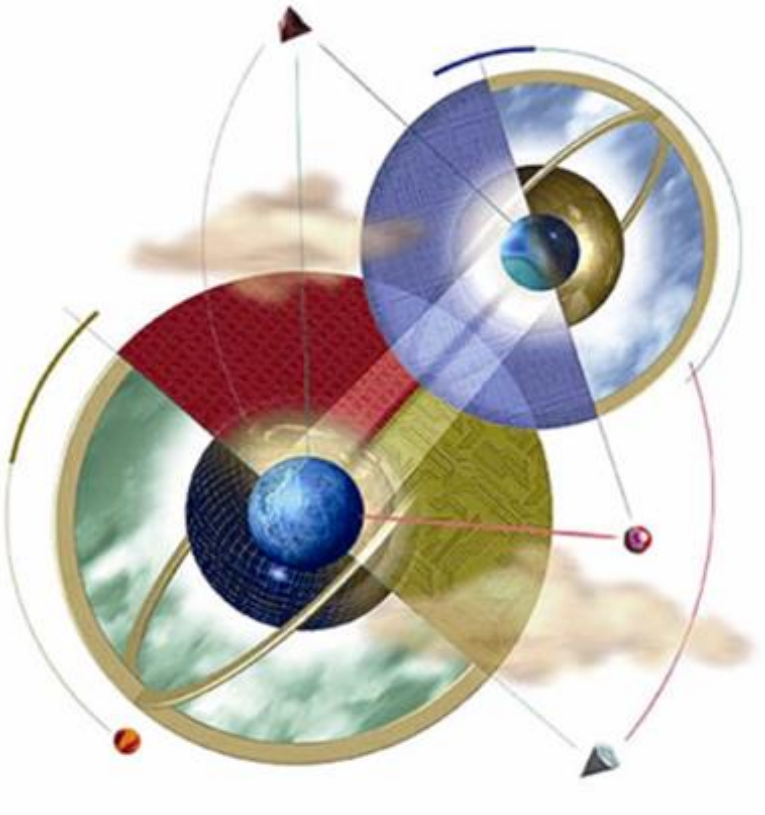


Chapter 3: Methods of Inference

Expert Systems: Principles and Programming, Fourth Edition

Objectives

- Learn the definitions of trees, lattices, and graphs
- Learn about state and problem spaces
- Learn about AND-OR trees and goals
- Explore different methods and rules of inference
- Learn the characteristics of first-order predicate logic and logic systems

Objectives

- Discuss the resolution rule of inference, resolution systems, and deduction
- Compare shallow and causal reasoning
- How to apply resolution to first-order predicate logic
- Learn the meaning of forward and backward chaining

Objectives

- Explore additional methods of inference
- Learn the meaning of Metaknowledge
- Explore the Markov decision process

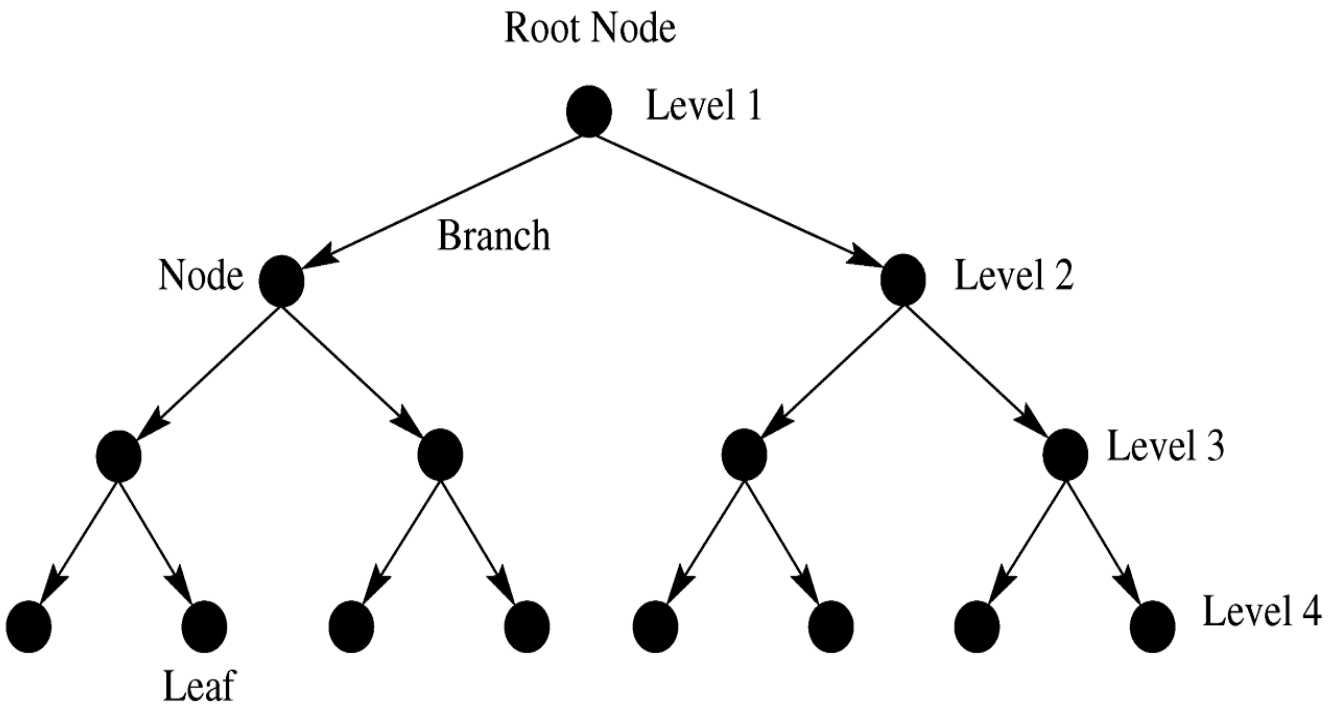
Trees

- A tree is a hierarchical data structure consisting of:
  - Nodes – store information
  - Branches – connect the nodes
- The top node is the root, occupying the highest hierarchy.
- The leaves are at the bottom, occupying the lowest hierarcy.

Trees

- Every node, except the root, has exactly one parent.
- Every node may give rise to zero or more child nodes.
- A binary tree restricts the number of children per node to a maximum of two.
- Degenerate trees have only a single pathway from root to its one leaf.

Figure 3.1 Binary Tree



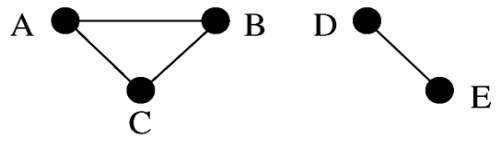
Graphs

- Graphs are sometimes called a network or net.
- A graph can have zero or more links between nodes – there is no distinction between parent and child.
- Sometimes links have weights – weighted graph; or, arrows – directed graph.
- Simple graphs have no loops – links that come back onto the node itself.

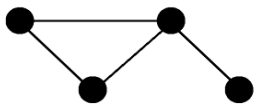
Graphs

- A circuit (cycle) is a path through the graph beginning and ending with the same node.
- Acyclic graphs have no cycles.
- Connected graphs have links to all the nodes.
- Digraphs are graphs with directed links.
- Lattice is a directed acyclic graph.

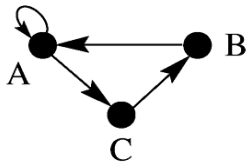
Figure 3.2 Simple Graphs



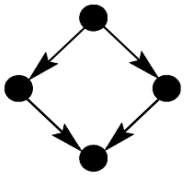
(a) A nonconnected graph



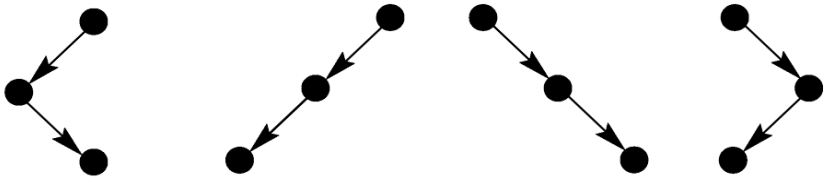
(b) A connected graph



(c) A digraph with a self-loop and circuit



(d) A lattice



(e) Degenerate binary trees of three nodes

Making Decisions

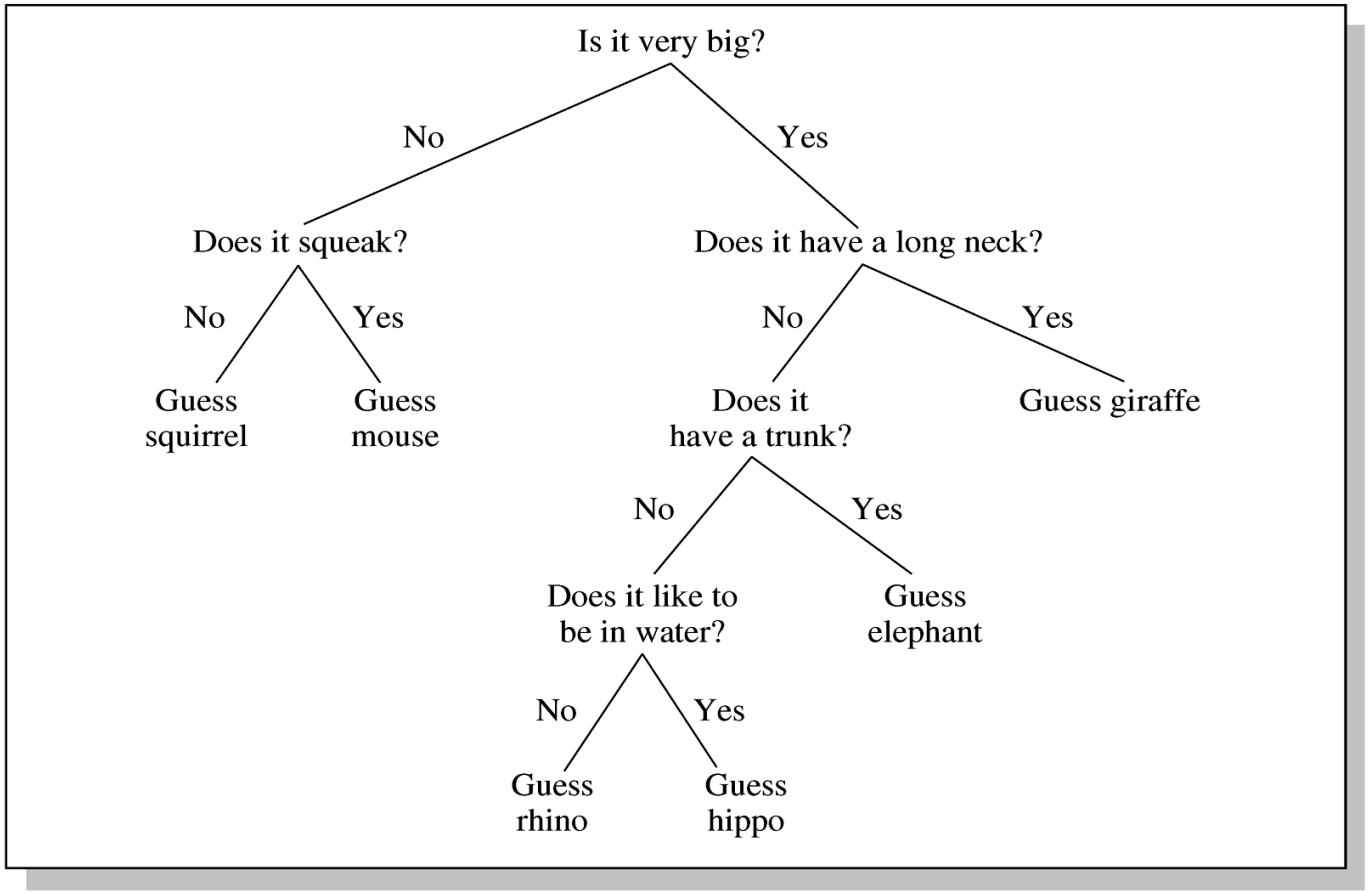
- Trees / lattices are useful for classifying objects in a hierarchical nature.
- Trees / lattices are useful for making decisions.
- We refer to trees / lattices as structures.
- Decision trees are useful for representing and reasoning about knowledge.

Binary Decision Trees

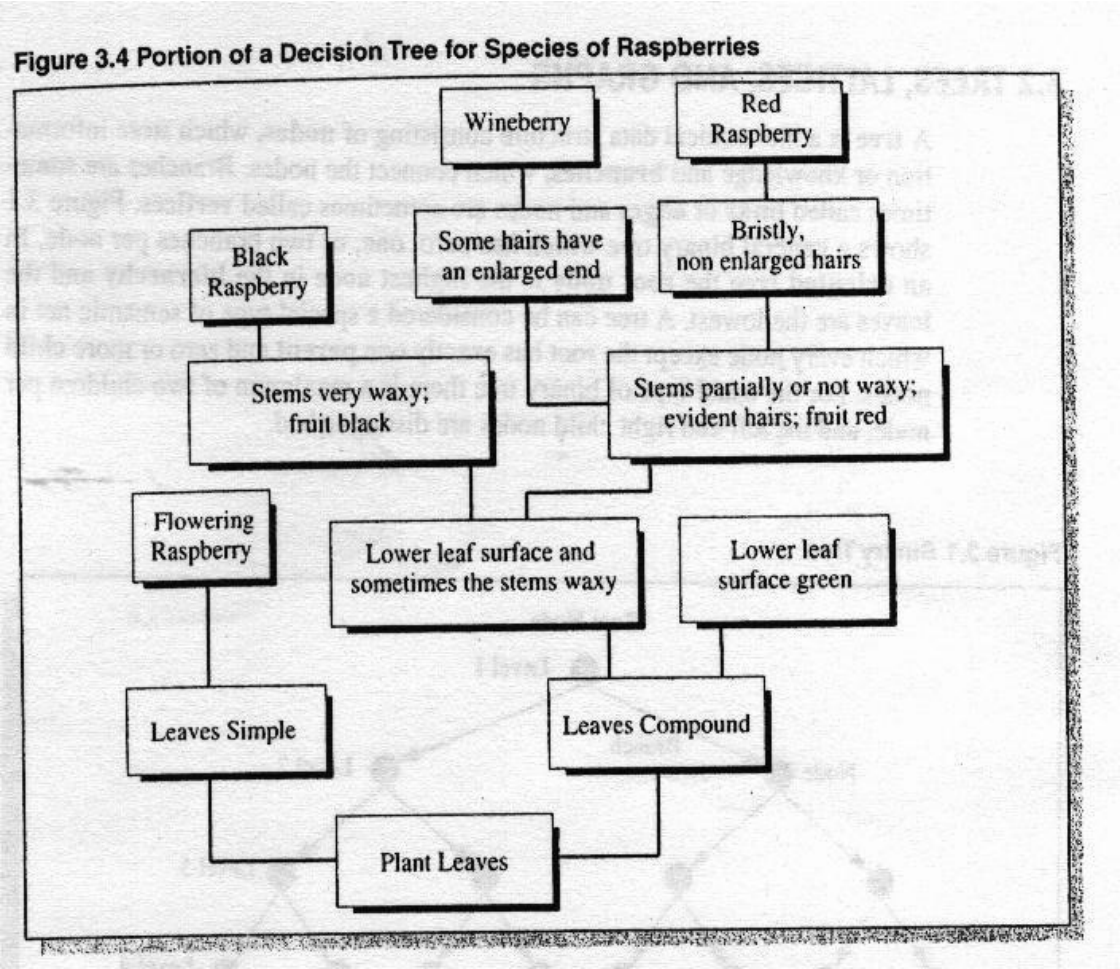
- Every question takes us down one level in the tree.
- A binary decision tree having N nodes:
  - All leaves will be answers.
  - All internal nodes are questions.
  - There will be a maximum of 2^N answers for N questions.
- Decision trees can be self learning.
- Decision trees can be translated into production rules.

Decision Tree Example

Figure 3.3 Decision Tree Showing Knowledge About Animals



Decision Tree Example



State and Problem Spaces

- A state space can be used to define an object's behavior.
- Different states refer to characteristics that define the status of the object.
- A state space shows the transitions an object can make in going from one state to another.

Finite State Machine

- A FSM is a diagram describing the finite number of states of a machine.
- At any one time, the machine is in one particular state.
- The machine accepts input and progresses to the next state.
- FSMs are often used in compilers and validity checking programs.

Using FSM to Solve Problems

- Characterizing ill-structured problems – one having uncertainties.
- Well-formed problems:
  - Explicit problem, goal, and operations are known
  - Deterministic – we are sure of the next state when an operator is applied to a state.
  - The problem space is bounded.
  - The states are discrete.

Figure 3.5 State Diagram for a Soft Drink Vending Machine Accepting Quarters (Q) and Nickels (N)

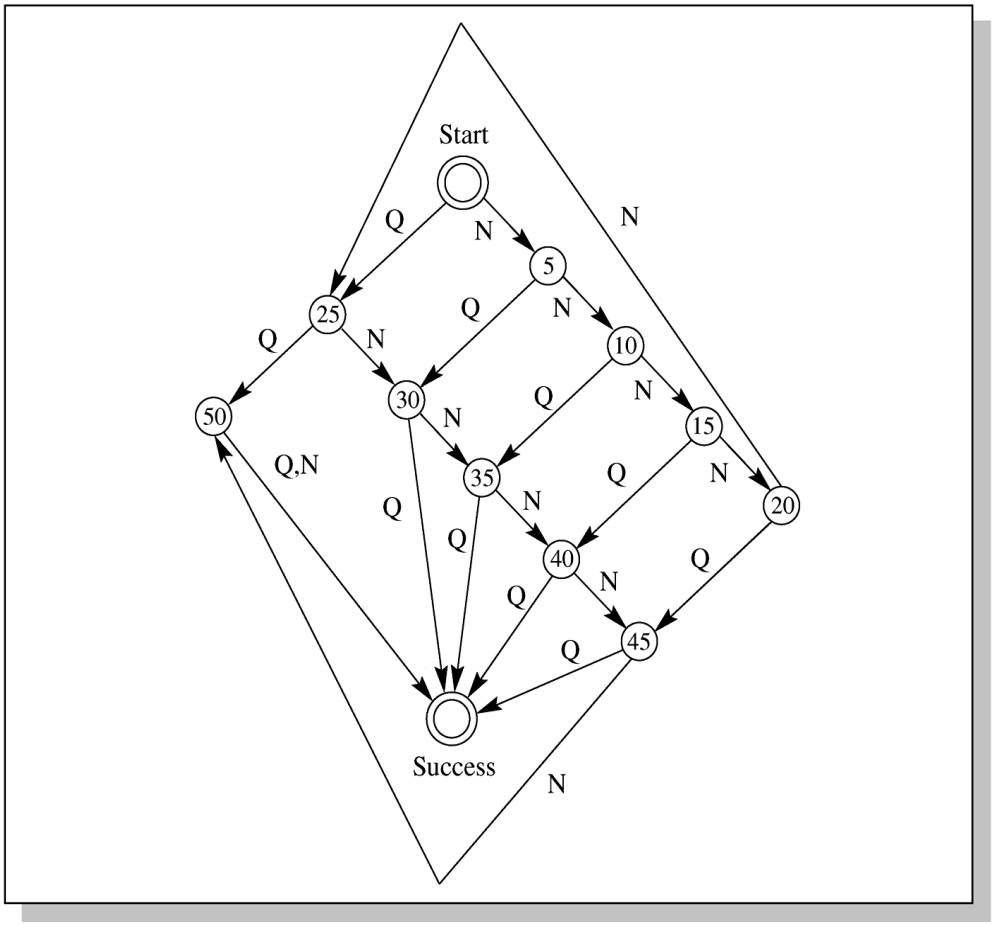


Figure 3.6 Part of a Finite State Machine for Determining Valid Strings WHILE, WRITE, and BEGIN.

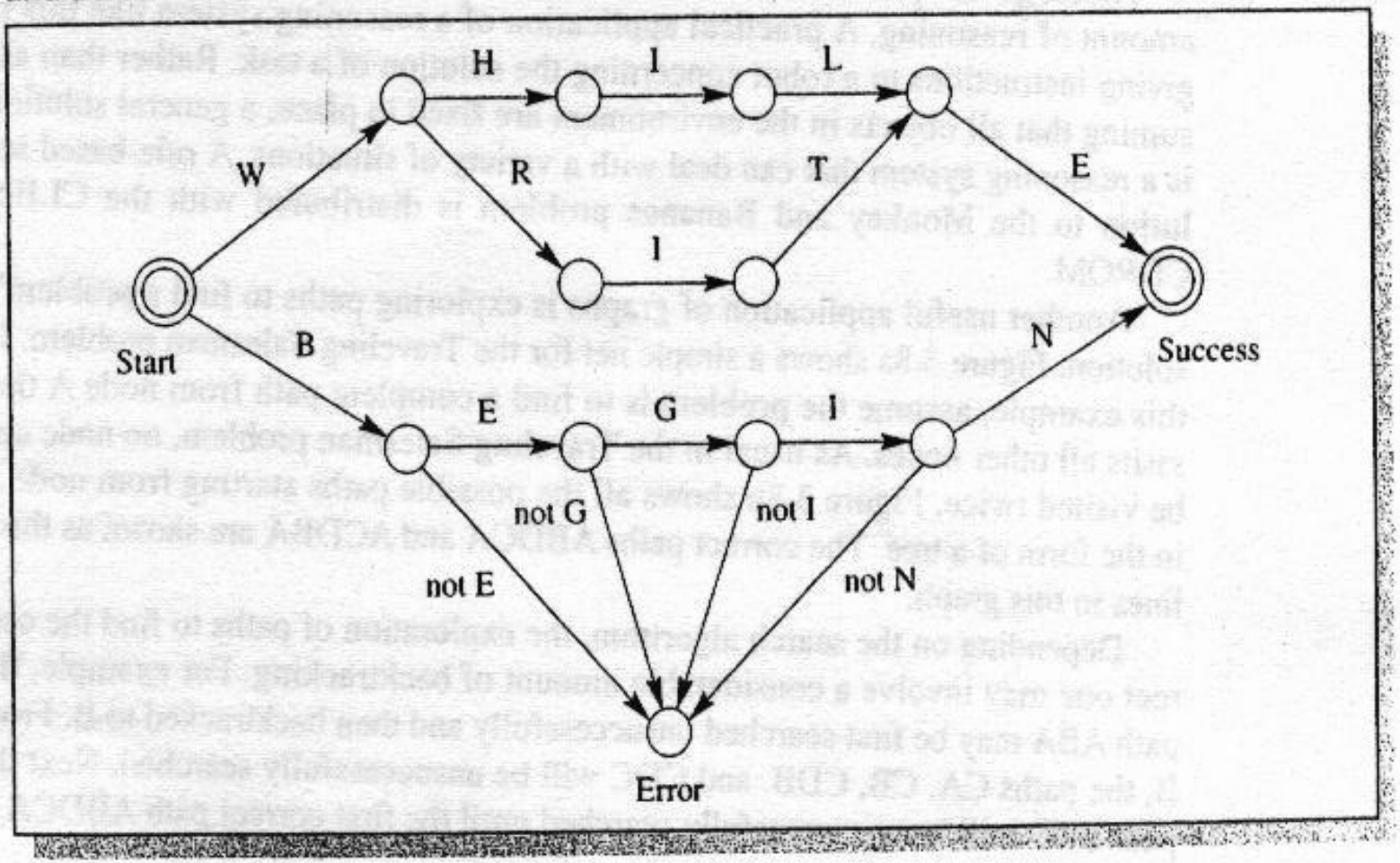
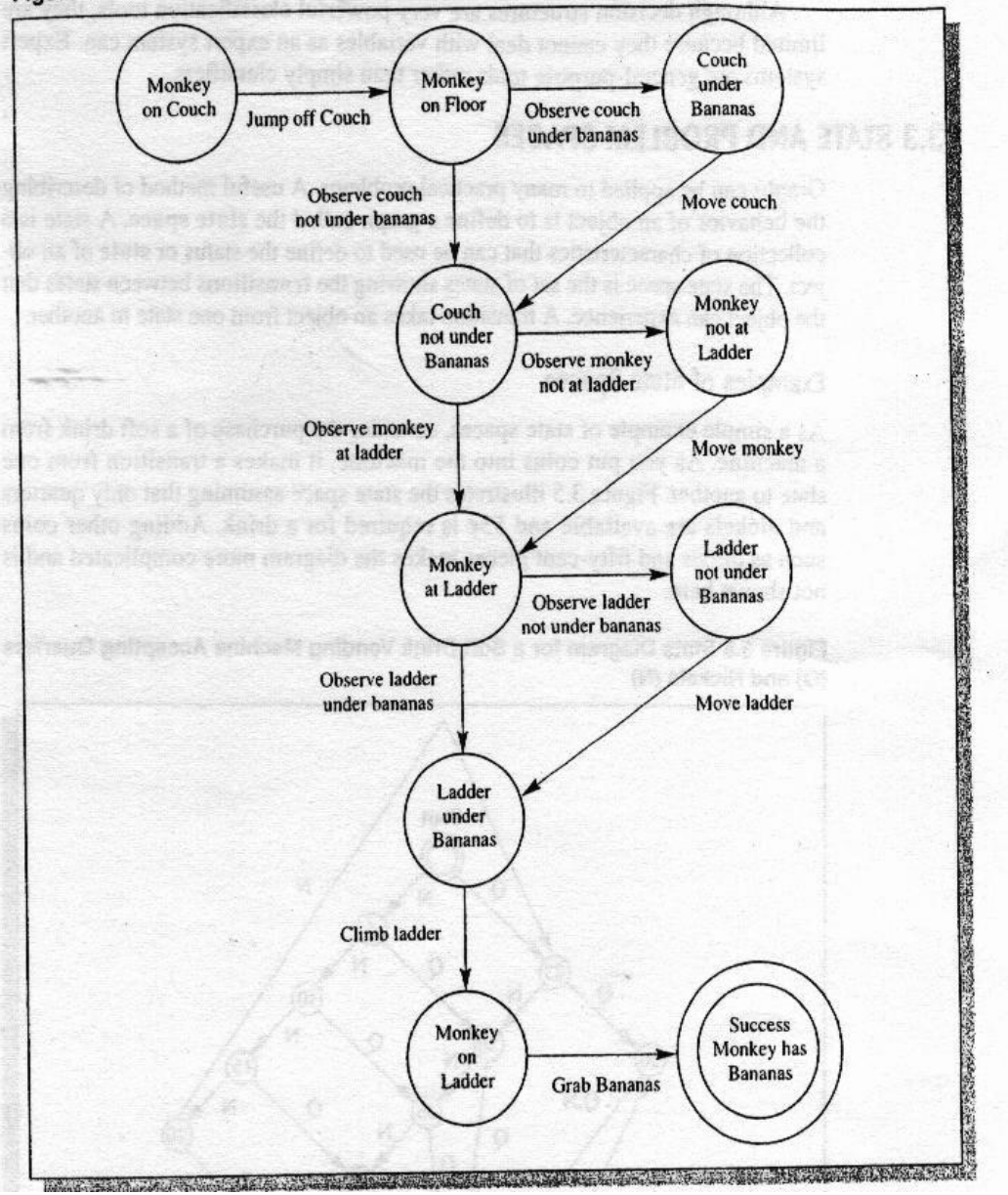


Figure 3.7 The State Space for the Monkey and Bananas Problem

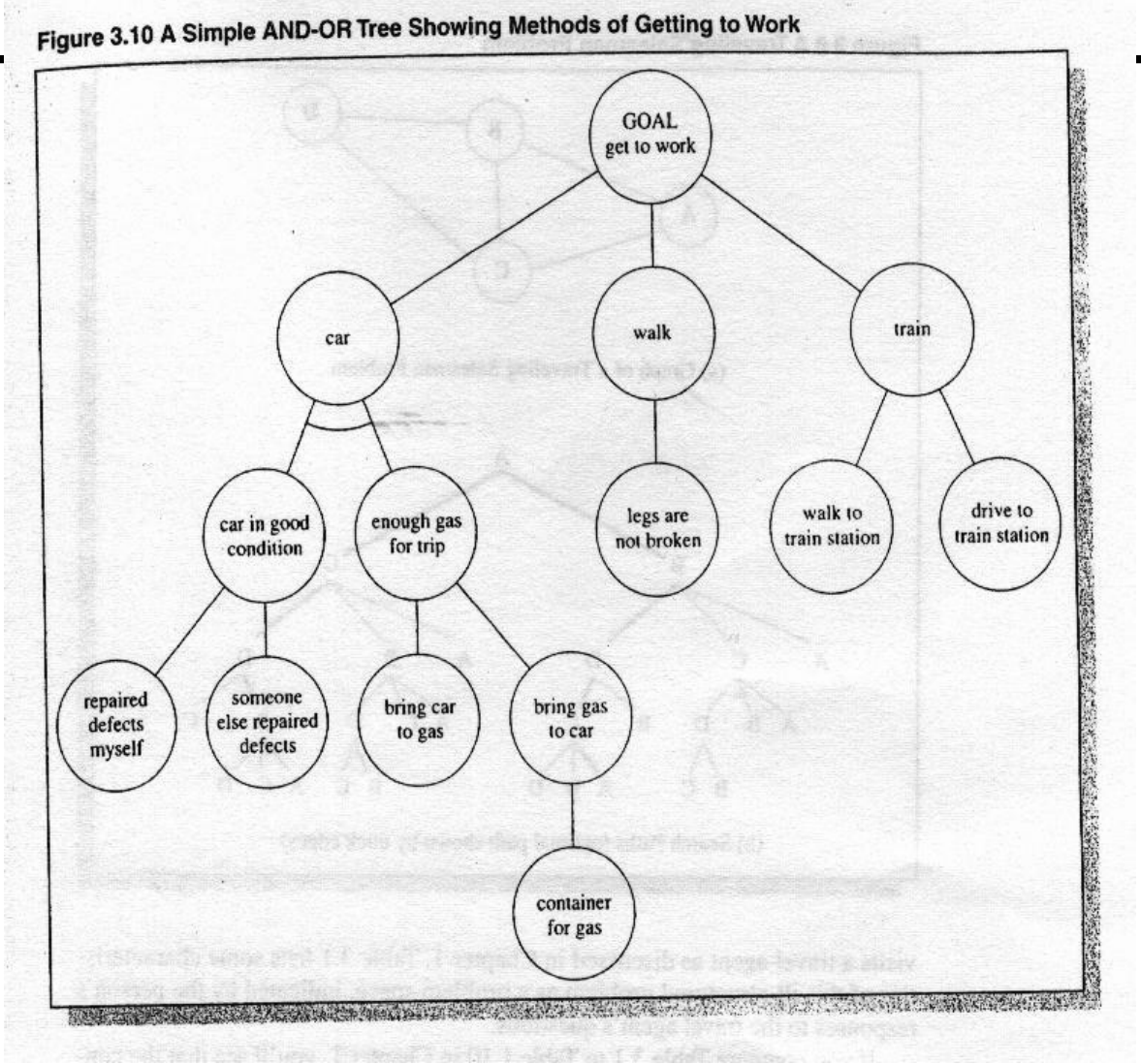


AND-OR Trees and Goals

- 1990s, PROLOG was used for commercial applications in business and industry.
- PROLOG uses backward chaining to divide problems into smaller problems and then solves them.
- AND-OR trees also use backward chaining.
- AND-OR-NOT lattices use logic gates to describe problems.

AND-OR Trees and Goals

Figure 3.10 A Simple AND-OR Tree Showing Methods of Getting to Work



Types of Logic

- Deduction – reasoning where conclusions must follow from premises
- Induction – inference is from the specific case to the general
- Analogy – inferring conclusions based on similarities with other situations
- Abduction – reasoning back from a true condition to the premises that may have caused the condition

Types of Logic

- Default – absence of specific knowledge
- Autoepistemic – self-knowledge

- Intuition – no proven theory
- Heuristics – rules of thumb based on experience
- Generate and test – trial and error

Deductive Logic

- Argument – group of statements where the last is justified on the basis of the previous ones
- Deductive logic can determine the validity of an argument.
- Syllogism – has two premises and one conclusion
- Deductive argument – conclusions reached by following true premises must themselves be true

Syllogisms vs. Rules

- Syllogism:
  - All basketball players are tall.
  - Jason is a basketball player.
  - ⑧ Jason is tall.

- IF-THEN rule:

  IF All basketball players are tall and
  Jason is a basketball player

  THEN Jason is tall.

Figure 3.21 Causal Forward Chaining