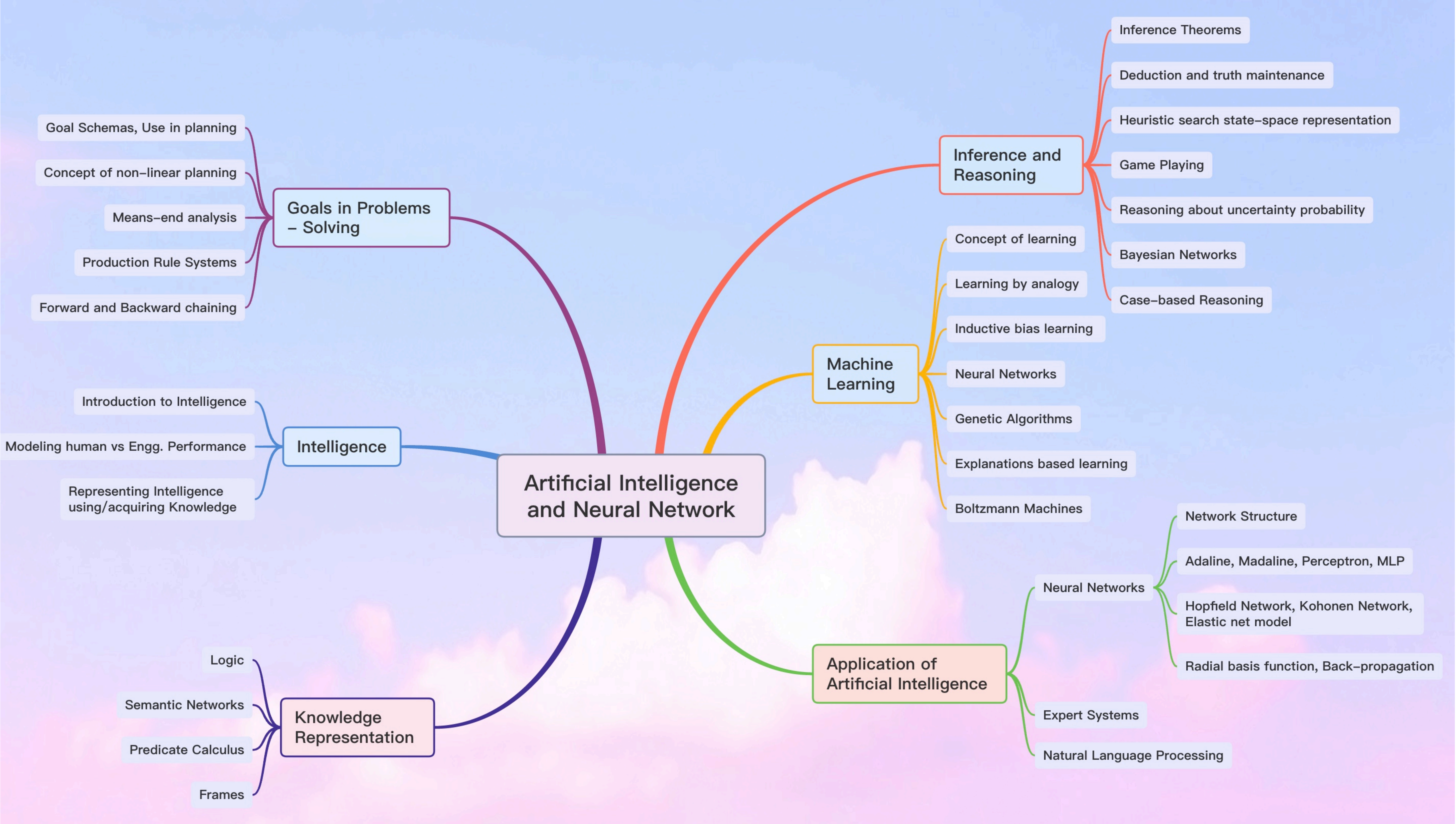
UNIT 4

Inference and Reasoning


Artificial Intelligence and Neural Network
Goals in Problems – Solving
Goal Schemas, Use in planning
Concept of non-linear planning
Means-end analysis
Production Rule Systems
Forward and Backward chaining
Intelligence
Introduction to Intelligence
Modeling human vs Engg. Performance
Representing Intelligence using/acquiring Knowledge
Knowledge Representation
Logic
Semantic Networks
Predicate Calculus
Frames
Inference and Reasoning
Inference Theorems
Deduction and truth maintenance
Heuristic search state-space representation
Game Playing
Reasoning about uncertainty probability
Bayesian Networks
Case-based Reasoning
Machine Learning
Concept of learning
Learning by analogy
Inductive bias learning
Neural Networks
Genetic Algorithms
Explanations based learning
Boltzmann Machines
Application of Artificial Intelligence
Neural Networks
Network Structure
Adaline, Madaline, Perceptron, MLP
Hopfield Network, Kohonen Network, Elastic net model
Radial basis function, Back-propagation
Expert Systems
Natural Language Processing

Contents

Heuristic Search State Space Representation

- Strategy of problem solving where problem specific knowledge is known along with problem definition
- These search find solutions more efficiently by the use of heuristics
- Heuristic is a search technique that improves the efficiency of the search process
- By eliminating the unpromising states and their descendants from consideration, heuristic algorithms can find acceptable solutions

Heuristic Search State Space Representation

- Heuristics are fallible i.e. they are likely to make mistakes as well
- It is the approach following an informed guess of next step to be taken
- It is often based on experience or intuition
- Heuristic have limited information and hence can lead to suboptimal solution or even fail to find any solution at all

Heuristic Search State Space Representation

- **State Space:** where each state corresponds to a stable situation
  - Initial State
  - Rules for transition from one state to another
  - Final State → Ultimate Goal
- **State Space Representation** → forms the basis for AI methods
- **State Space Structure corresponds to the structure of problem solving in two ways**
  - It allows for a formal definition of the problem
  - It permits to define the process of solving the particular problem as a combination of known techniques and searching mechanism

Heuristic Search State Space Representation

A restricted state-transition system is a triple
$\Sigma=(S, A,\gamma)$, where:

- $S=\{s1,s2,\ldots\}$ is a set of states;
- $A=\{a1,a2,\ldots\}$ is a set of actions;
- $\gamma:S\times A\rightarrow S$ is a state transition function.

Heuristic Search State Space Representation

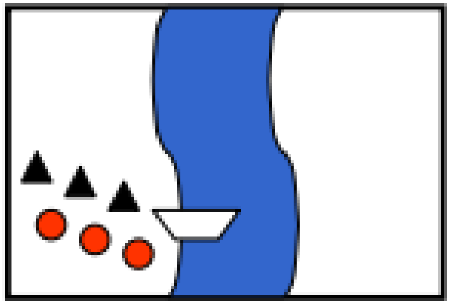
Search Problems:

- initial state
- set of possible actions/applicability conditions
  - successor function: *state* → set of <*action*, *state*>
  - successor function + initial state = state space
  - path (solution)
- goal
  - goal state or goal test function
- path cost function
  - for optimality
  - assumption: path cost = sum of step costs

Heuristic Search State Space Representation

Missionaries and Cannibals: Initial State and Actions

- initial state:
  - all missionaries, all cannibals, and the boat are on the left bank



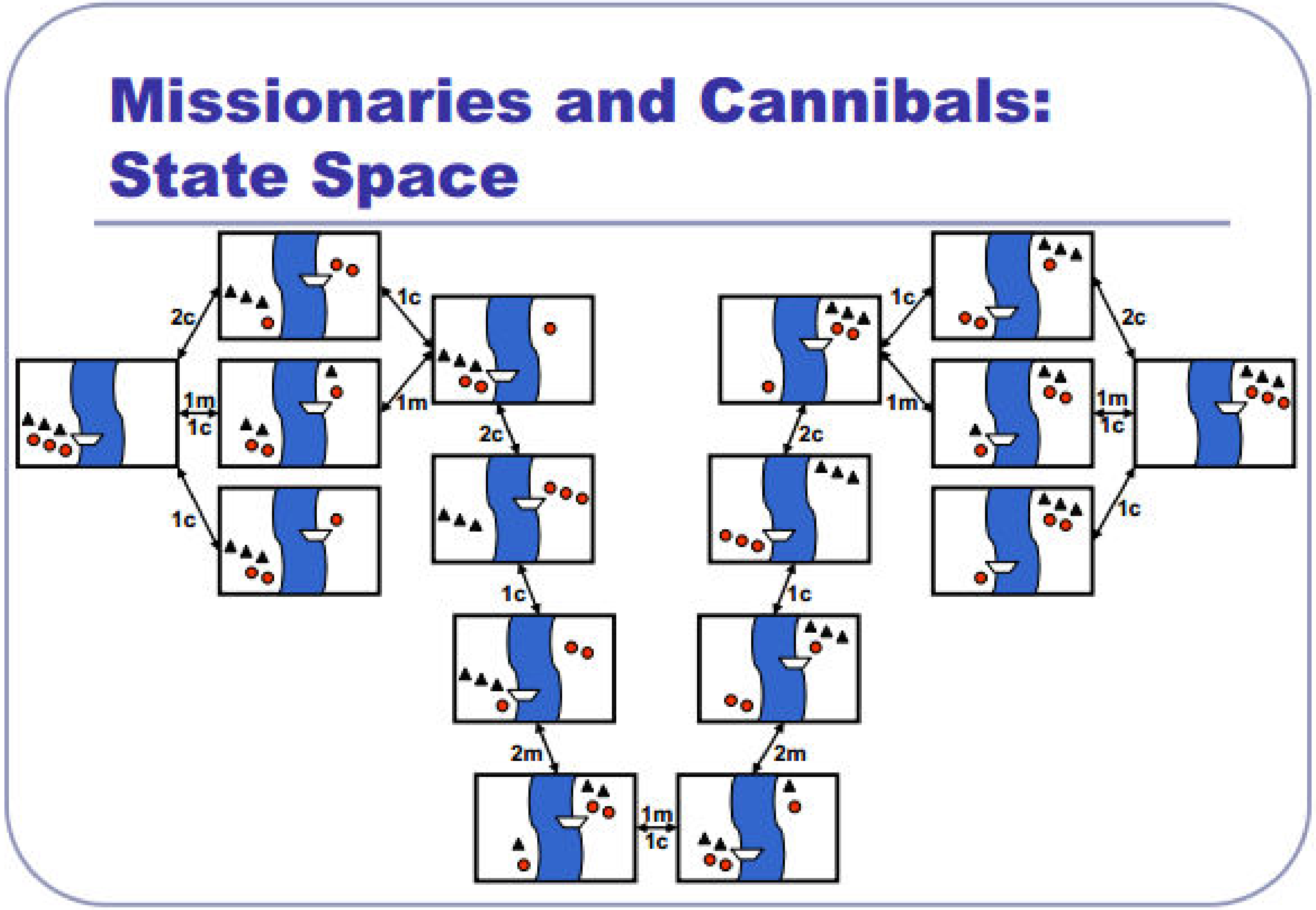
- 5 possible actions:
  - one missionary crossing
  - one cannibal crossing
  - two missionaries crossing
  - two cannibals crossing
  - one missionary and one cannibal crossing

Heuristic Search State Space Representation

Missionaries and Cannibals: Successor Function

state	set of <*action*, *state*>
(L:3m,3c,b-R:0m,0c) →	{<2c, (L:3m,1c-R:0m,2c,b)>, <1m1c, (L:2m,2c-R:1m,1c,b)>, <1c, (L:3m,2c-R:0m,1c,b)>}
(L:3m,1c-R:0m,2c,b) →	{<2c, (L:3m,3c,b-R:0m,0c)>, <1c, (L:3m,2c,b-R:0m,1c)>}
(L:2m,2c-R:1m,1c,b) →	{<1m1c, (L:3m,3c,b-R:0m,0c)>, <1m, (L:3m,2c,b-R:0m,1c)>}
⋮	⋮

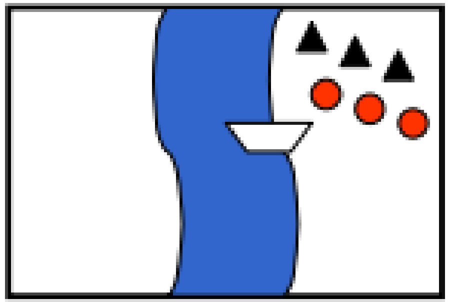
Heuristic Search State Space Representation



Heuristic Search State Space Representation

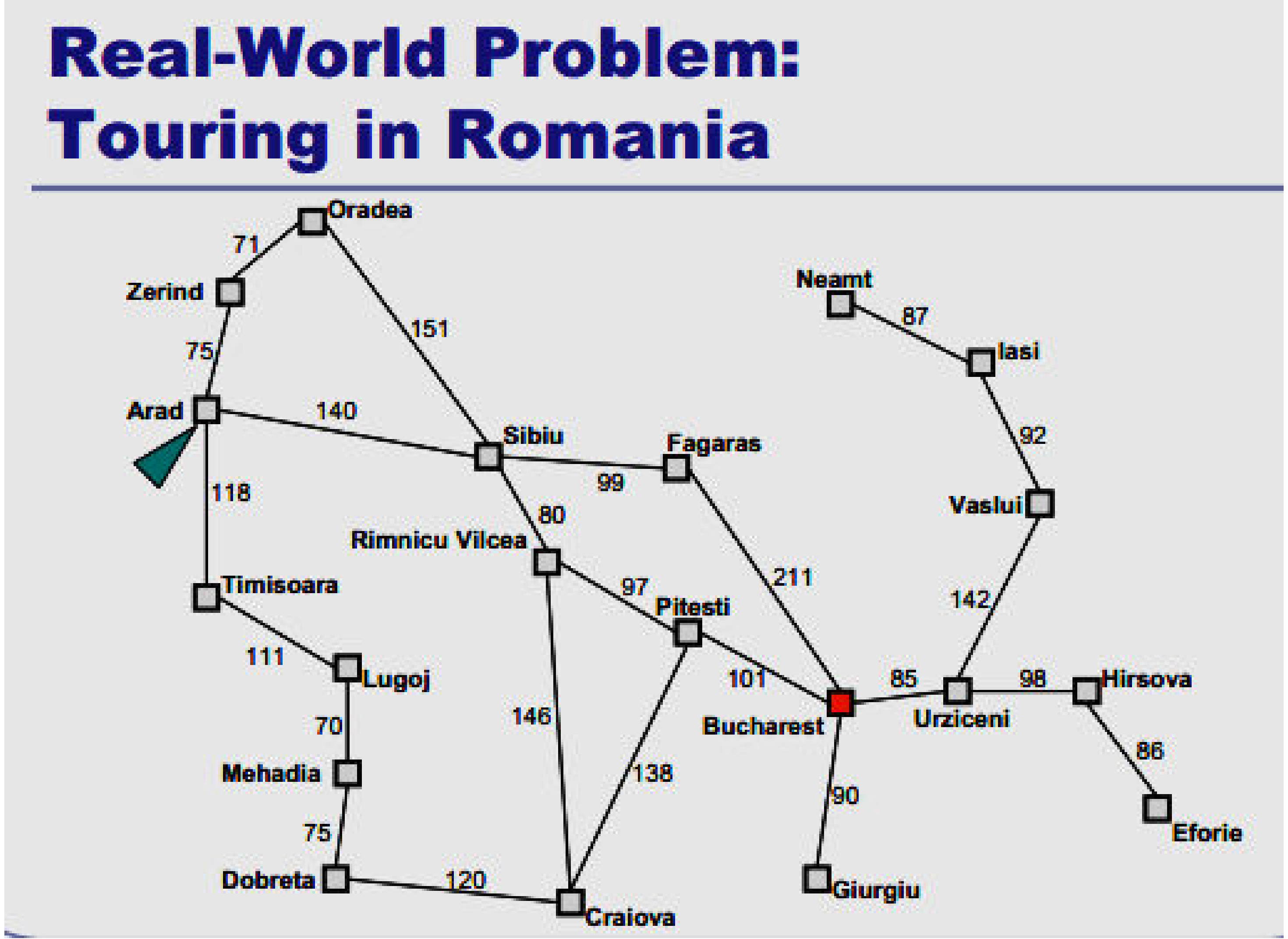
Missionaries and Cannibals: Goal State and Path Cost

- goal state:
  - all missionaries, all cannibals, and the boat are on the right bank



- path cost
  - step cost: 1 for each crossing
  - path cost: number of crossings = length of path
- solution path:
  - 4 optimal solutions
  - cost: 11

Heuristic Search State Space Representation



Heuristic Search State Space Representation

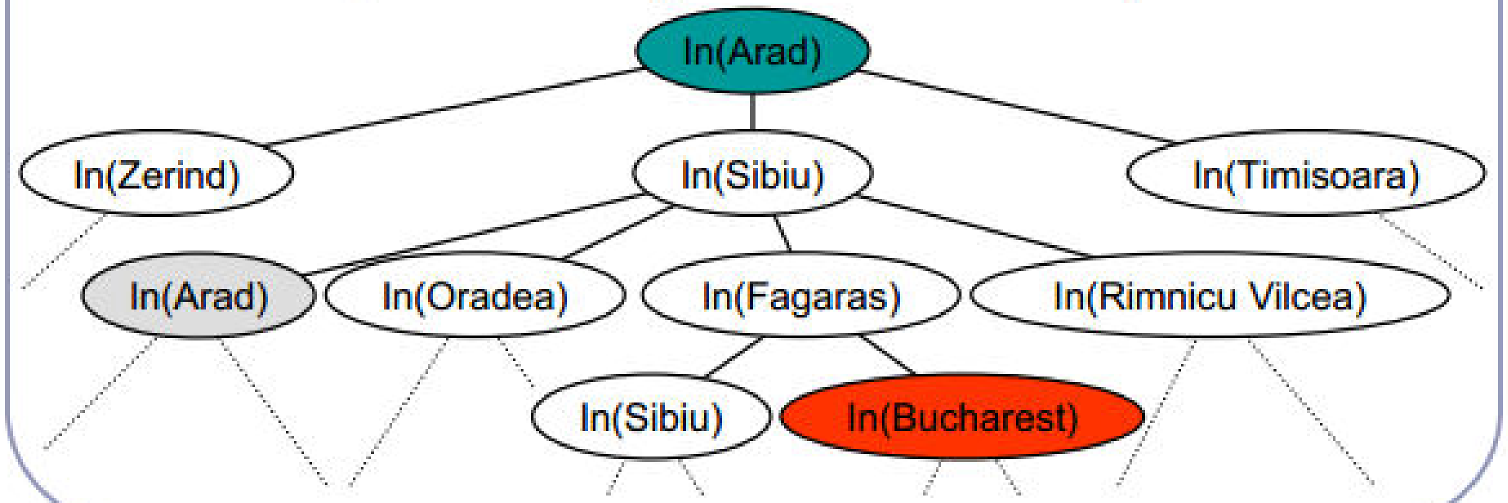
Touring Romania: Search Problem Definition

- initial state:
  - In(Arad)
- possible Actions:
  - DriveTo(Zerind), DriveTo(Sibiu), DriveTo(Timisoara), etc.
- goal state:
  - In(Bucharest)
- step cost:
  - distances between cities

Heuristic Search State Space Representation

Search Trees

- search tree: tree structure defined by initial state and successor function
- Touring Romania (partial search tree):



Heuristic Search State Space Representation

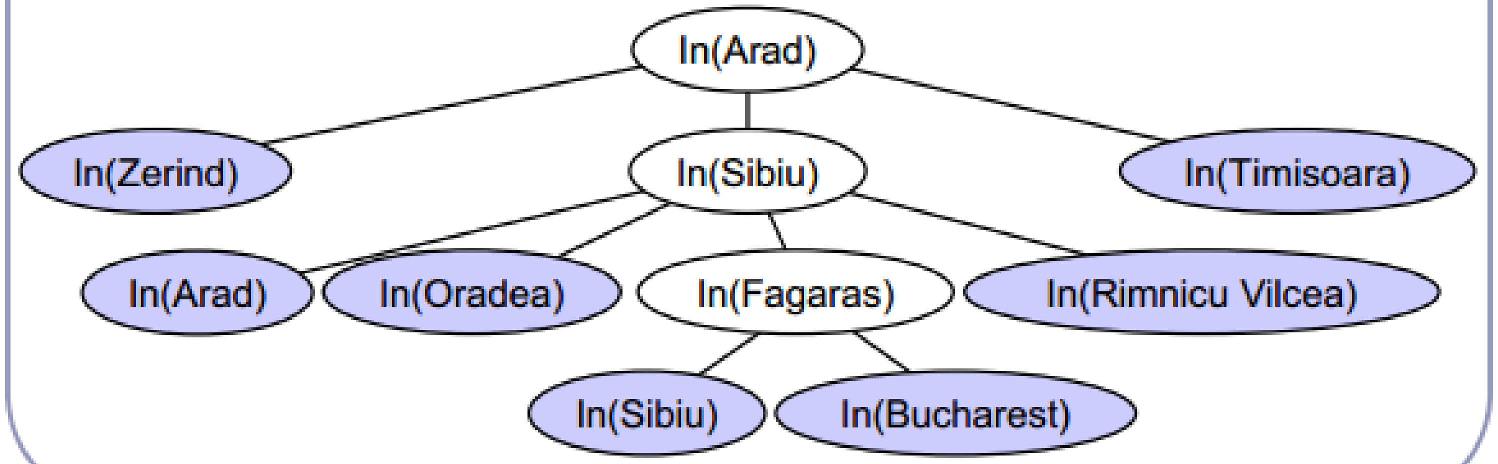
Search Nodes

- search nodes: the nodes in the search tree
- data structure:
  - *state*: a state in the state space
  - *parent node*: the immediate predecessor in the search tree
  - *action*: the action that, performed in the parent node's state, leads to this node's state
  - *path cost*: the total cost of the path leading to this node
  - *depth*: the depth of this node in the search tree

Heuristic Search State Space Representation

Fringe Nodes in Touring Romania Example

fringe nodes: nodes that have not been expanded



Heuristic Search State Space Representation

Search (Control) Strategy

- search or control strategy: an effective method for scheduling the application of the successor function to expand nodes
  - selects the next node to be expanded from the fringe
  - determines the order in which nodes are expanded
  - aim: produce a goal state as quickly as possible
- examples:
  - LIFO/FIFO-queue for fringe nodes
  - alphabetical ordering

Heuristic Search State Space Representation

General Tree Search Algorithm

function treeSearch(*problem*, *strategy*)
    fringe ← { **new**
            searchNode(*problem*.initialState) }
    loop
        if empty(*fringe*) **then return** failure
        node ← selectFrom(*fringe*, *strategy*)
        if *problem*.goalTest(*node.state*) **then**
            return pathTo(*node*)
        fringe ← *fringe* + expand(*problem*, *node*)

Heuristic Search State Space Representation

Uninformed vs. Informed Search

- uninformed search (blind search)
  - no additional information about states beyond problem definition
  - only goal states and non-goal states can be distinguished
- informed search (heuristic search)
  - additional information about how "promising" a state is available

Heuristic Search State Space Representation

Best-First Search

- an instance of the general tree search or graph search algorithm
  - strategy: select next node based on an $\boxed{\text{evaluation function } f\text{: state space} \rightarrow \mathbb{R}}$
  - select node with lowest value $f(n)$
- implementation:
  selectFrom(*fringe*, *strategy*)
  - priority queue: maintains fringe in ascending order of f-values

Heuristic Search State Space Representation

Heuristic Functions

- heuristic function h: state space $\rightarrow \mathbb{R}$
- $h(n)$ = estimated cost of the cheapest path from node n to a goal node
- if n is a goal node then $h(n)$ must be 0
- heuristic function encodes problem-specific knowledge in a problem-independent way

Heuristic Search State Space Representation

Greedy Best-First Search

- use heuristic function as evaluation function: $f(n) = h(n)$
  - always expands the node that is closest to the goal node
  - eats the largest chunk out of the remaining distance, hence, “greedy”

Heuristic Search State Space Representation

Touring in Romania: Heuristic

- $h_{SLD}(n)$ = straight-line distance to Bucharest

Arad	366	Hirsova	151	Rimnicu Vilcea	193
Bucharest	0	Iasi	226		
Craiova	160	Lugoj	244	Sibiu	253
Dobreta	242	Mehadia	241	Timisoara	329
Eforie	161	Neamt	234	Urziceni	80
Fagaras	176	Oradea	380	Vaslui	199
Giurgiu	77	Pitesti	100	Zerind	374

Heuristic Search State Space Representation

A* Search

- best-first search where

$$f(n) = h(n) + g(n)$$

  - $h(n)$ the heuristic function (as before)
  - $g(n)$ the cost to reach the node n

- evaluation function:

  $f(n)$ = estimated cost of the cheapest solution through n

- A* search is optimal if $h(n)$ is <u>admissible</u>

Heuristic Search State Space Representation

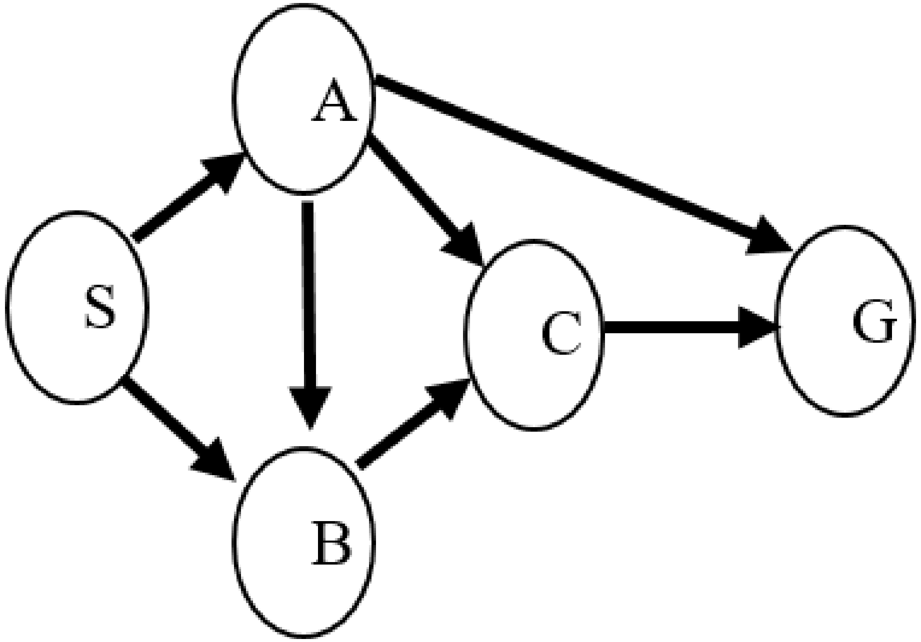
Admissible Heuristics

A heuristic $h(n)$ is admissible if it *never overestimates* the distance from n to the nearest goal node.

- example: h_{SLD}
- A* search: If $h(n)$ is admissible then $f(n)$ never overestimates the true cost of a solution through n.

A* Search

- Example…



	S	A	B	C	G
S		1	4		
A			2	5	12
B				2	
C					3
G					

State	H(n)
S	7
A	6
B	2
C	1
G	0

Game Playing

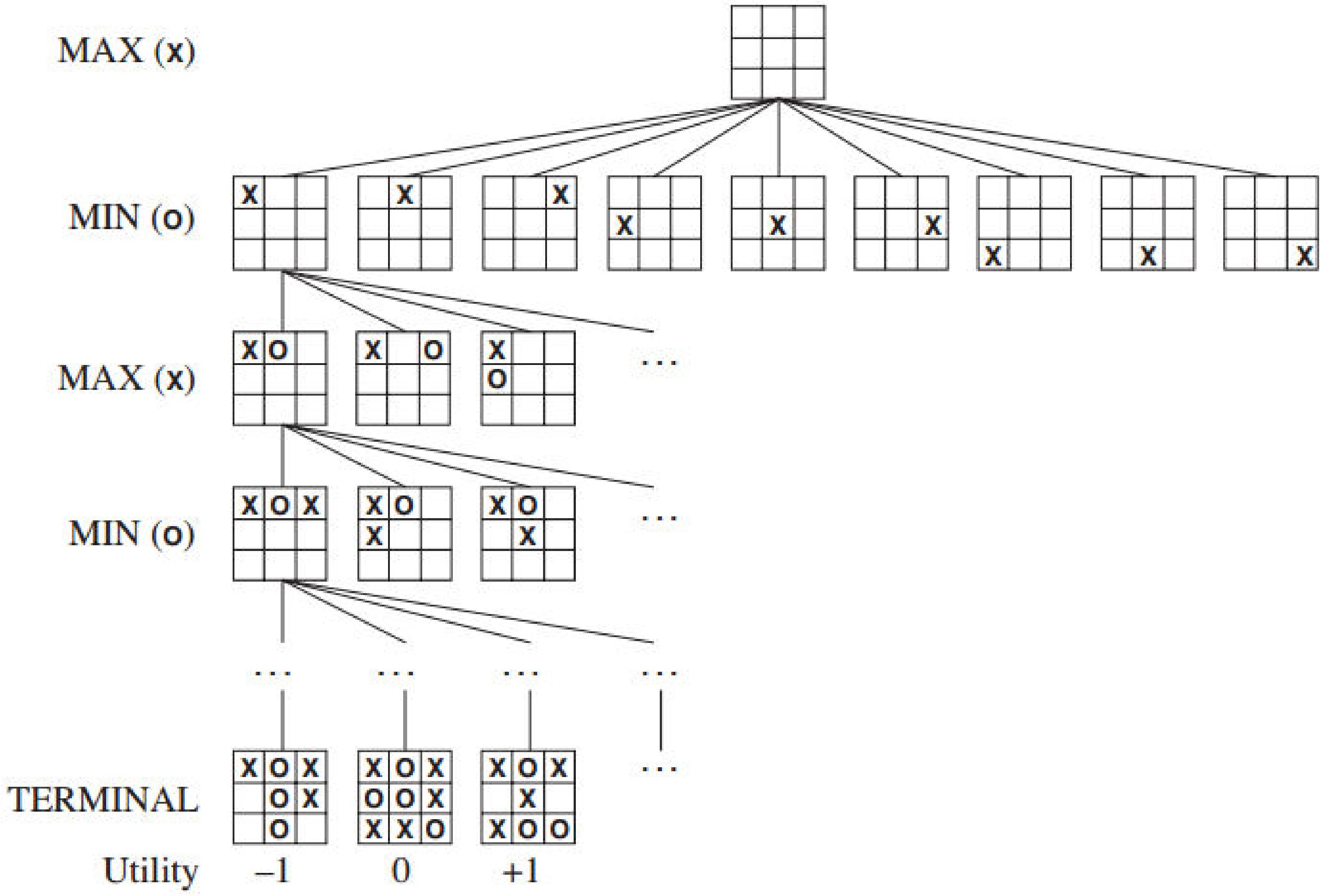
- Major Topic in AI since very beginning
- Closely related to "Intelligence", well defined states and Rules
- Search → Most common AI technique in Game
- In some other problem-solving activities, state change is solely caused by the action of the system itself
- In multi-player games, states also depend on the actions of other players (systems) who usually have different goals

Game Playing

Min-Max Algorithm

- Max is considered as the first player in the game and Min as the second player
- This algorithm computes the minimax decision from the current state
- It uses a recursive computation of minimax values of each successor state directly implementing some defined function
- The recursion proceeds from the initial node to all the leaf nodes
- Then the minimax values are backed up through the tree as the recursion unwinds
- It performs the depth first exploration of a game tree in a complete way

Game Playing

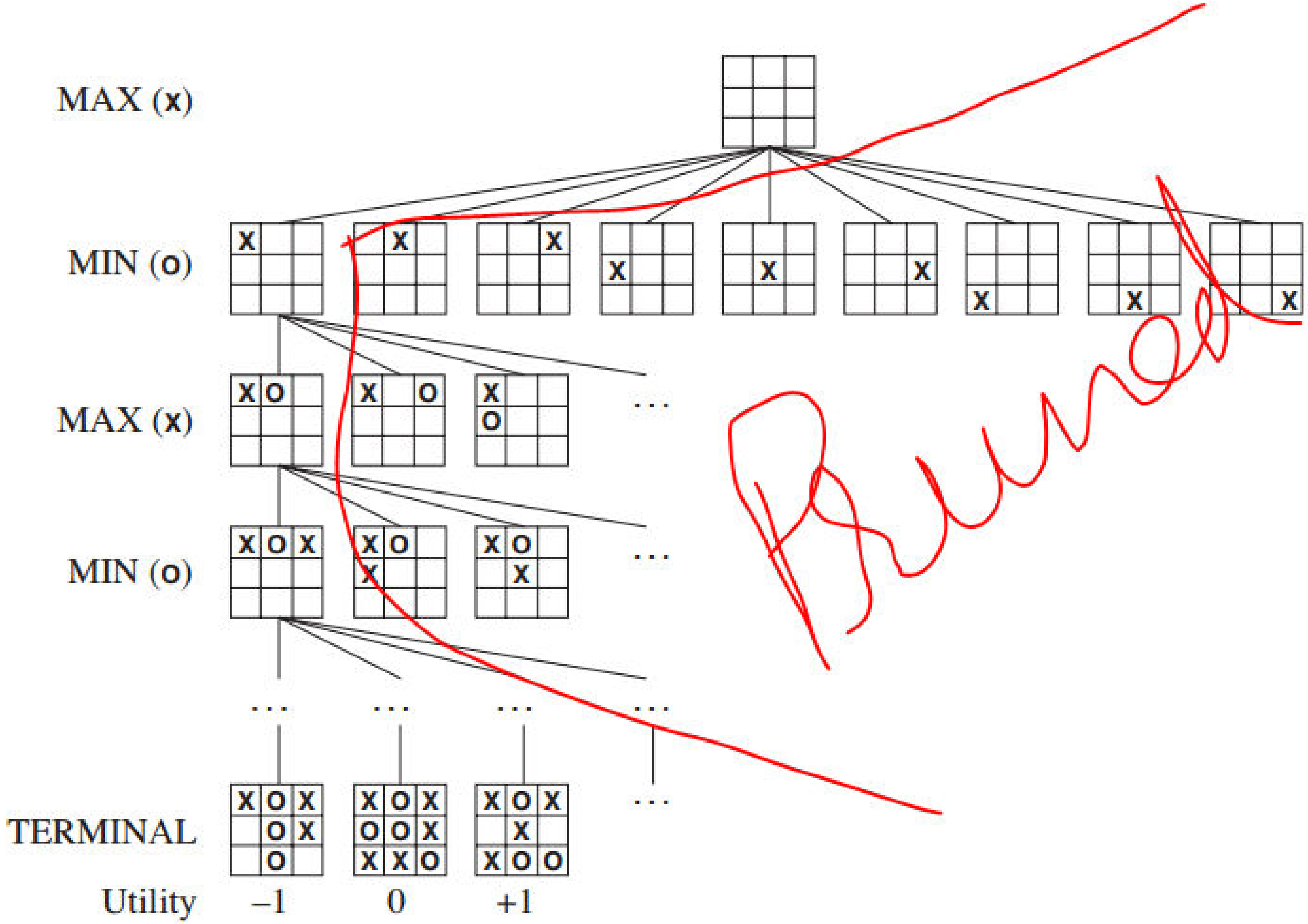


Game Playing

Alpha Beta Pruning

- Minimax algorithm has to examine exponentially increasing number of moves
- As the exponential rise can't be avoided Pruning cut it into halves
- By not considering a large part of the tree number of states to be calculated is cut down
- When applied to a standard minimax tree, alpha beta pruning returns the same move as minimax would, but prunes away the branches which couldn't possibly influence the final decision
- Alpha beta pruning could be applied to the trees of any depth

Game Playing



Reasoning about Uncertainty Probability

- One of the most common characteristics of the human information available is its imperfection due to partial observability, non deterministic or combination of both
- An agent may not know what state it is in or will be after certain sequence of actions
- Agent can cope with these defects and make rational judgments and rational decisions to handle such uncertainty and draw valid conclusions

Reasoning about Uncertainty Probability

What is uncertainty?

- The lack of the exact knowledge that would enable us to reach a perfectly reliable conclusion
- Classical Logic permits only exact reasoning i.e. perfect knowledge always exists

IF A is true THEN A is not false	and	IF B is true THEN B is not false

- In Real world such clear cut knowledge could not be provided to systems

Reasoning about Uncertainty Probability

Sources of Uncertain Knowledge

- **Weak Implication:** Domain experts and knowledge engineer have rather painful or hopeless task of establishing concrete correlation between IF(Condition) and THEN(action) part of rules. Vague Data.
- **Imprecise Language :** NLP is ambiguous and imprecise. We define facts in terms of **often, sometimes, frequently, hardly ever.** Such can affect IF-THEN implication
- **Unknown Data:** incomplete and missing data should be processes to an approx. reasoning with this values
- **Combining the views of different experts:** Large system uses data from many experts

Reasoning about Uncertainty Probability

- The basic Concept of probability plays significant role in our life like we try to determine the probability of rain, prospect of promotion, likely hood of winning in Black Jack
- The probability of an event is the proportion of cases in which the event occurs (Good, 1959)
- Probability, mathematically, is indexed between 0 and 1
- Most events have probability index strictly between 0 and 1, which means that each event has at lease two possible outcomes: favorable outcome or success and unfavorable outcomes or failure

$$\mathrm{P(success)} = \frac{The\ number\ of\ successes}{The\ number\ of\ possible\ outcomes}$$

$$\mathrm{P}(failure) = \frac{The\ number\ of\ failure}{The\ number\ of\ possible\ outcomes}$$

Reasoning about Uncertainty Probability

- If s is the number of success and f is the number of failure then:

$$\mathrm{P}(success) = \frac{s}{s+f}$$

$$\mathrm{P}(failure) = \frac{f}{s+f}$$

and

$$p + q = 1$$

Reasoning about Uncertainty Probability

- Let us consider classical examples with a coin and a dice. If we throw a coin, the probability of getting a head will be equal to the probability of getting a tail. In a single throw, s = f = 1, and therefore the probability of getting a head (or a tail) is 0.5.
- Consider now a dice and determine the probability of getting a 6 from a single throw. If we assume a 6 as the only success, then s = 1 and f = 5, since there is just one way of getting a 6, and there are five ways of not getting a 6 in a single throw. Therefore, the probability of getting a 6 is

$$\mathrm{P} = \frac{1}{1+5} = 0.1666$$

Likewise, the probability of not getting 6 is

$$\mathrm{q} = \frac{5}{1+5} = 0.8333$$

Reasoning about Uncertainty Probability

- Above instances are for independent events i.e. mutually exclusive events which can not happen simultaneously
- In the dice experiment, the two events of obtaining a 6 and, for example, a 1 are mutually exclusive because we cannot obtain a 6 and a 1 simultaneously in a single throw. However, events that are not independent may affect the likelihood of one or the other occurring. Consider, for instance, the probability of getting a 6 in a single throw, knowing this time that a 1 has not come up. There are still five ways of not getting a 6, but one of them can be eliminated as we know that a 1 has not been obtained. Thus,

$$p = \frac{1}{1 + (5 - 1)}$$

Reasoning about Uncertainty Probability

- Let A and B be two not mutually exclusive events, but occur conditionally on the occurrence of other.
- The probability of event A will occur if event B occurs is called conditional Probability

$$\mathrm{p(A|B)} = \frac{the\ number\ of\ times\ A\ and\ B\ can\ occur}{the\ number\ of\ times\ B\ can\ occur}$$

The probability of both A and B will occur is called joint probability $(\boldsymbol{A} \cap \boldsymbol{B})$

$\mathrm{p(A|B)} = \frac{p(A \cap B)}{p(B)}$, the probability of A occurring given B has occurred

$\mathrm{p(B|A)} = \frac{p(B \cap A)}{p(A)}$, the probability of B occurring given A has occurred

Reasoning about Uncertainty Probability

Joint probability is commutative, thus

$$p(A \cap B) = p(B \cap A)$$

Therefore,

$$p(A \cap B) = p(B|A) * p(A)$$

Now the final equation becomes:

$$p(A|B) = \frac{p(B|A)*p(A)}{p(B)} \text{ ---(a)}$$

Where:

p(A|B) is the conditional probability that event A occurs given event B has occurred
p(B|A)is the conditional probability that event B occurs given event A has occurred

p(A)is the probability of event A occurring   p(B)is the probability of event B occurring

The above equation (a) is known as **Bayesian Rule**

Reasoning about Uncertainty Probability

- For *n* number of mutually exclusive event B we have

$$p(A \cap B_1) = p(A|B_1) \times p(B_1)$$
$$p(A \cap B_2) = p(A|B_2) \times p(B_2)$$
$$\vdots$$
$$p(A \cap B_n) = p(A|B_n) \times p(B_n)$$

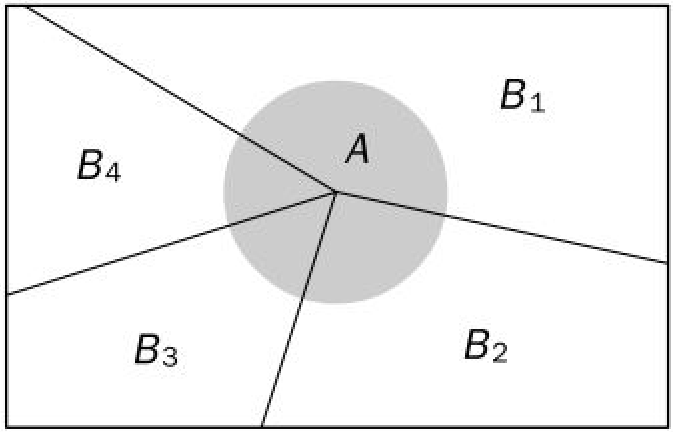
or when combined:

$$\sum_{i=1}^{n} p(A \cap B_i) = \sum_{i=1}^{n} p(A|B_i) \times p(B_i)$$

Reasoning about Uncertainty Probability

- Summed over an exhaustive list of events for Bi, we get :

$$\sum_{i=1}^{n} p(A \cap B_i) = p(A)$$



- $$p(A) = \sum_{i=1}^{n} p(A|B_i) \times p(B_i)$$

Reasoning about Uncertainty Probability

- If the occurrence of A depends on only two mutually exclusive events, i.e. B and NOT B. then above equation becomes

$$p(A) = p(A|B) \times p(B) + p(A|\neg B) \times p(\neg B)$$

- Similarly,

$$p(B) = p(B|A) \times p(A) + p(B|\neg A) \times p(\neg A)$$

- Substituting above equations in Bayesian Equation, We get:

$$p(A|B) = \frac{p(B|A) \times p(A)}{p(B|A) \times p(A) + p(B|\neg A) \times p(\neg A)}$$

Bayesian Networks

Why Bayesian Network???

- To represent the probabilistic relationship between two different classes
- To avoid dependencies between values of attributes by joint conditional probability distribution
- In Naïve Bayes classifier, attributes are conditionally independent

Bayesian Networks

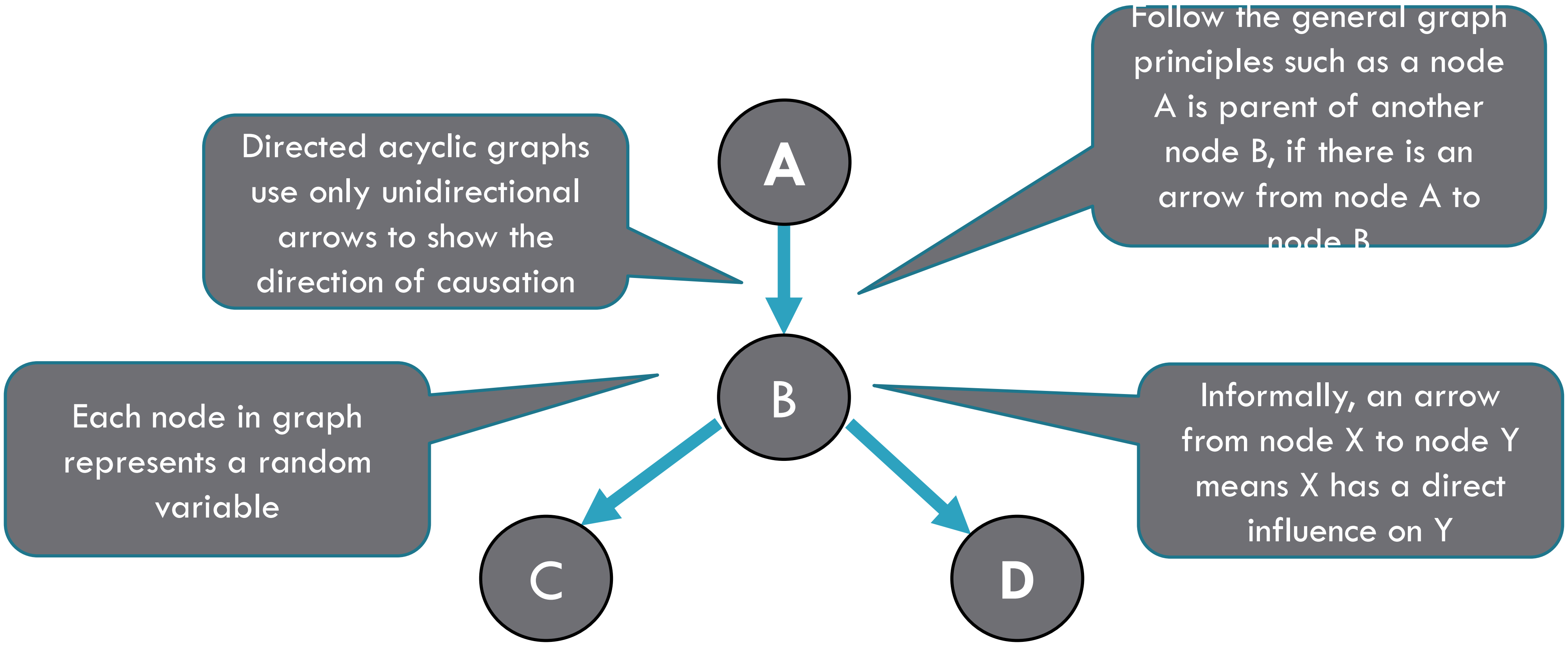
- Bayesian Network are also known as **Bayes Network, Belief Networks** and **Probabilistic Networks**

- A BN is defined by two parts, **Directed Acyclic Graph (DAG)** and **Conditional Probability Tables (CPT)**

  Nodes → Random Variables

  Arcs→ Indicates Probabilistic dependencies between nodes

Bayesian Networks



Bayesian Networks

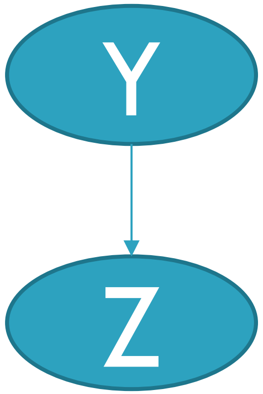
A BN is a directed graph with the following properties:

- **Nodes:** Set of Random Variables which may be discrete or continuous
- **Directed Links (Arcs)** : The real meaning od a link from node X to node Y is that X has a direct influence on Y
- Each node has a Conditional Probability Distribution $\mathbf{P}(X_i|Parents(X_i))$ that quantifies the effects that the parent have on the node
- The graph has no directed cycles

Bayesian Networks

A BN is a directed graph with the following properties (contd…)

- If an arc is drawn from Y to Z, then Y is a parent or immediate predecessor of Z, and Z is a descendant of Y



- Each variable is conditionally independent of its non-descendants in the graph, given its parents

Bayesian Networks

Incremental Network Construction:

1. ***Nodes***: First determine the set of variables that are required to model the domain. Now order them, $\{X_1, X_2, \ldots, X_n\}$. Any order will work, but the resulting network will be more compact if the variables are ordered such that causes precede effects
2. ***Links*** : for $i = 1\ to\ n \quad do$:
   1. Choose, from $X_1, \ldots, X_{n-1}$, a minimal set of parents for X_i such that equation $\mathbf{P}(X_i|X_{i-1}, \ldots, X_1) = \mathbf{P}(X_i|Parents(X_i))$ is satisfied
   2. For each parent insert a link from the parent to X_i
   3. CPTs: Write down the Conditional Probability Table, $\mathbf{P}(X_i|Parents(X_i))$

Bayesian Networks

Conditional Independence:

$$\mathbf{P}(X_1, X_2, \dots, X_n) = \mathbf{P}(X_n | X_{n-1}, \dots, X_1)\, \mathbf{P}(X_{n-1}, \dots, X_1)$$
$$= \mathbf{P}(X_n | X_{n-1}, \dots, X_1)\, \mathbf{P}(X_{n-1}, \dots, X_1) \dots \mathbf{P}(X_2 | X_1) \mathbf{P}(X_1)$$
$$= \sum_{i=1}^{n} \mathbf{P}(X_i | Parents(X_i))$$

A BN represents Conditional Independence

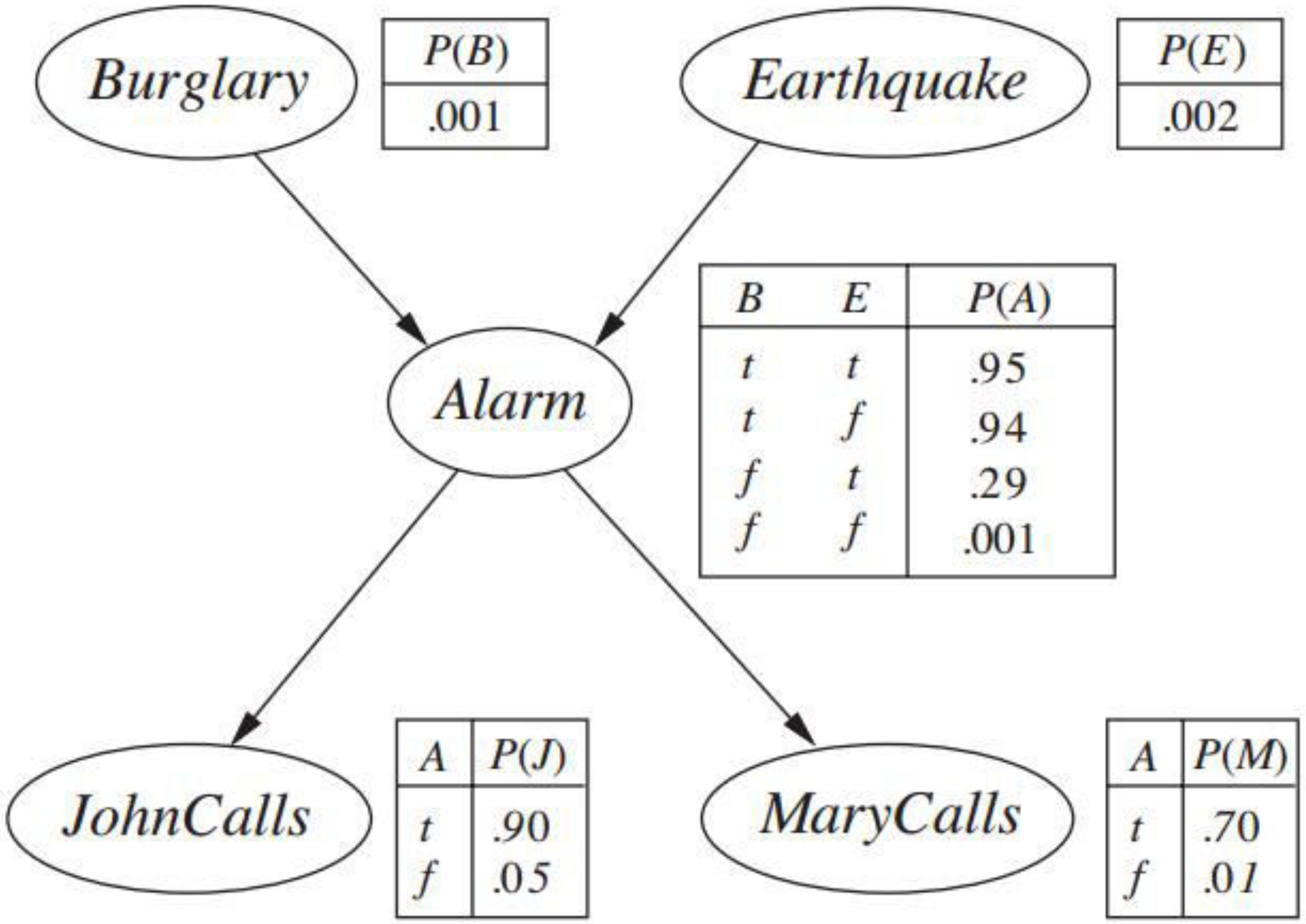
$$\mathbf{P}(X_i | X_{i-1}, \dots, X_1) = \mathbf{P}(X_i | Parents(X_i))$$

Bayesian Networks

Example

- Burglar Alarm at Home
  - Fairly reliable at detecting a Burglary
  - Also Respond at times of Earthquake
- Two neighbors (John and Mary) on hearing Alarm calls you
  - John always calls when he hears the alarm, but sometimes confuses the telephone ringing with the alarm and calls then too
  - Mary likes aloud music and sometimes misses the alarm altogether

Bayesian Networks



Bayesian Networks

- Inference from Effect to cause; given Burglary, what is $P(J|B)$?

  $P(J \mid B) = ?$

  first calculate probability of Alarm ringing on burglary:

  $P(A \mid B) = P(B)P(\neg E)P(B\cap \neg E) + P(B)P(E)P(B\cap E)$

  $P(A \mid B) = 1*(0.998)*(0.94) + 1*(0.002)*(0.95)$

  $P(A \mid B) = 0.94$

  Now, Let us calculate $P(J \mid B)$

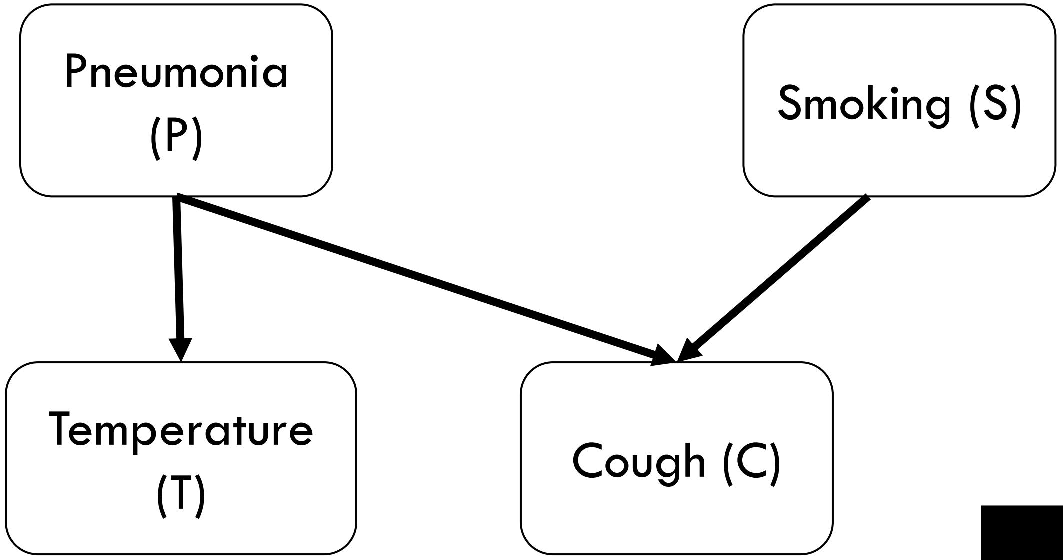
  $P(J \mid B) = P(A \mid B)*P(J) + P(\neg(A \mid B))*P(\neg J)$

  $P(J \mid B) = (0.94) * (0.9) + (0.06) * (0.05) = 0.85$

- Also calculate $P(M|B) = ?$

Bayesian Network

Pneumonia	
True	0.1
False	0.9



smoking	
True	0.2
False	0.8

	Temperature	
Pneumonia	Yes	No
Yes	0.9	0.1
No	0.2	0.8

		Cough	
Pneumonia	**Smoking**	True	False
True	Yes	0.95	0.05
True	No	0.8	0.2
False	Yes	0.6	0.4
False	No	0.05	0.95

Find:

a. P(C | S ^ P)

b. P(S | C)

Bayesian Networks

Benefits of BN:

- It can readily handle incomplete data sets
- It allows one to learn about causal relationships
- It readily facilitate use of prior knowledge
- It Provide a natural representation for conditional independence
- It is more complex to construct the graph

Case Base Reasoning

- Reasoning that adapts previous solutions for similar problem in solving new problem in hand
  - Many problem decision makers encountered are similar to old cases
  - Often more efficient to start with the previous solution to a similar problem than to generate the entire solution again from scratch
  - Experts solve problem based on previous cases
    - Ex. Court Legal Cases, etc.

Case Base Reasoning

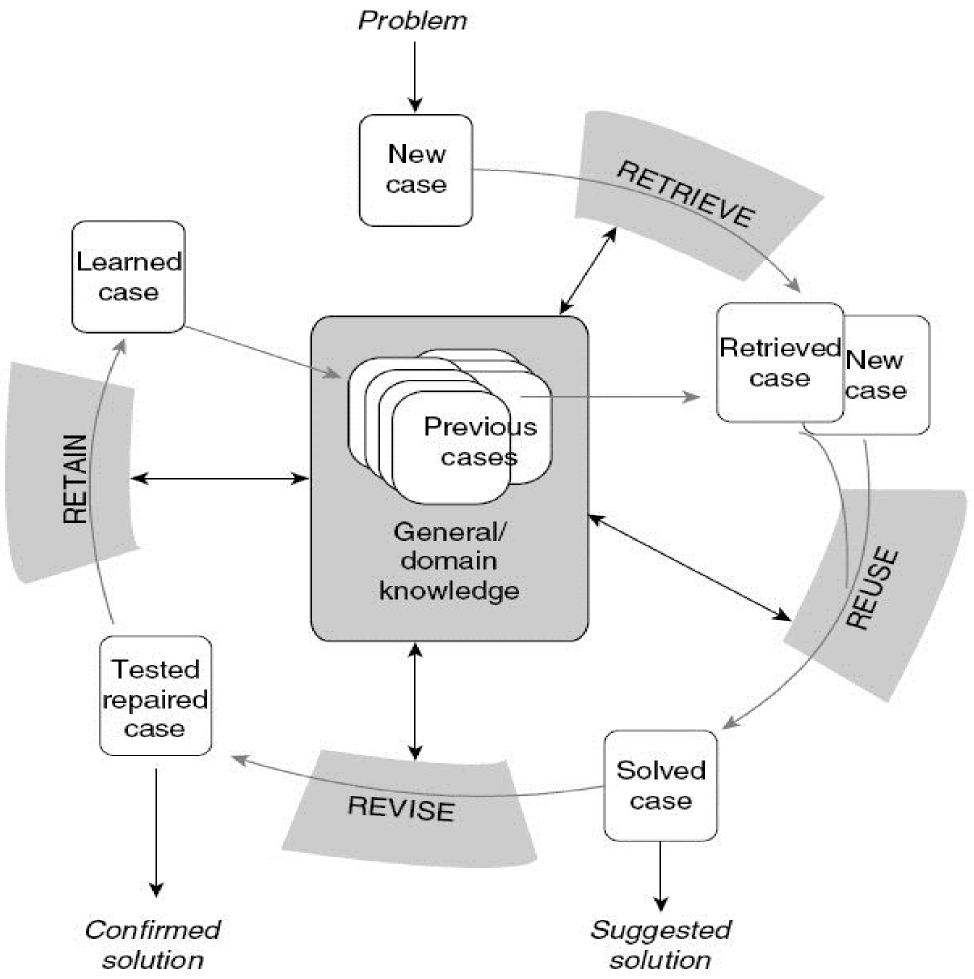
4 Re's

Retrieve

Reuse

Revise

Retain



Case Base Reasoning

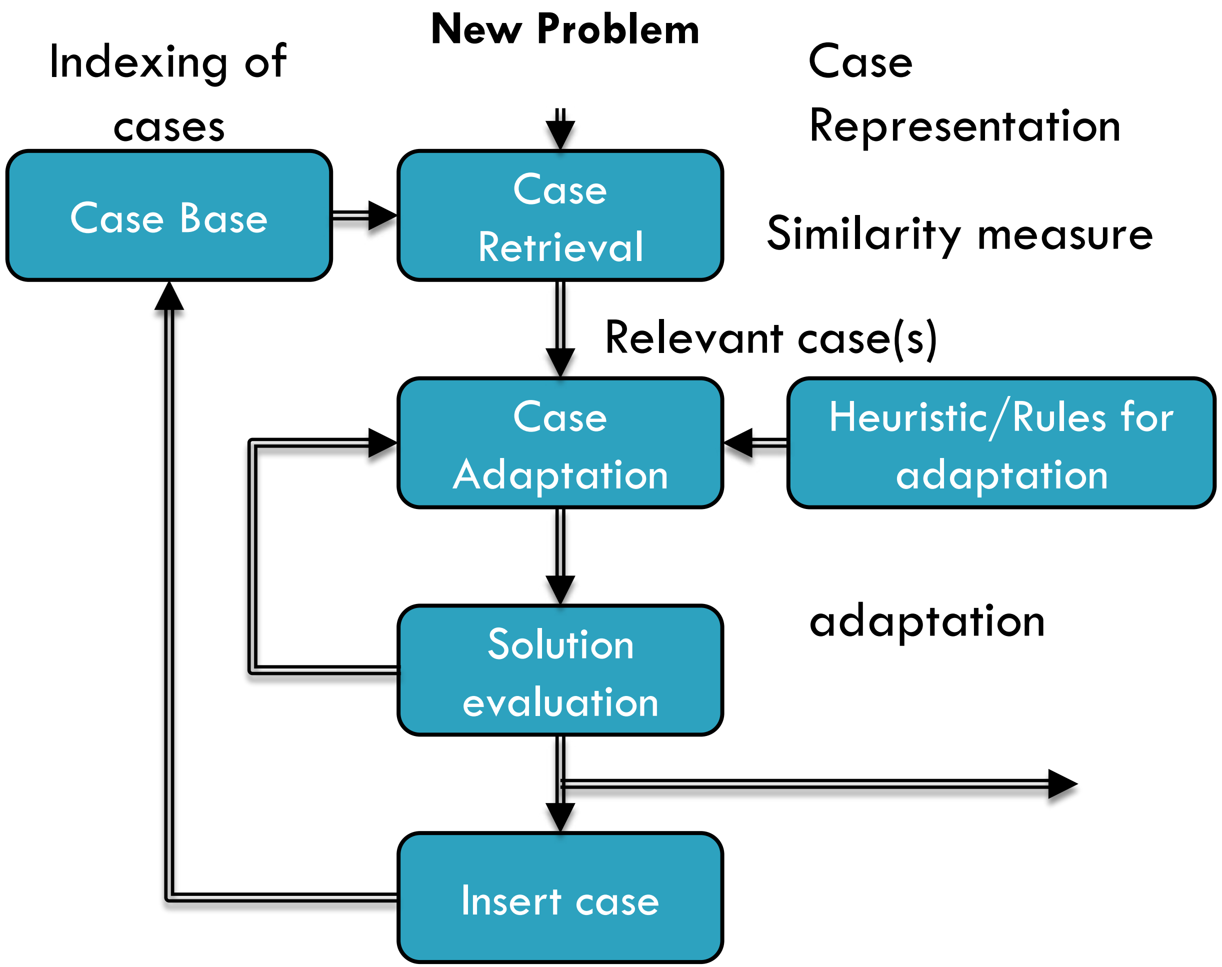


Fig: Case Base Organization

Case Base Reasoning

Components of CBR

- Case Representation
  - The Problem: describes the status of the world when the case occurs
  - The solution: states the derived solution to that problem, and/or
  - The outcome: the state of the world after the case occurred
  - Text, numbers, symbols, plans, multimedia, etc

Case Base Reasoning

Components of CBR

- Case Representation
  - What to store in a case
    - Appropriate ***structure*** to describe case contents
  - How to organize and index for effective retrieval and reuse
    - Functionality and ease of acquisition

Case Base Reasoning

Components of CBR

- Case Indexing
  - Assign indices to cases to facilitate their retrieval
  - Features and dimensions tend to be predictive
  - The system has to retrieve the right case at the right time
  - Predictive, useful, abstract and concrete

[Note: Abstract enough to allow widening the future use of the case-base; Not too abstract to avoid retrieving too many cases]

Case Base Reasoning

Components of CBR

- Case base organization
  - Flat Memory
    - Sequentially in a simple list, array or file
  - Hierarchical organization
    - Large case base
    - Only small subset needs to be considered during the retrieval
    - Organize specific cases which share similar attributes under a more general structure

Case Base Reasoning

Components of CBR

- Case base organization
  - Flat Memory
    - Nearest Neighbor
    - Weighting: by experts
  - Hierarchical organization
    - Tree search
    - Find the node that best matches the input

Case Base Reasoning

Components of CBR

- Case Adaptation
  - Structural adaptation
    - Adaptation rules are applied directly to the solution stored in cases
  - Derivational adaptation
    - Reuses the algorithms, methods or rules that generated the original solution to produce a new solution to the current problem
  - Simple or complex techniques, depend on the problem domain

Case Base Reasoning

Development of CBR

- Case Representation
  - Attributes that identify problems
  - Indices for storage and retrieval
- Similarity measure
  - Features that explain solutions
- Adaptation
  - Domain theory of impact of attributes on solutions
- Case base organization
  - A CBR system is heavily dependent on structure and content of case base

Case Base Reasoning

CBR Applications

- Legal reasoning [Hypo, JUDGE]
- Diagnosis[CASEY, Protos]
- Design[Caliver]
- Schedule[CABINS]
- Help desk support[CASCADE, ReMind]
- Planning[Chef]

References

- Rich and Knight
- Russel and Norvig