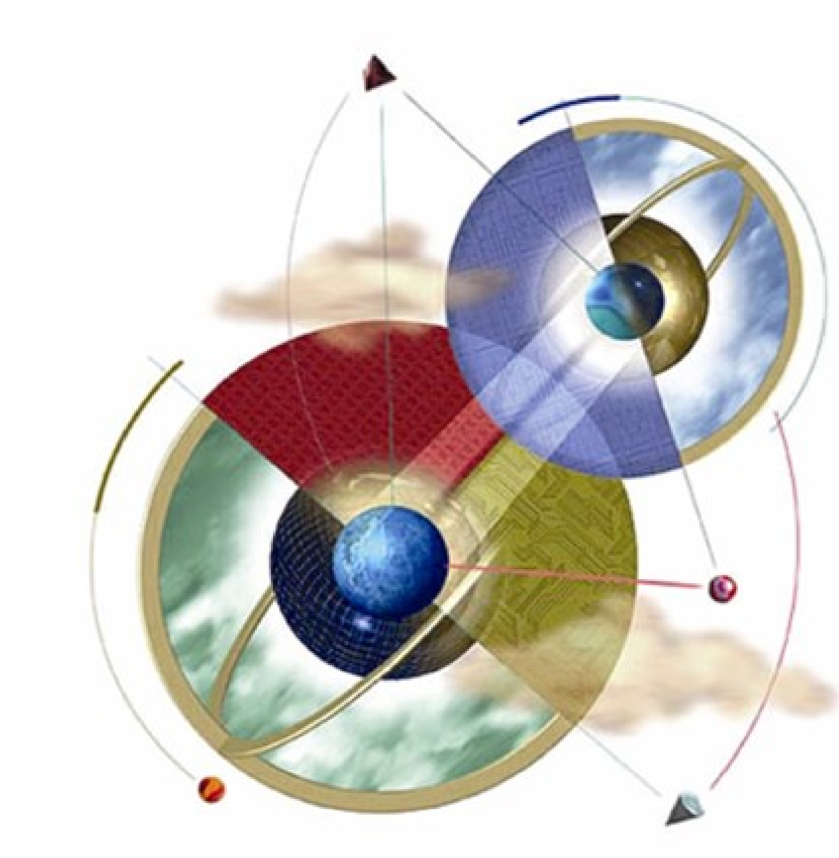


Chapter 1: Introduction to **Expert Systems**

Expert Systems: Principles and Programming, Fourth Edition

What is an Expert System?

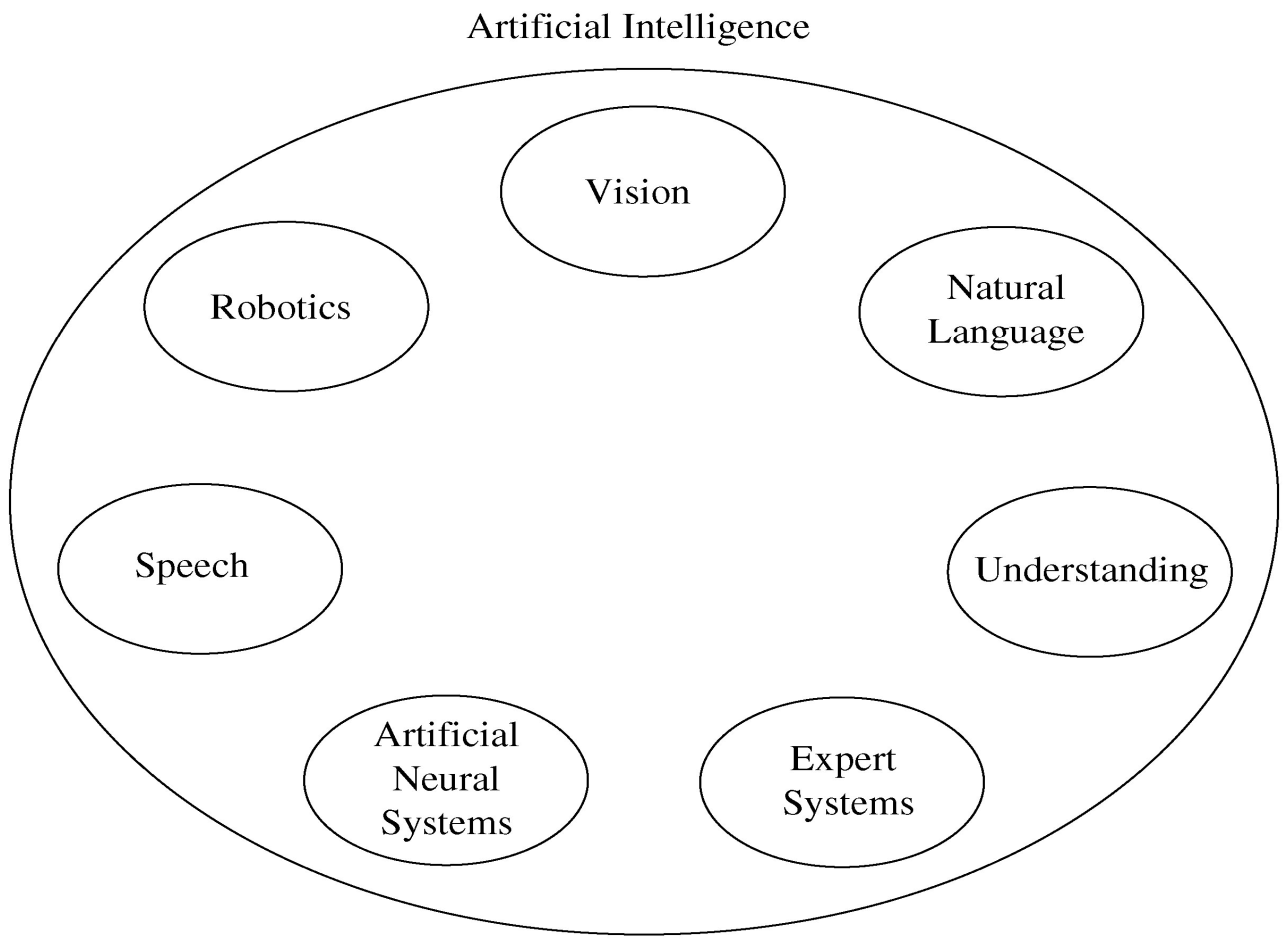
"An expert system is a computer system that emulates, or acts in all respects, with the decision-making capabilities of a human expert."

Prof. **Edward Feigenbaum**

Stanford University

Father of Expert Systems

Fig 1.1: Areas of Artificial Intelligence



Expert System Technology may include:

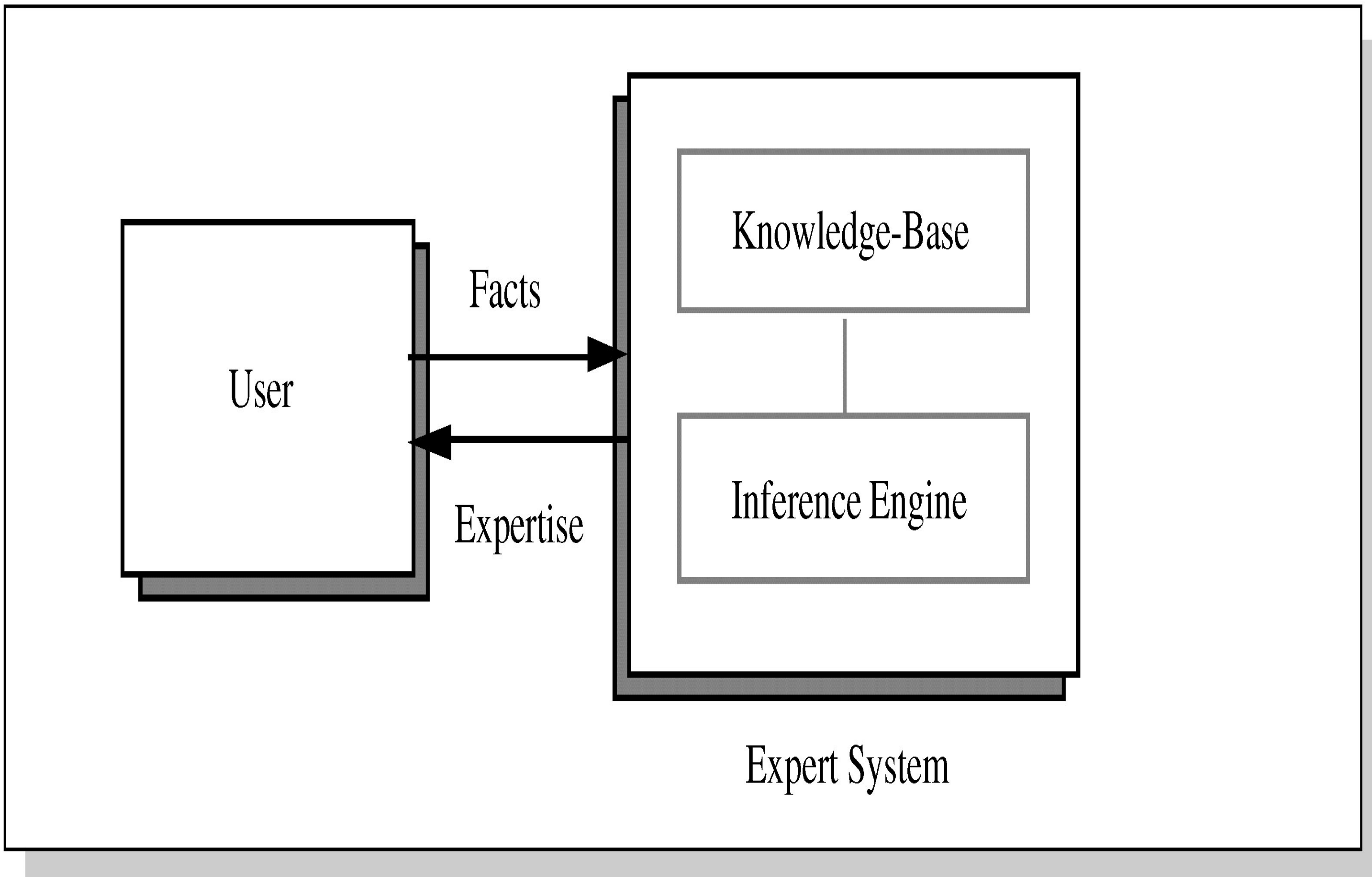
- Special expert system languages – CLIPS
- Programs
- Hardware designed to facilitate the implementation of those systems

Expert System Main Components

- Knowledge base – obtainable from books, magazines, knowledgeable persons, etc.

- Inference engine – draws conclusions from the knowledge base

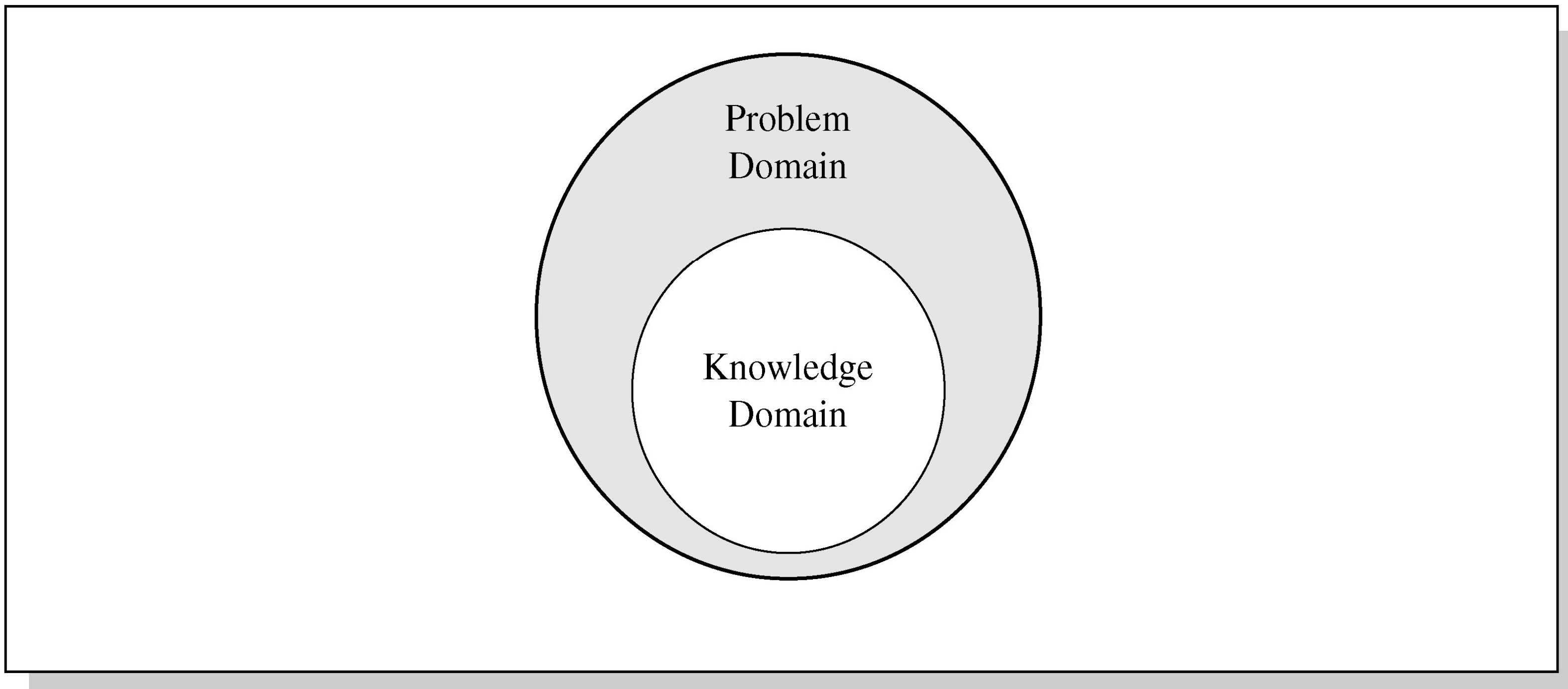
Figure 1.2: Basic Functions of Expert Systems



Problem Domain vs. Knowledge Domain

- An expert’s knowledge is specific to one problem domain – medicine, finance, science, engineering, etc.
- The expert’s knowledge about solving specific problems is called the knowledge domain.
- The problem domain is always a superset of the knowledge domain.

Figure 1.3: Problem and Knowledge Domain Relationship



Advantages of Expert Systems

- Increased availability
- Reduced cost
- Reduced danger
- Performance
- Multiple expertise
- Increased reliability

Advantages Continued

- Explanation
- Fast response
- Steady, unemotional, and complete responses at all times
- Intelligent tutor
- Intelligent database

Representing the Knowledge

The knowledge of an expert system can be represented in a number of ways, including IF-THEN rules:

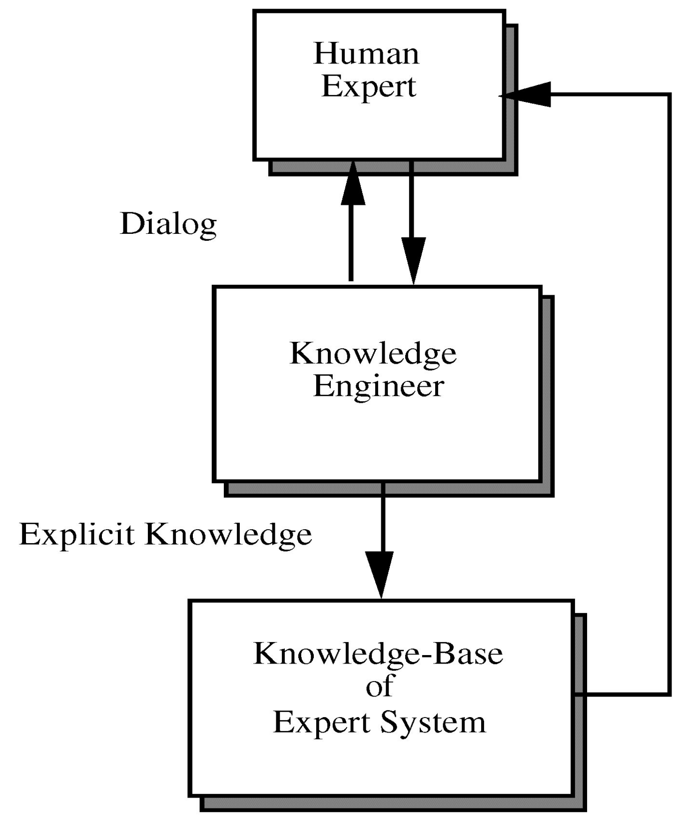
```
IF you are hungry THEN eat
```

Knowledge Engineering

The process of building an expert system:

1. The knowledge engineer establishes a dialog with the human expert to elicit knowledge.
2. The knowledge engineer codes the knowledge explicitly in the knowledge base.
3. The expert evaluates the expert system and gives a critique to the knowledge engineer.

Development of an Expert System



The Role of AI

- An algorithm is an ideal solution guaranteed to yield a solution in a finite amount of time.
- When an algorithm is not available or is insufficient, we rely on artificial intelligence (AI).
- Expert system relies on inference – we accept a "reasonable solution."

Uncertainty

- Both human experts and expert systems must be able to deal with uncertainty.
- It is easier to program expert systems with shallow knowledge than with deep knowledge.
- Shallow knowledge – based on empirical and heuristic knowledge.
- Deep knowledge – based on basic structure, function, and behavior of objects.

Limitations of Expert Systems

- Typical expert systems cannot generalize through analogy to reason about new situations in the way people can.

- A knowledge acquisition bottleneck results from the time-consuming and labor intensive task of building an expert system.

Early Expert Systems

- DENDRAL – used in chemical mass spectroscopy to identify chemical constituents
- MYCIN – medical diagnosis of illness
- DIPMETER – geological data analysis for oil
- PROSPECTOR – geological data analysis for minerals
- XCON/R1 – configuring computer systems

Table 1.3: Broad Classes of Expert Systems

Class	General Area
Configuration	Assemble proper components of a system in the proper way.
Diagnosis	Infer underlying problems based on observed evidence.
Instruction	Intelligent teaching so that a student can ask *why*, *how,* and *what if* questions just as if a human were teaching.
Interpretation	Explain observed data.
Monitoring	Compares observed data to expected data to judge performance.
Planning	Devise actions to yield a desired outcome.
Prognosis	Predict the outcome of a given situation.
Remedy	Prescribe treatment for a problem.
Control	Regulate a process. May require interpretation, diagnosis, monitoring, planning, prognosis, and remedies.

Problems with Algorithmic Solutions

- Conventional computer programs generally solve problems having algorithmic solutions.
- Algorithmic languages include C, Java, and C#.
- Classical AI languages include LISP and PROLOG.

Considerations for Building Expert Systems

- Can the problem be solved effectively by conventional programming?
- Is there a need and a desire for an expert system?
- Is there at least one human expert who is willing to cooperate?
- Can the expert explain the knowledge to the knowledge engineer can understand it.
- Is the problem-solving knowledge mainly heuristic and uncertain?

Languages, Shells, and Tools

- Expert system languages are post-third generation.
- Procedural languages (e.g., C) focus on techniques to represent data.
- More modern languages (e.g., Java) focus on data abstraction.
- Expert system languages (e.g. CLIPS) focus on ways to represent knowledge.

Expert Systems Vs Conventional Programs - I

Characteristic	Conventional Program	Expert System
Control by …	Statement order	Inference engine
Control & Data	Implicit integration	Explicit separation
Control Strength	Strong	Weak
Solution by …	Algorithm	Rules & Inference
Solution search	Small or none	Large
Problem solving	Algorithm	Rules

Expert Systems Vs Conventional Programs - II

Characteristic	Conventional Program	Expert system
Input	Assumed correct	Incomplete, incorrect
Unexpected input	Difficult to deal with	Very responsive
Output	Always correct	Varies with the problem
Explanation	None	Usually
Applications	Numeric, file & text	Symbolic reasoning
Execution	Generally sequential	Opportunistic rules

Expert Systems Vs Conventional Programs - III

Characteristic	Conventional Program	Expert System
Program Design	Structured design	Little or no structure
Modifiability	Difficult	Reasonable
Expansion	Done in major lumps	Incremental

Elements of an Expert System

- User interface – mechanism by which user and system communicate.
- Exploration facility – explains reasoning of expert system to user.
- Working memory – global database of facts used by rules.
- Inference engine – makes inferences deciding which rules are satisfied and prioritizing.

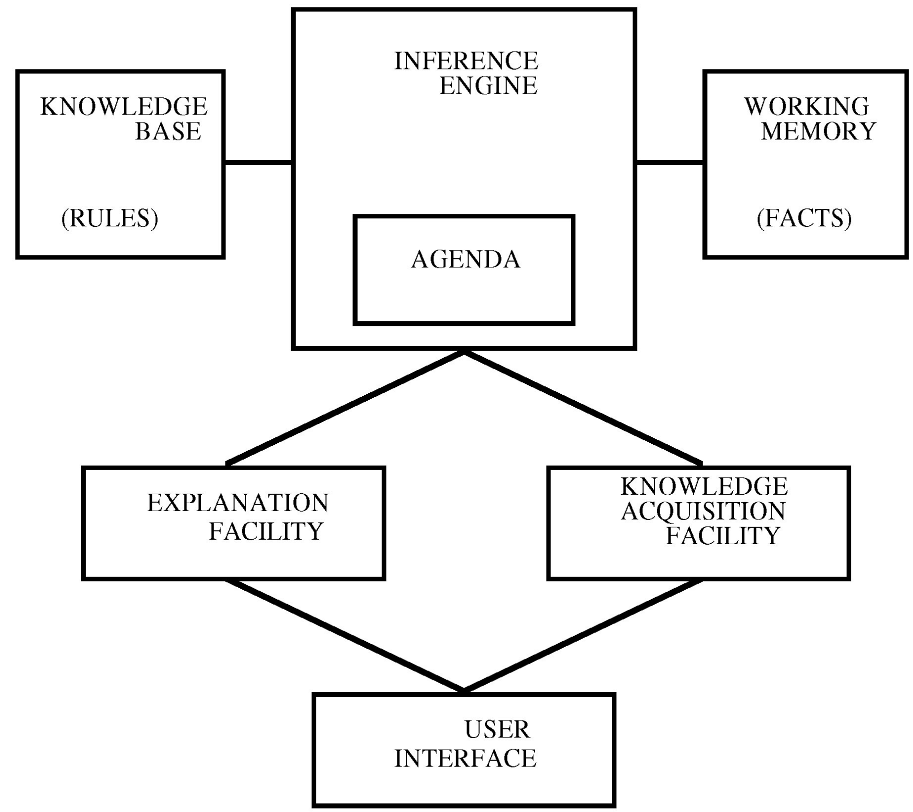
Elements Continued

- Agenda – a prioritized list of rules created by the inference engine, whose patterns are satisfied by facts or objects in working memory.
- Knowledge acquisition facility – automatic way for the user to enter knowledge in the system bypassing the explicit coding by knowledge engineer.
- Knowledge Base – includes the rules of the expert system

Production Rules

- Knowledge base is also called production memory.
- Production rules can be expressed in IF-THEN pseudocode format.
- In rule-based systems, the inference engine determines which rule antecedents are satisfied by the facts.

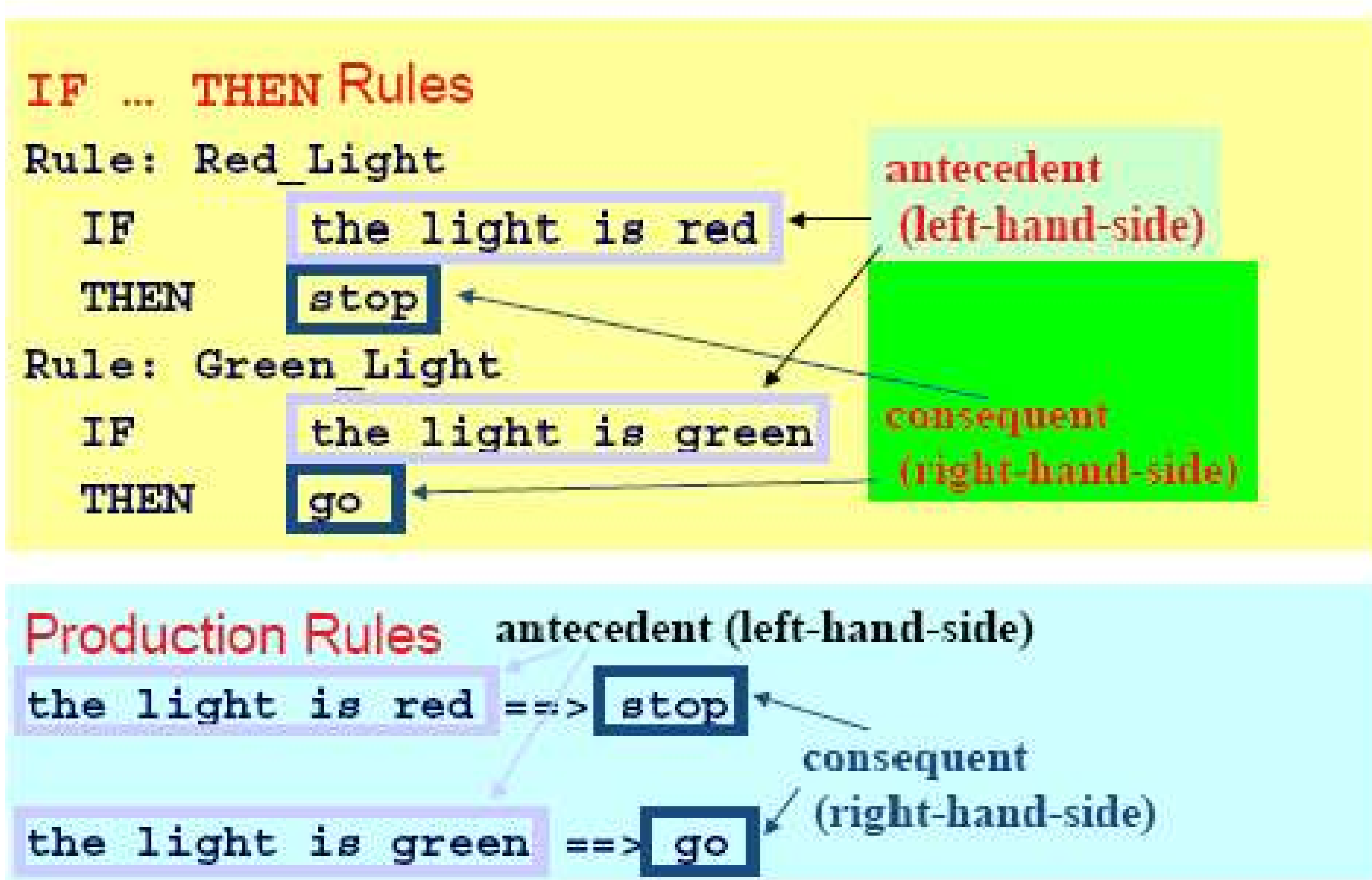
Figure 1.6: Structure of a Rule-Based Expert System



Rule-Based ES

- knowledge is encoded as `IF` ... `THEN` rules
  - these rules can also be written as *production rules*
- the inference engine determines which rule antecedents are satisfied
  - the left-hand side must "match" a fact in the working memory
- satisfied rules are placed on the agenda
- rules on the agenda can be activated ("fired")
  - an activated rule may generate new facts through its right-hand side
  - the activation of one rule may subsequently cause the activation of other rules

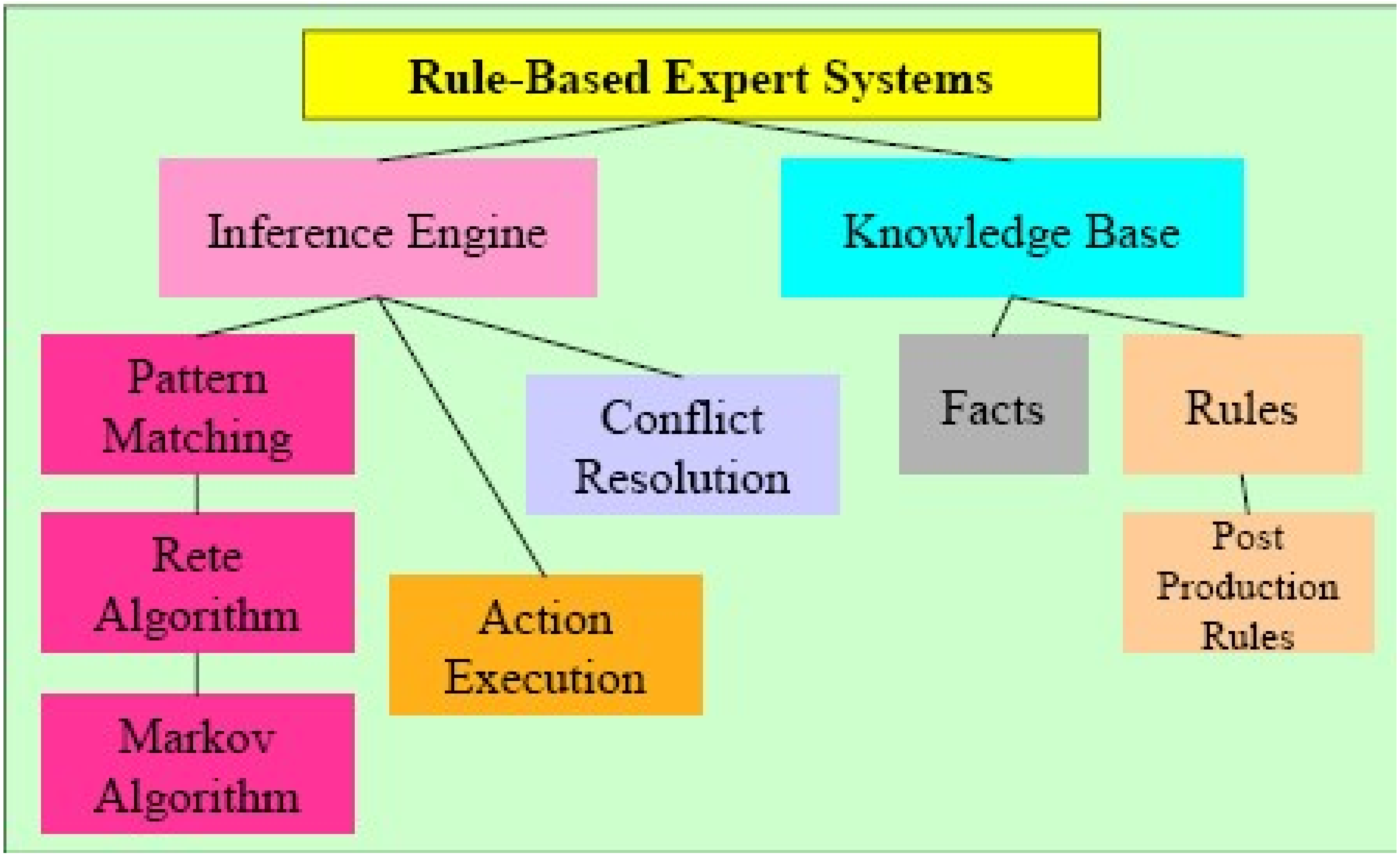
Example Rules



Inference Engine Cycle

- The inference engine determines the execution of the rules by the following cycle:
  - conflict resolution
    - select the rule with the highest priority from the agenda
  - execution (Act)
    - perform the actions on the consequent of the selected rule
    - remove the rule from the agenda
  - match
    - update the agenda
      - add rules whose antecedents are satisfied to the agenda
      - remove rules with non-satisfied agendas
- the cycle ends when no more rules are on the agenda, or when an explicit stop command is encountered

Foundation of Expert Systems



General Methods of Inferencing

- Forward chaining (data-driven)– reasoning from facts to the conclusions resulting from those facts – best for prognosis, monitoring, and control.
  - Examples: CLIPS, OPS5
- Backward chaining (query/Goal driven)– reasoning in reverse from a hypothesis, a potential conclusion to be proved to the facts that support the hypothesis – best for diagnosis problems.
  - Examples: MYCIN

Production Systems

- Rule-based expert systems – most popular type today.
- Knowledge is represented as multiple rules that specify what should/not be concluded from different situations.
- Forward chaining – start w/facts and use rules do draw conclusions/take actions.
- Backward chaining – start w/hypothesis and look for rules that allow hypothesis to be proven true.

Post Production System

- Basic idea – any mathematical / logical system is simply a set of rules specifying how to change one string of symbols into another string of symbols.
    - these rules are also known as rewrite rules
    - simple syntactic string manipulation
    - no understanding or interpretation is required\also used to define grammars of languages
        - e.g BNF grammars of programming languages.
- Basic limitation – lack of control mechanism to guide the application of the rules.

Markov Algorithm

- An ordered group of productions applied in order or priority to an input string.
- If the highest priority rule is not applicable, we apply the next, and so on.
- inefficient algorithm for systems with many rules.
- Termination on (1) last production not applicable to a string, or (2) production ending with period applied
- Can be applied to substrings, beginning at left

Markov Algorithm

(1) $\alpha xy \rightarrow y\alpha x$
(2) $\alpha \rightarrow$ ^
(3) ^ $\rightarrow \alpha$

Rule	Success or Failure	String
1	F	ABC
2	F	ABC
3	S	α ABC
1	S	B α AC
1	S	BC α A
1	F	BC α A
2	S	BCA

Table 1.11 Execution Trace of a Markov Algorithm

Rete Algorithm

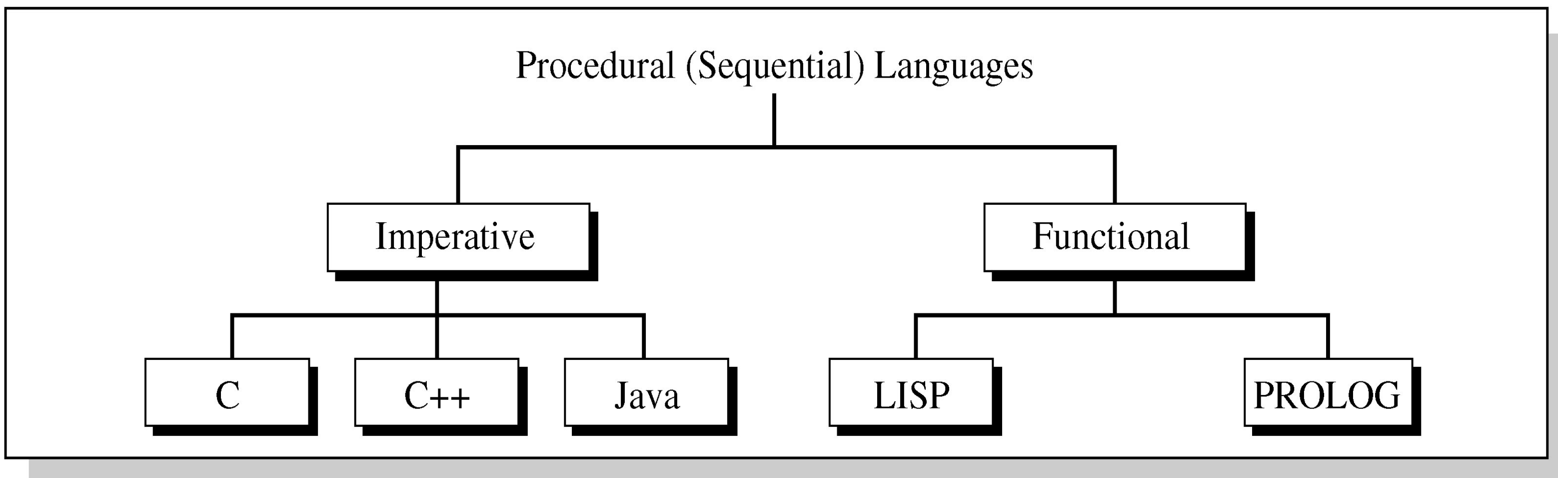
- Markov: too inefficient to be used with many rules
- Functions like a net – holding a lot of information.
- Much faster response times and rule firings can occur compared to a large group of IF-THEN rules which would have to be checked one-by-one in conventional program.
- Takes advantage of temporal redundancy and structural similarity.
- Looks only for changes in matches (ignores static data)
- Drawback is high memory space requirements.

Procedural Paradigms

- Algorithm – method of solving a problem in a finite number of steps.
- Procedural programs are also called sequential programs.
- The programmer specifies exactly how a problem solution must be coded.

Figure 1.8: Procedural Languages

Figure 1.8 Procedural Languages



Imperative Programming

- Also known as statement-oriented
- During execution, program makes transition from the initial state to the final state by passing through series of intermediate states.
- Provide rigid control and top-down-design.
- Not efficient for directly implementing expert systems.

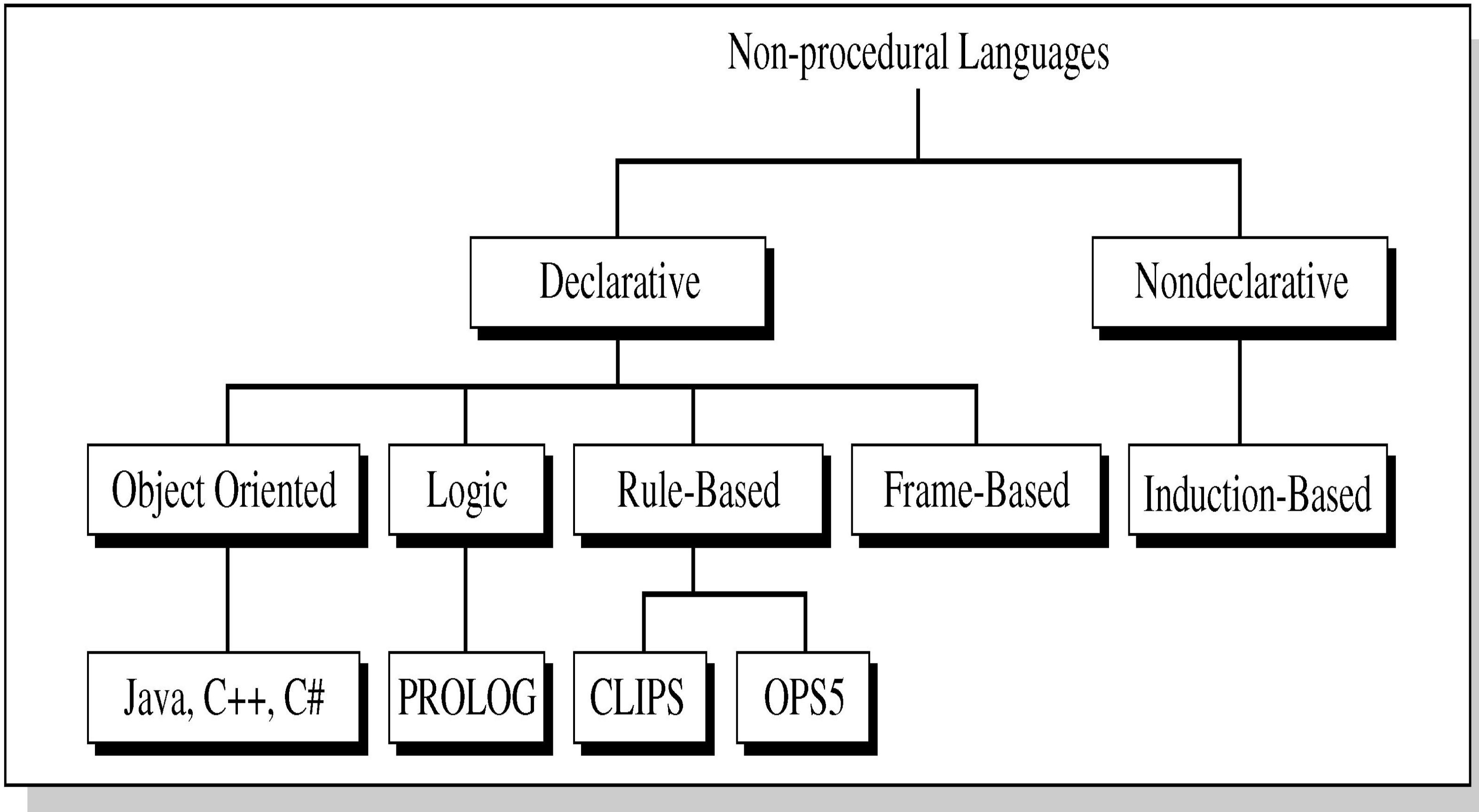
Functional Programming

- Function-based (association, domain, co-domain); f: $S \rightarrow T$
- Not much control
- Bottom-up→ combine simple functions to yield more powerful functions.
- Mathematically a function is an association or rule that maps members of one set, the domain, into another set, the codomain.
- e.g. LISP and Prolog

Nonprocedural Paradigms

- Do not depend on the programmer giving exact details how the program is to be solved.
- Declarative programming – goal is separated from the method to achieve it.
- Object-oriented programming – partly imperative and partly declarative – uses objects and methods that act on those objects.
- Inheritance – (OOP) subclasses derived from parent classes.

Figure 1.9: Nonprocedural Languages



What are Expert Systems?

Can be considered declarative languages:

- Programmer does not specify how to achieve a goal at the algorithm level.
- Induction-based programming – the program learns by generalizing from a sample.

Artificial Neural Systems

In the 1980s, a new development in programming paradigms appeared called artificial neural systems (ANS).

- Based on the way the brain processes information.
- Models solutions by training simulated neurons connected in a network.
- ANS are found in face recognition, medical diagnosis, games, and speech recognition.

ANS Characteristics

- A complex pattern recognition problem – computing the shortest route through a given list of cities.
- ANS is similar to an analog computer using simple processing elements connected in a highly parallel manner.
- Processing elements perform Boolean / arithmetic functions in the inputs
- Key feature is associating weights w/each element.

Table 1.13 Traveling Salesman Problem

Number of Cities	Routes
1	1
2	1–2–1
3	1–2–3–1
	1–3–2–1
4	1–2–3–4–1
	1–2–4–3–1
	1–3–2–4–1
	1–3–4–2–1
	1–4–2–3–1
	1–4–3–2–1

Advantages of ANS

- Storage is fault tolerant
- Quality of stored image degrades gracefully in proportion to the amount of net removed.
- Nets can extrapolate (extend) and interpolate (insert/estimate) from their stored information.
- Nets have plasticity.
- Excellent when functionality is needed long-term w/o repair in hostile environment – low maintenance.

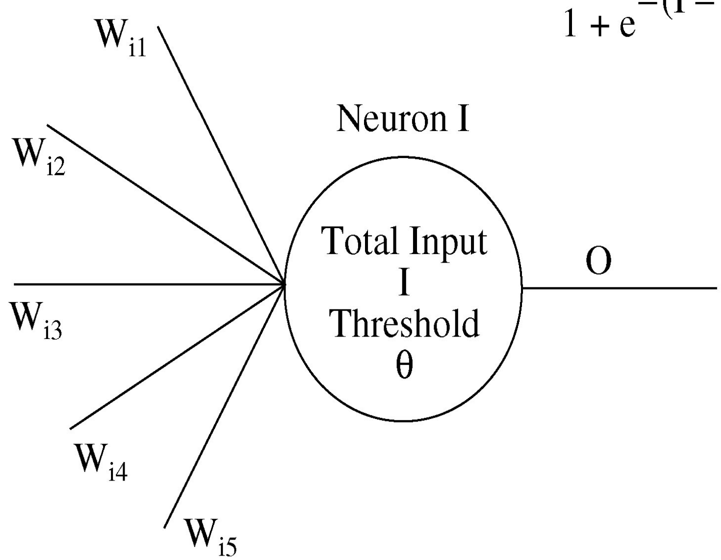
Disadvantage of ANS

- ANS are not well suited for number crunching or problems requiring optimum solution.

Figure 1.10: Neuron Processing Element

$$I = \text{Neuron Input}_i = \sum_j W_{ij} I_j$$

$$O = \text{Neuron Output} = \frac{1}{1 + e^{-(I - \theta)}}$$



Sigmoid Function

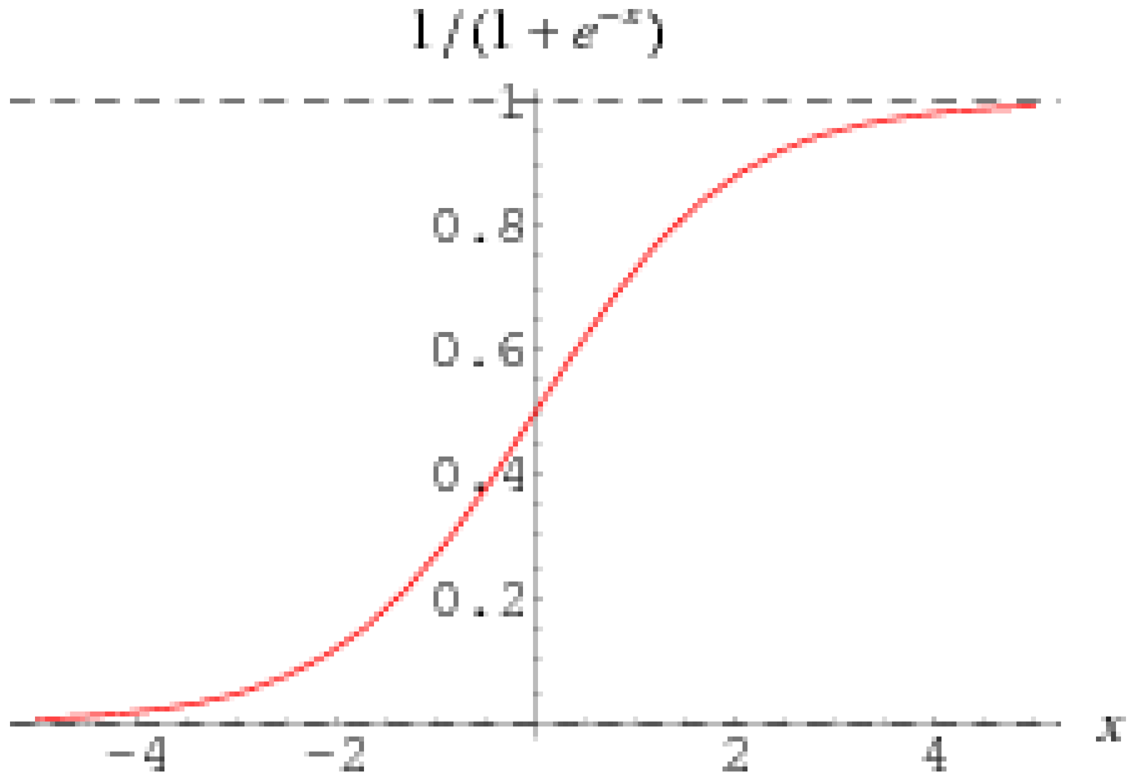


Figure 1.11: A Back-Propagation Net

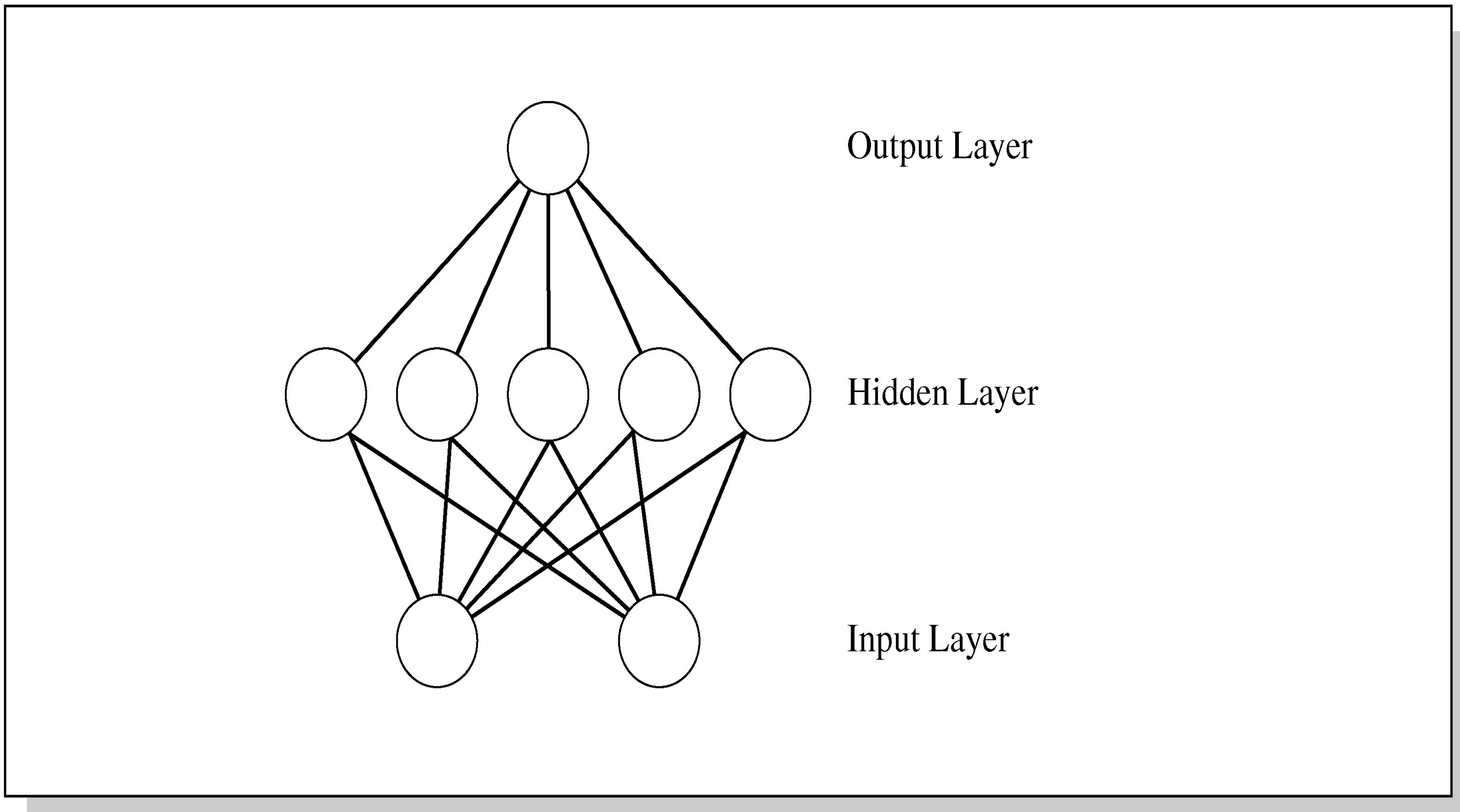
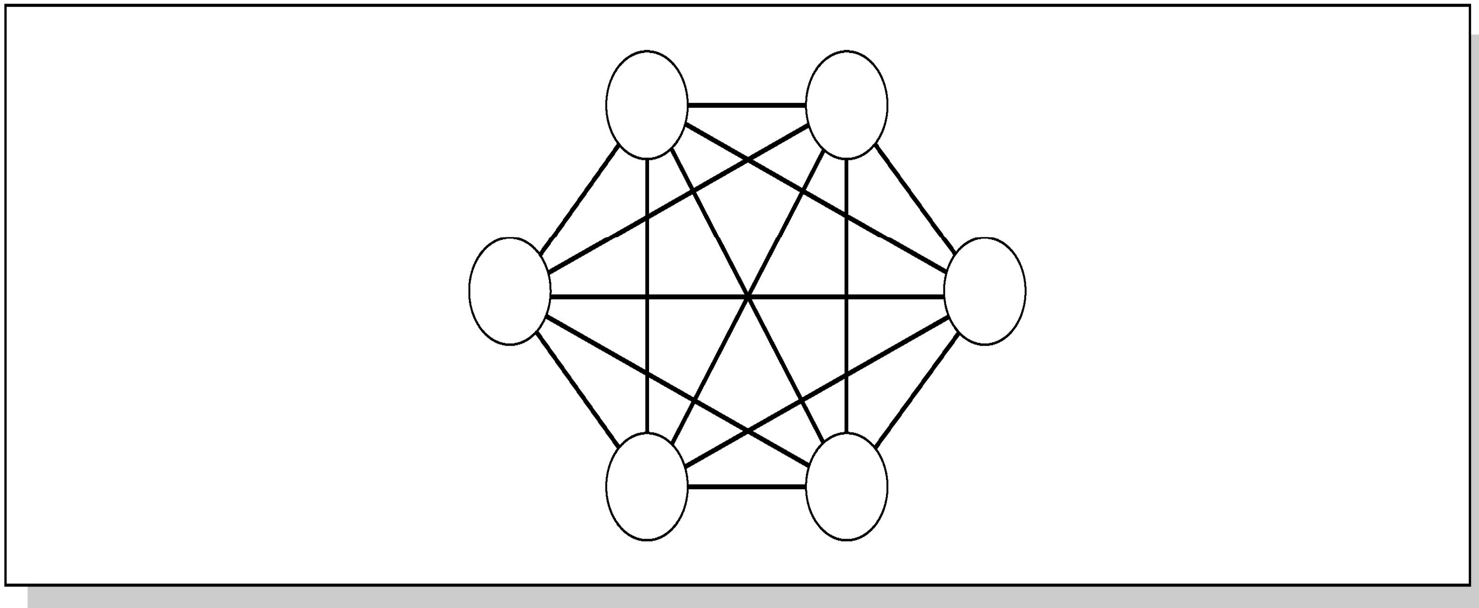


Figure 1.12: Hopfield Artificial Neural Net



MACIE

- An inference engine called MACIE (Matrix Controlled Inference Engine) uses ANS knowledge base.
- Designed to classify disease from symptoms into one of the known diseases the system has been trained on.
- MACIE uses forward chaining to make inferences and backward chaining to query user for additional data to reach conclusions.