

# Artificial Intelligence

Chapter I: Introduction to AI

# Outline

- **Overview**
- **Definition of AI**
- AI and Related Fields
- Brief History of AI
- Applications of AI
- Importance of AI
- Definition of Knowledge and Learning
- Importance of Knowledge and Learning

# Overview- Background

- Humans and Mental Capacities
- How we Think?
  - How we Perceive, Understand, Predict and Manipulate a large and complicated world?
- Understand Intelligent Entities
- Build Intelligent Entities

# Overview- AI and Its Subfields

- General Purpose Areas
  - Learning
  - Perception
  - Natural Language Processing
  - Common-sense Reasoning
  - Robot Control
- Specific Tasks
  - Games (Chess, Backgammon, Cards, Checkers, Tic-Tac-Toe)
  - Mathematical Theorems (Geometry, Calculus, Logic, Proving properties)
  - Scientific Analysis
  - Medical Analysis
  - Financial Analysis
  - Writing Literatures (Poems)

AI systemizes and automates intellectual tasks.

# Definition of AI- What is AI?

- A thought process
- Reasoning
- Fidelity to Human Performance
- Rationality (doing right thing)

# Definition of AI- Approaches to AI

- Act Humanly: Turing Test Approach
- Think Humanly: Cognitive Modelling Approach
- Think Rationally: The “Laws of Thought” Approach
- Act Rationally: The Rational Agent Approach

# Act Humanly: Turing Test Approach

- The art of creating machines that perform functions that require intelligence when performed by people.(Kurzwail, 1990)
- The study of how to make computers do things at which, at the moment, people are better.(Rich and Knight, 1991)
- Based on Turing Test (Alan Turing, 1950)
  - Test based on indistinguishability from undeniably intelligent entities (human).
  - The computer passes a test if a human interrogator, after posing some written questions, can't tell whether the responses were made by a human or not.

# Act Humanly: Turing Test Approach

- Capabilities need of the computer for the tests:
  - Natural Language Processing (ability to communicate successfully)
  - Knowledge Representation (store what it knows or hears)
  - Automated Reasoning (use the stored information to answer questions and draw new conclusions)
  - Machine Learning (adapt to new circumstances and to detect and extrapolate patterns)
- And
  - Computer Vision (to perceive objects)
  - Robotics (to manipulate objects and move about)

# Think Humanly: Cognitive Modelling Approach

- The exciting new effort to make computers think... machines with minds, in the full and literal sense. (Haugeland, 1985)
- “The automation of activities that we associate with human thinking, activities such as decision-making, problem solving, learning...”(Bellman, 1978)
- Based on Cognitive Science
  - Cognitive science brings together compute models from AI and experimental techniques from psychology to try to construct precise and testable theories of the workings of the human mind.
  - Needs understanding of how human thinks?
  - Example: General Problem Solver-GPS

# Think Rationally: The Laws of Thought Approach

- The study of mental faculties through the use of computational models. (Charniak and McDermott, 1985)
- The study of the computations that make it possible to perceive, reason and act. (Winston, 1992)
- Based on Rational Thinking (Right Thinking)
  - Irrefutable Reasoning Process
  - Syllogisms providing patterns for argument structures that always yielded correct conclusions
  - Logic: the Laws of thought
  - Logician tradition within AI hopes to build on such programs to create intelligent systems

# Act Rationally: The Rational Agent Approach

- Computational Intelligence is the study of the design of intelligent agents. (Poole et al., 1998)
- AI... is concerned with intelligent behaviour in artifacts. (Nilsson, 1998)
- Based on Intelligent Agents
  - Agents are the things that act. Computer agents are expected to have other attributes that distinguish them from mere “programs”, such as operating under autonomous control, perceiving their environment, persisting over a prolonged time period, adapting to change, and being capable of taking on another’s goals.
  - Rational agents are those who act so as to achieve the best outcome or, the best expected outcome when there is uncertainty.

# AI and Related Fields- The Foundations of AI

- Philosophy (428 B.C. – Present)
  - Drawing conclusions from the rules
  - Mental mind and physical brain
  - Where does knowledge come from?
  - How does knowledge lead to action?
    - Dualism, Materialism, Empiricism, Induction, Logical Positivism, Confirmation Theory
- Mathematics (800 A.D. – Present)
  - What are the formal rules to draw conclusions?
  - What can be computed or manipulated?
  - How do we reason with uncertain information?
    - Algorithms, NP Completeness, Probability, Incompleteness Theorem, Intractability

# AI and Related Fields- The Foundations of AI

- Economics (1776 A.D. – Present)
  - How decisions can be made to maximize payoff?
  - How can something be done when others may not go along?
  - How can this be done when payoff may be far in future?
    - Decision Theory, Game Theory, Operations Research, Satisficing
- Neuroscience (1861 A.D. – Present)
  - How are information processed by the human brain?
    - Neurons
- Psychology (1879 A.D. – Present)
  - How do humans and animals think and act?
    - Behaviourism, Cognitive Psychology, Cognitive Science

# AI and Related Fields- The Foundations of AI

- Computer Engineering (1940 A.D. – Present)
  - How can we build an efficient computer?
- Control theory and Cybernetics (1948 A. D. – Present)
  - How can artifacts operate under their own control?
    - Control Theory, Cybernetics, Objective Function
- Linguistics (1957 A.D. – Present)
  - How does language relate to thought?
    - Computational Linguistics, NLP, Knowledge Representation

# Brief History of AI- The Gestation Period (1943-1955)

- Warren McCulloch and Walter Pitts (1943): 3 Sources (Knowledge of basic physiology and functions of neurons in brain; a formal analysis of propositional logic due to Russell and Whitehead; Turing's theory of computation → proposed model of artificial neurons characterized by on/off logic that could even learn)
- Hebbian Learning (1949) by Donald Hebb: demonstration of simple updating rule for modification of the connections strengths between the neurons
- Marvin Minsky and Dean Edmonds (1951): first neural network computer → SNARC (3000 Vacuum Tubes and a pilot mechanism from B-24 bomber to simulate a network of 40 neurons <Von Neumann>)
- Alan Turing (1950): “Computing Machinery and Intelligence” (Articulated a complete vision of AI, introducing Turing Test, Machine Learning, Genetic Algorithms, Reinforcement Learning)

# Brief History of AI- The Birth (1956)

- John McCarthy, Marvin Minsky, Claude Shannon and Nathaniel Rochester focused researches on automata theory, neural nets, and intelligence organizing a 2-month workshop (1956)
- Two participants Allen Newell and Herbert Simon presented works on reasoning program named the Logic Theorist that was claimed to think non numerically and prove many theorems but the paper was not recognized by the *Journal of Symbolic Logic*
- But the workshop laid the foundation for AI and the participants of the workshop became the leaders in the field of Artificial Intelligence

# Brief History of AI: The Early Period (1952-1969)

- General Problem Solver – Thinking Humanly Purpose
- Nathaniel Rochester in IBM came with some of the first AI Programs
- Herbert Gelernter (1959) – Geometry Theorem Prover
- Arthur Samuel (1952) – Series of Programs for checkers leading to skilled checker program that could play better than its creator
- John McCarthy (1958) – Contributions
  - Lisp- a high level dominant AI programming language
  - Paper entitled *Programs and Common Sense* described the Advice Taker as a complete AI System- use knowledge to search for solutions to problems
  - AI Lab at Stanford

# Brief History of AI: The Early Period (1952-1969)

- Marvin Minsky (1958) – anti logical outlook
- J. A. Robinson – discovery of Resolution Method
- Cordell Green (1969) – Question answering and planning system
- Minsky's Students focused on study to solve limited problems that seems to require AI and this domain is called microworlds.
- James Slagle (1963) – SAINT program solved closed form calculus integration problems
- Tom Evan (1968) – ANALOGY program solved geometric analogy problems

# Brief History of AI: The Early Period (1952-1969)

- Daniel Bobrow (1967) – STUDENT program solved algebra problems
  - David Huffman (1971) – The vision project
- David Waltz (1975) – The vision and constraint propagation
- Patrik Winston (1970) – The learning theory
- Terry Winoguard (1972) – The natural language understanding program
- Scott Fahlman (1974) – The planner
- Block World – Rearrange the blocks using robot hand
- McCulloch and Pitts – Neural Network
  - Bernie Widrow (1962) – Adalines
  - Frank Rosenblatt (1962) – Perceptron and Perceptron Convergence theorem

# Brief History of AI: Reality Dawns (1966-1973)

- Problems were faced while realization of AI Projects:
  - The most early programs contained little or no knowledge in their subject matter; success was merely based on simple syntactic manipulation
  - Intractability of many of the problems; microworlds were comparatively less complicated than real world problems
  - Fundamental limitations on the basic structures being used to generate intelligent behaviour → Limitations of existing neural network methods identified
- AI failed to convince the funding agencies as the expectations were not matched

# Brief History of AI: Knowledge Based Systems (1969-1979)

- Problem Solving in prior period was based on weak methods → those try to string together the elementary reasoning steps to find complete solutions from a general purpose context
- Alternative was suggested → domain specific knowledge that allows larger reasoning steps and can be easily used to handle typically occurring cases of narrow area of expertise
- Development of knowledge based Systems
- Buchanan et al. (1969) – The DENDRAL Program that solve the problem of inferring molecular structure from the information provided by mass spectrometer

# Brief History of AI: Knowledge Based Systems (1969-1979)

- Heuristic Programming Project to identify where could Expert Systems be used
- MYCIN Program → used 450 rules to diagnose blood infections
  - Performed better than junior doctors
- Roger Schank and his students developed a series of programs related to AI and Linguistics
- Development of Successful Rule based Expert Systems
- Minsky (1975) developed idea of frames → that adopted structured approach to assemble facts about particular object and event types and arrange them into a large taxonomy hierarchy analogous to a biological taxonomy

# Brief History of AI: AI as an Industry (1980-Present)

- R1 (1986) → first successful commercial expert system by DEC
  - Helps to configure orders for new computers
  - Saved \$40 million for DEC
- DEC (1988), developed 40 Expert Systems
- Du Pont, 100 in use and 500 in pipeline
- 1981, Japan announced “Fifth Generation” Computers which were intelligent and US based company MCC also announced similar computer → Could not come to reality
- AI Winter in the future due to unrealistic promises that were not delivered

# Brief History of AI: Return of Neural Networks (1986-Present)

- Neural networks return to popularity
- Major advances in machine learning algorithms and applications
- Reinvention of back-propagation learning algorithm in mid 1980s
  - Concept of Parallel Distributed Processing
- Connectionist models of intelligent systems were seen which focused on unjustifiability of symbolic manipulation in decision making

# Brief History of AI: AI as a Science (1987-Present)

- AI focuses on scientific study
- Integration of learning, reasoning, knowledge representation in AI
- AI methods used in vision, language, data mining, etc.
- Bayesian networks as a knowledge representation framework
- Hidden Markov Models based on mathematical theory and training theories
- Emergence of Intelligent Agents

# Brief History of AI: Success Stories

- Deep Blue defeated the reigning world chess champion Garry Kasparov in 1997
- AI program proved a mathematical conjecture (Robbins conjecture) unsolved for decades
- During the 1991 Gulf War, US forces deployed an AI logistics planning and scheduling program that involved up to 50,000 vehicles, cargo, and people
- NASA's on-board autonomous planning program controlled the scheduling of operations for a spacecraft
- Proverb solves crossword puzzles better than most humans
- Robot driving: DARPA grand challenge 2003-2007
- 2006: face recognition software available in consumer cameras

# Applications of AI

- Autonomous Planning and Scheduling
- Game Playing
- Autonomous Control
- Diagnosis
- Logistics Planning
- Robotics
- Language understanding and Problem Solving

# Importance of AI

- Create a never-ending thought process and collective that could solve our problems
- Thinking of every possible solution
- With artificial intelligence, we could build computers, upon thousands of computers, that could all work in unison to solve our great and most dire problems

# Definition of Knowledge and Learning

- Knowledge is the justified true belief
  - Data → Information → Knowledge
- Learning is the process of acquiring new or modifying and reinforcing the existing knowledge, behaviours, skills, or values through the synthesis and manipulation of information
- Machine Learning → embedding the learning ability into the machine or computers

# Importance of Knowledge and Learning

- For Understanding the Environment
- For Updating the Knowledge base
- For Problem Solving
- For Decision Making
- For Building Intelligent Systems

# References

- Russell, S. and Norvig, P., 2011, Artificial Intelligence: A Modern Approach, Pearson, India.
- Rich, E. and Knight, K., 2004, Artificial Intelligence, Tata McGraw hill, India.

# Thank You

Any Queries?

One Day Machine will be Intelligent. What about Man?

# UNIT 1

Goals in Problem Solving

# Contents

2

- Goal Schemas Use in Planning
- Concept of Non-Linear Planning
- Means-End Analysis
- Production Rule System
- Forward and Backward Chaining
- Mycin-Style Probabilities and Its Application

# Goal Schemas Use in Planning

3

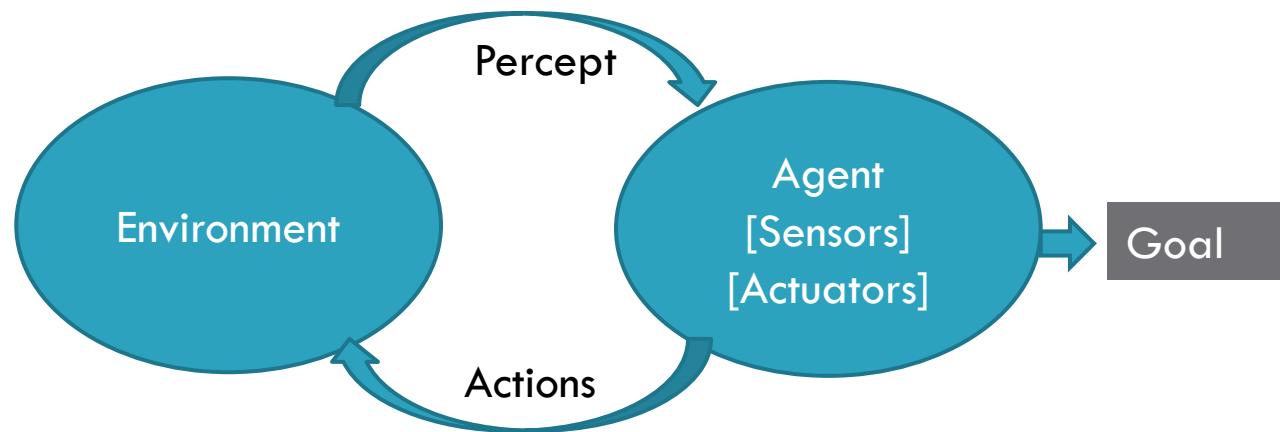
**Goal:** The state of affairs that a plan is intended to achieve and that (when achieved) terminates behavior intended to achieve it.

OR

The place designated at the end.

# Goal Schemas Use in Planning

4



Given the goal, the agent wants to find the sequence of actions that will let him/her achieve the goal.  
→ Given current situation what will be the next step?

# Goal Schemas Use in Planning

5

To build a system to solve a particular problem, we need to do four things:

- **Define the problem:** It includes precise specifications of what the initial situation(s) will be as well as what final situations constitute acceptable solutions to the problem
- **Analyse the problem:** Few very important aspects those having immense impact on the appropriateness of various possible techniques for problem solving is to be critically examined
- **Isolate and Represent the task knowledge:** Knowledge necessary to solve the problem must be identified, isolated and represented
- **Choose and apply the best technique:** Among the alternatives identify the best technique to solve the problem and apply it

# Planning

6

- To find a sequence of actions that achieves a given goal
- OR
- Given a set of operator instances ( defining possible primitive actions by an agent), an initial state description, and a goal state description or predicate, the planning agent computes a plan.
- **Problem Solving Agents + Knowledge-based Agents = Planning Agents**
- **Plan:** A sequence of operator instances, such that "executing" them in the initial state will change the world to a state satisfying the goal state description. Goals are usually specified as a conjunction of goals to be achieved

# Planning

7

**Linear Planning:** Works on one goal until completely solved before moving on to the next goal.

- Planning algorithm maintains goal stack.
- Implications:
  - No interleaving of goal achievement
  - Efficient search if goals do not interact (much)
- Advantages:
  - Reduced search space, since goals are solved one at a time
  - Advantageous if goals are (mainly) independent
  - Linear planning is sound
- Disadvantages:
  - Linear planning may produce suboptimal solutions (based on the number of operators in the plan)
  - Linear planning is incomplete.

# Planning

8

## Non Linear Planning :

- Use goal set instead of goal stack. Include in the search space all possible sub goal orderings.
- Handles goal interactions by interleaving.
- Advantages:
  - Non-linear planning is sound.
  - Non-linear planning is complete.
  - Non-linear planning may be optimal with respect to plan length (depending on search strategy employed)
- Disadvantages:
  - Larger search space, since all possible goal orderings may have to be considered.
- Somewhat more complex algorithm; more bookkeeping.

# Means-End Analysis

9

- Reducing differences between current state and goal state
  - Stop when difference is 0 (no difference)
- Subgoals
  - Intermediate goals – not your final goal-state
  - Means-end analysis is also considered a way to break up a problem into pieces (subgoals)

# Means-End Analysis

10

- Allows both backward and forward searching.
- This means we could solve major parts of a problem first and then return to smaller problems when assembling the final solution.
- GPS was the first AI program to exploit means-ends analysis.
- STRIPS (A robot Planner) is an advanced problem solver that incorporates means-ends analysis and other techniques.
- Very loosely the means-ends analysis algorithm is:
  - ▣ Until the goal is reached or no more procedures are available:
    - Describe the current state, the goal state and the differences between the two.
    - Use the difference to describe a procedure that will hopefully get nearer to goal.
    - Use the procedure and update current state.
  - ▣ If goal is reached then **success** otherwise **fail**.

# Analogy

11

- Borrowing a solution already used to solve a similar problem
- Example problem
  - Patient has a tumor in location that makes it inoperable
  - One possibility is to use a high-powered beam to destroy the tumor from the outside
  - Problem: beam will also damage surrounding healthy tissue

# Similar problem

12

- Evil king lives in a castle with his army
- Good king wants to destroy the evil king
- Good king amasses a huge army to defeat the evil king
- Problem: only narrow roads bordered by natural (immovable) obstacles lead to evil king's castle; no single road can hold the entire good king's army

# Solution to castle problem

13

- Good king divides the army into smaller divisions
  - Each division goes down a separate road at the same time
  - All divisions meet at the castle simultaneously to overtake the evil king and his army

# Solution to tumor problem

14

- Divide beam into weaker beams
- Send all weak beams into body simultaneously but from different angles
- Combined strength of beams at tumor site will destroy tumor

# Production Rule System

15

- A production system consists of a set of rules, each consisting of a left side (a pattern), that determines the applicability of the rule and a right side that describes the operation to be performed if the rule is applied.
- Example: [A, clean] → move right

Pattern	Action
---------	--------

- Have one or more knowledge base (database) that contain whatever information is appropriate for the particular task

# Production Rule System

16

- A **Control Strategy** that specifies the order in which the rules will be selected and a way of resolving the conflicts that arise when several rules matched at once
  - ▣ The first requirement of a good control strategy is that it cause motion  
    No motion action: filling the jug each time
  - ▣ The second requirement of a good control strategy is that it should be systematic  
    No systematic action: random selection action
  - ▣ The systematic control strategy is the good search technique
- Have a rule applier

# Defining Problems as a State Space Search: An Example – A Water Jug Problem

17

## Problem

- You are given two jugs, a 4 litre one and a 3 litre one. None of them have any measuring markers on it. There is a pump that can be used to fill the jugs with water. How can you get exactly a 2 litres of water in to the 4 litre jug?

# Defining Problems as a State Space Search: An Example – A Water Jug Problem

18

## State Space

- Set of ordered pair of integers  $(x, y)$ , such that  $x = 0, 1, 2, 3$ , or  $4$  and  $y = 0, 1, 2$ , or  $3$ , where  $x$  represents the quantity of water in 4 litre jug and  $y$  represents the quantity of water in 3 litre jug
- The start state is  $(0, 0)$
- The final state is  $(2, n)$

# Defining Problems as a State Space Search: An Example – A Water Jug Problem → Rules

19

1.  $(x, y)$  if  $x < 4 \rightarrow (4, y)$   
Fill the 4 litre jug
2.  $(x, y)$  if  $y < 3 \rightarrow (x, 3)$   
Fill the 3 litre jug
3.  $(x, y)$  if  $y > 0 \rightarrow (x, 0)$   
Empty the 3 litre jug
4.  $(x, y)$  if  $x > 0 \rightarrow (0, y)$   
Empty the 4 litre jug
5.  $(x, y)$  if  $x > 0 \rightarrow (x-d, y)$   
Put some water out of 4 l jug
6.  $(x, y)$  if  $y > 0 \rightarrow (x, y-d)$   
Put some water out of 3 l jug
7.  $(x, y)$  if  $x+y \geq 4 \text{ & } y > 0 \rightarrow (4, y-(4-x))$   
Put water from 3l jug into 4l jug to fill it
8.  $(x, y)$  if  $x+y \geq 3 \text{ & } x > 0 \rightarrow (x-(3-y), 3)$   
Put water from 4l jug into 3l jug to fill it
9.  $(x, y)$  if  $x+y < 3 \text{ & } x > 0 \rightarrow (0, x+y)$   
Put all the water from 4l jug in to 3l jug
10.  $(x, y)$  if  $x+y < 4 \text{ & } y > 0 \rightarrow (x+y, 0)$   
Put all the water from 3l jug in to 4l jug

# Defining Problems as a State Space Search: An Example – A Water Jug Problem → Solution

20

Water Quantity in 4 l Jug	Water Quantity in 3 l Jug	Rule Applied
0	0	2
0	3	10
3	0	2
3	3	7
4	2	4
0	2	10
2	0	

- A farmer has to cross a river with his Fox, Goose, and Grain. Each trip, his boat can only carry himself and one of his possessions. How can he cross the rivers if an unguarded fox eats the goose and an unguarded goose the grain.
- Start state [Left(Fa, Go, Fr, Fo) | Right( )]
- Goal state [Left( ) | Right(Fa, Go, Fr, Fo)]

# Forward and Backward Chaining

22

## Inference Scheme:

Inference engines in ES are responsible for deciding how the knowledge data in the KB should be used. They are responsible for the control and execution of the reasoning strategies used by ES. Backward chaining and forward chaining are strategies used to specify how rules contained in a knowledge base rule system are to be executed.

# Forward and Backward Chaining

23

**IF** the weather is rainy

**AND** the distance is  $\geq$  100 kilometers

**THEN** transportation is by car (1)

**IF** transportation is by car

**THEN** passenger insurance will be considered (2)

**IF** passenger insurance is considered

**THEN** transportation insurance cost = Rs 10,000 (3)

# Forward and Backward chaining

**Backward Chaining:** suppose we want to establish the fact that “transportation insurance cost = 10000” assuming we know only that “the weather is rainy” and “the distance is 150 km”. Backward chaining works backward from the conclusion:

Is this fact known? → No  
Can it be obtained from the rule? → Yes, from rule 3  
Which fact needs to be known? → “Passenger insurance is considered”  
Is this fact known? → No  
Can it be obtained from the rule? → yes, from rule 2  
Which fact need to know? → “Transportation by car”  
Is this fact known? → No  
Can it be obtained from the rule? → yes, from rule 1  
Which fact need to be known? → “The weather is rainy” and “Distance  $\geq 100$  km”  
Are these facts known? → Yes, “The weather is rainy” and “distance is 150 km”  
Therefore, it is true that “transportation is by car”.  
Therefore, it is true that “passengers insurance is considered”.  
Therefore, it is true that transportation insurance cost = 10000”.  
We started with the fact we wanted to prove and tried to establish all the facts needed to reach the goal. This reasoning method is called backward chaining. Backward chaining is applied when a goal or hypothesis is chosen as the starting point for problem solving. Backward chaining is also known as goal-directed, top-down or consequence driven.

# Forward and Backward chaining

25

**Forward Chaining:** this approach goes forward from starting point, via the conclusion generated at each step.

Suppose we want to prove that “transportation insurance cost = 10000” assuming we know only that “the weather is rainy” and “the distance is 150 km”

**Forward Chaining:** this approach goes forward from starting point, via the conclusion generated at each step. Suppose we want to prove that “transportation insurance cost = 10000” assuming we know only that “the weather is rainy” and “the distance is 150 km”

Is the fact known? → No

Which fact do we know? → “The weather is rainy.” And  
“The distance  $\geq 100$  km.”

Which fact follow from it? → “Transportation is by car.” Rule (1)

Is this what we want to prove? → No

What fact follow from it? → “Passenger insurance will be considered.”  
Rule (2)

Is this what we want to prove? → No

What fact follow from it? → “Transportation insurance cost = 10000.”  
Rule (3)

Is this what we want to prove? → Yes.

# Mycin-Style Probabilities and its Applications

26

- An expert system for treating blood infections
- MYCIN would attempt to diagnose patients based on reported symptoms and medical test results
- Could ask some more information and lab test results for diagnosis
- It would recommend a course of treatment, if requested MYCIN would explain the reasoning that lead to its diagnosis and recommendations
- Uses about 500 production rules
- MYCIN operated at roughly the same level of competence as human specialists in blood infections
- Uses backward chaining for reasoning

# References

27

- Elaine Rich and Kevin Knight, Artificial Intelligence, 2e
- Russel & Norvig, 3e

# UNIT 2

Intelligence

# Intelligence

2

- The ability of a system to calculate, reason, perceive relationships and analogies, learn from experience, store and retrieve information from memory, solve problems, comprehend complex ideas, use natural language fluently, classify, generalize, and adapt new situations.

# Types of Intelligence

3

Intelligence	Description	Example
Linguistic Intelligence	The ability to Speak, recognize, and use mechanisms of phonology (Speech Sounds), Syntax (Grammar), and Semantics (Meaning)	Narrator, Orator
Musical Intelligence	The ability to create, communicate wit and understand meanings made of sound, understanding of pitch, rhythm.	Musicians, Singers, Composers
Logical-Mathematical Intelligence	The ability to use and understand meanings relationships in the absence of action or objects. Understanding complex and abstract ideas	Mathematicians, Scientists

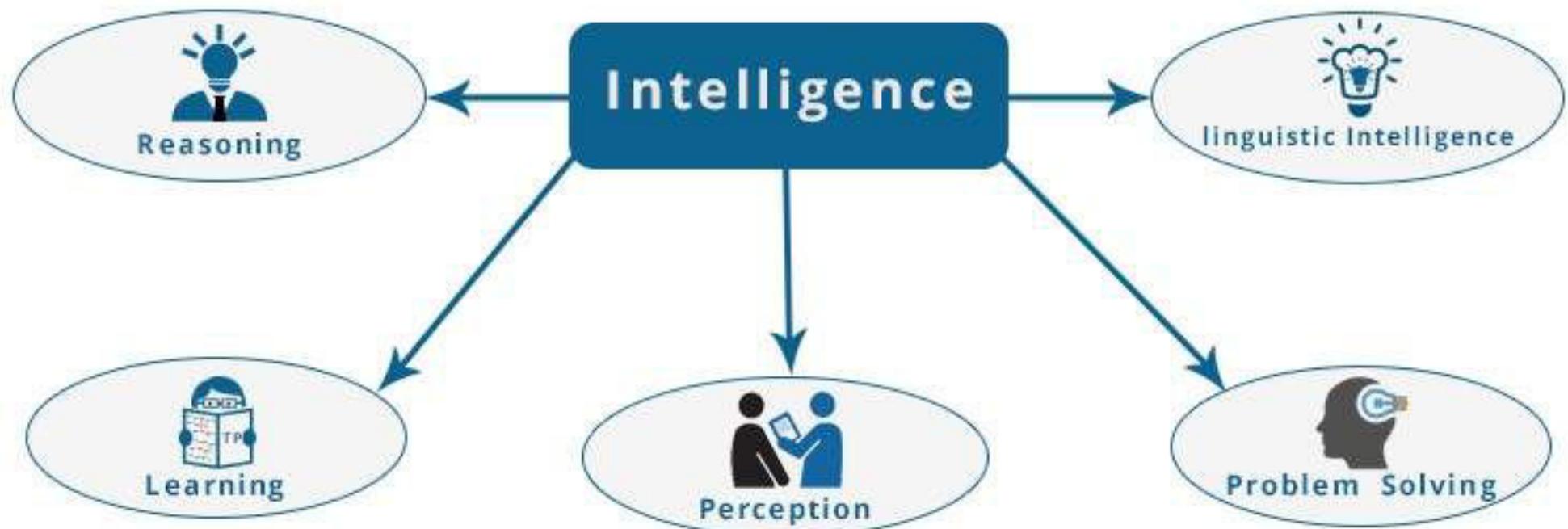
# Types of Intelligence

4

Intelligence	Description	Example
Spatial Intelligence	The ability to perceive visual or spatial information, change it, and re-create visual images without reference to the objects, construct 3D images, and to move and rotate	Map readers, Astronauts, Physicists
Bodily-Kinesthetic intelligence	The ability to use complete or part of the body to solve problems or fashion products, control over fine and coarse motor skills, and manipulate the objects.	Players, Dancers
Intra-personal intelligence	The ability to distinguish among one's own feelings, intentions, and motivations.	Gautam Buddha
Inter-personal Intelligence	The ability to recognize and make distinctions among other people's feelings, beliefs, and intentions	Mass Communicators, Interviewers

# What is Intelligence Composed Of?

5



# What are Intelligence Composed of?

6

- **Reasoning** – It is the set of processes that enables us to provide basis for judgement, making decisions, and prediction. There are broadly two types –
  - Inductive Reasoning
  - Deductive Reasoning

# What are Intelligence Composed of?

7

Inductive Reasoning	Deductive Reasoning
It conducts specific observations to makes broad general statements.	It starts with a general statement and examines the possibilities to reach a specific, logical conclusion.
Even if all of the premises are true in a statement, inductive reasoning allows for the conclusion to be false.	If something is true of a class of things in general, it is also true for all members of that class.
Example – "Nita is a teacher. All teachers are studious. Therefore, Nita is studious."	Example – "All women of age above 60 years are grandmothers. Shalini is 65 years. Therefore, Shalini is a grandmother."

# What are Intelligence Composed of?

8

- **Learning:** It is the activity of gaining knowledge or skill by studying, practising, being taught, or experiencing something. Learning enhances the awareness of the subjects of the study.  
The ability of learning is possessed by humans, some animals, and AI-enabled systems. Learning is categorized as:
  - **Auditory Learning:** It is learning by listening and hearing. For example, students listening to recorded audio lectures.
  - **Episodic Learning:** To learn by remembering sequences of events that one has witnessed or experienced. This is linear and orderly.
  - **Motor Learning:** It is learning by precise movement of muscles. For example, picking objects, Writing, etc.
  - **Observational Learning:** To learn by watching and imitating others. For example, child tries to learn by mimicking her parent

# What are Intelligence Composed of?

9

- **Problem solving:** It is the process in which one perceives and tries to arrive at a desired solution from a present situation by taking some path, which is blocked by known or unknown hurdles.  
Problem solving also includes **decision making**, which is the process of selecting the best suitable alternative out of multiple alternatives to reach the desired goal are available.
- **Perception:** It is the process of acquiring, interpreting, selecting, and organizing sensory information.  
Perception presumes **sensing**. In humans, perception is aided by sensory organs. In the domain of AI, perception mechanism puts the data acquired by the sensors together in a meaningful manner.
- **Linguistic Intelligence:** It is one's ability to use, comprehend, speak, and write the verbal and written language. It is important in interpersonal communication.

# Difference between Human and Machine Intelligence

10

- Humans perceive by patterns whereas the machines perceive by set of rules and data.
- Humans store and recall information by patterns, machines do it by searching algorithms. For example, the number 40404040 is easy to remember, store and recall as its pattern is simple.
- Humans can figure out the complete object even if some part of it is missing or distorted; whereas the machines cannot correctly.

# UNIT 3

Knowledge Representation

# Contents

2

- Knowledge Representation
  - Knowledge Based Agents
  - Formal logic
  - Connectives
  - Truth tables
  - Syntax
  - Semantics
  - Tautology
  - Knowledge Models
  - Validity
  - Well Formed Formula
- Propositional Logic
- Predicate Logic
  - FOPL
  - Interpretation
  - Quantification
  - Horn Clauses

# Outline

3

- **Inference**
  - Rules of Inference
  - Unification
  - Resolution Refutation System
  - Answer Extraction from RRS
  - Rule based Deduction System
- Statistical Reasoning
  - Probability and Bayes Theorem
  - Causal Networks
  - Reasoning in Belief Network

# Knowledge Representation

4

- An area of AI whose fundamental goal is to represent knowledge in a manner that facilitates inferring or drawing conclusion from knowledge
- Analyses how to think formally, how to use symbol to represent a domain of discourse along with the function that allow inference about the objects

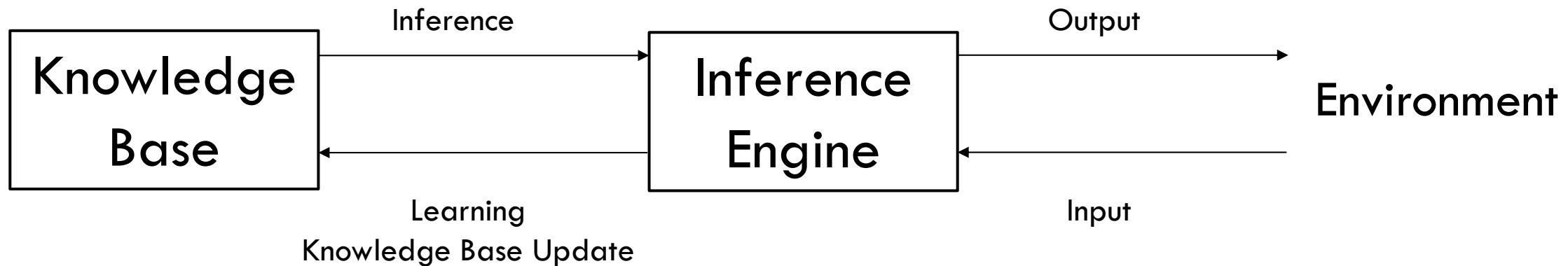
# Knowledge Representation

5

- Helps to address problems like:
  - How do we represent facts about the world?
  - How do we reason about them?
  - What representations are appropriate for dealing with the real world?
- Its objective is to express knowledge in a computer tractable form so that agent can perform well.

# Knowledge Representation

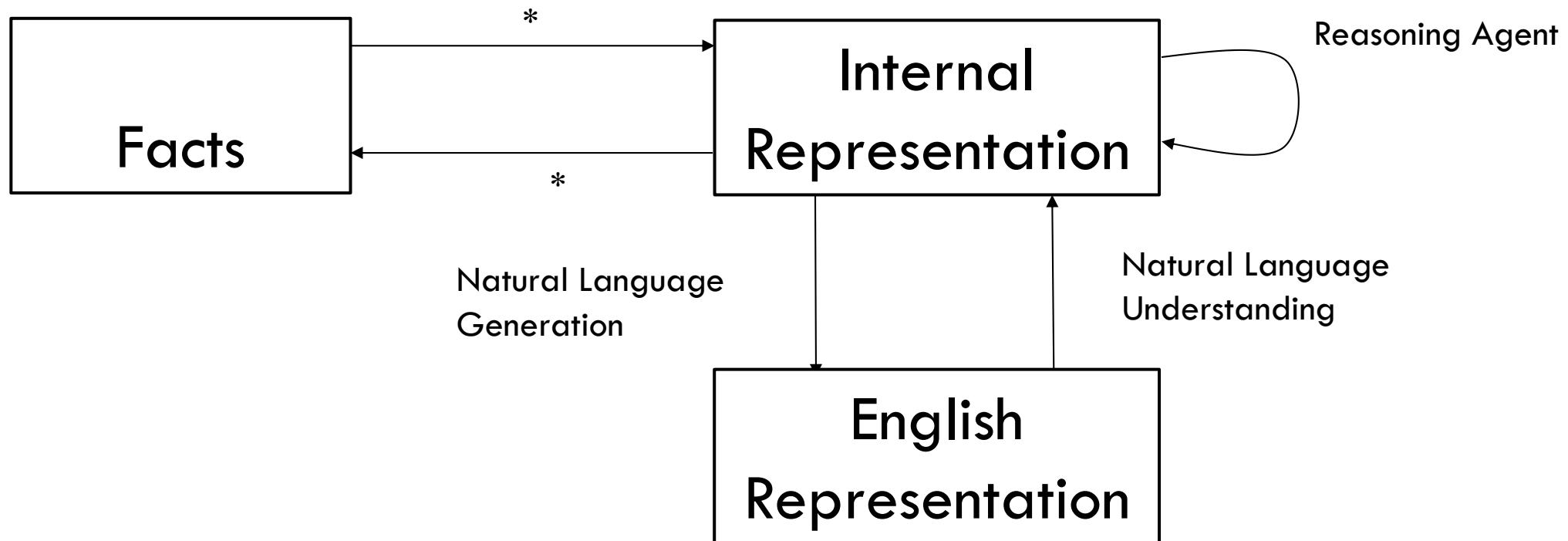
6



# Knowledge Representation

7

Figure: Mapping Facts and Representation



# Knowledge Representation: Approaches

8

- A good system for knowledge representation should have
  - Representable Adequacy: Ability to represent all kind of knowledge that are needed in the domain
  - Inferential Adequacy: Ability to manipulate the representational structure in such a way as to derive new structures corresponding to new knowledge inferred from old
  - Inferential Efficiency: Ability to incorporate into the knowledge structure additional information that can be used to focus the attention of the inference mechanism in the most promising direction
  - Acquisitional Efficiency: Ability to acquire new information easily

# Knowledge Representation: Types

9

- Simple Relational Knowledge
  - The simplest way to represent declarative facts is as a set of relations of the same sort used in database system
- Inheritable Knowledge
  - Structure must be designed to correspond to the inference mechanism that are desired
- Inferential Knowledge
  - Represents knowledge as formal logic
  - Based on reasoning from facts or from other inferential knowledge
  - Useless unless there is also an inference procedure that can exploit it
- Procedural (Imperative) Knowledge
  - Knowledge exercised in the performance of some task
  - Processed by an intelligent agent

# Knowledge Representation: Issues

10

- Are any attributes of objects so basic that they have been occurred in almost every problem domain?
- Are there any important relationships that exist among attributes of objects
- At what level should knowledge be represented?
- How should sets of objects be represented?
- How can relevant parts be accessed when they are needed?

# Knowledge Based Agent

11

- Knowledge Base: a set of sentences
- An agent having a knowledge base
- Each sentence in a knowledge base is expressed in a language called a knowledge representation language
- There must be a way to add new sentences to the knowledge base
- Logical Agents must infer from the knowledge base that has the information from the past or background knowledge

# Knowledge Based Agent: Levels of Knowledge Base

12

- **Knowledge Level**
  - The most abstract level
  - Describes agent by saying what it knows
  - Example:
    - An intelligent taxi might know that the Bagmati Bridge connects Kathmandu with Lalitpur
- **Logical Level**
  - The level at which the knowledge is encoded into formal sentences
  - Example:
    - Joins(Bagmati bridge, Kathmandu, Lalitpur)
- **Implementation Level**
  - Physical representation of the sentences in the logical level
  - Example:
    - Objects, string, dams, etc.

# Approaches of system building

13

- Declarative approach
  - Designing the representation language to make it easy to express the knowledge in the form of sentences
- Procedural approach
  - Encoded desired behaviour directly as program code

# Logic

14

- Logic
- Syntax: Formal standard to express sentences so that the sentences are well formed
- Semantics: Has to do with the meaning of sentences
  - Defines the truth of the sentences with respect to respective possible world
- Connectives: Joins the different components of the sentence
- Model and Real World
- Entailment: the idea that a sentence follows logically from another sentence
  - Example:  $\alpha \models \beta$ , where  $\alpha$  &  $\beta$  are sentences and  $\beta$  follows from  $\alpha$

# Logic

15

- An inference algorithm that derives only entailed sentences is called sound or truth preserving
- Completeness is desirable
  - ▣ An inference algorithm is complete if it can derive any sentence that is entailed
- If knowledge base is true in the real world, then any sentence derived from the knowledge base by a sound inference procedure is also true in the real world

# Logic

16

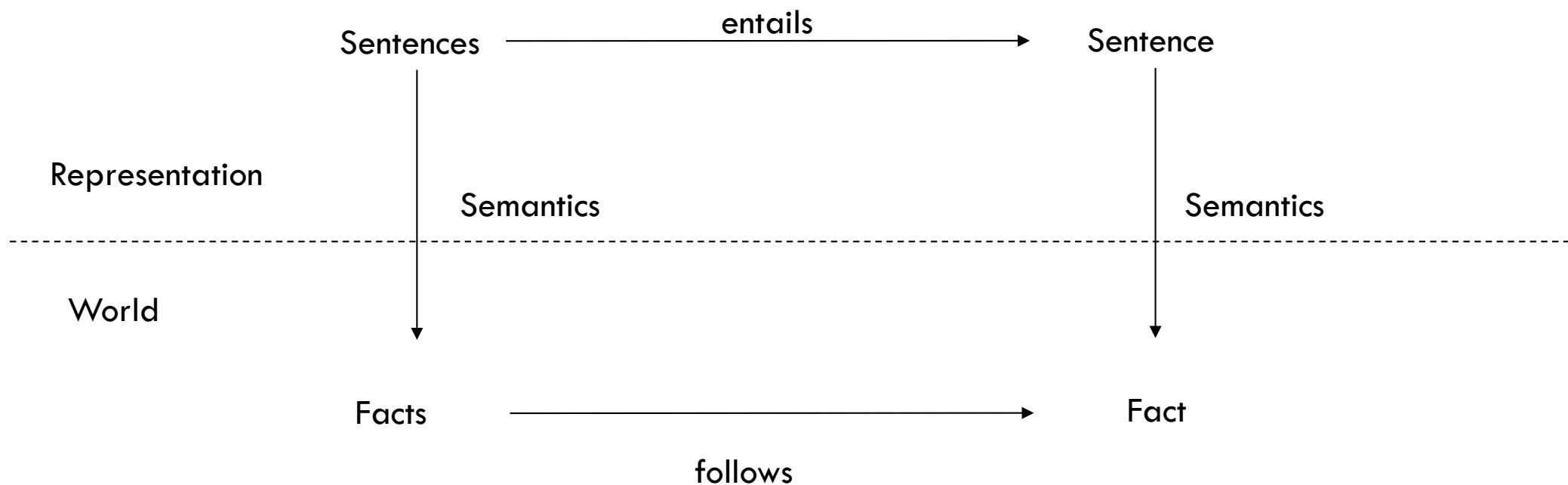


Figure: semantics map sentences in logic to fact in the world

# Logic

17

- Example
- Knowledge Base
  - Socrates is a man
  - All men are Mortal
  - All men are kind
- Inference algorithm is applied to the above base
- Inferring “Socrates is Mortal”
- “Socrates is kind” follows the sentence “All men are Kind”

# Truth Table

18

P	Q	$\neg P$	$P \vee Q$	$P \wedge Q$
False	False	True	False	False
False	True	True	True	False
True	False	False	True	False
True	True	False	True	True

# Tautology and Validity

19

- A notation used in formal logic which is always true and valid.
- Example: A OR (NOT A)  
I am eating food OR I am not eating food
- If all the conditions for a statement is true its tautology
- Tautologies are also called valid sentences

# Knowledge Models

20

- A model is a world in which a sentence is true under a particular interpretation
- There can be several models at once that have the same interpretations
- Types:
  - First order logic
  - Procedural Representation Model
  - Relational Representation Model
  - Hierarchical Representation Model
  - Semantic Nets

# Knowledge Models: Types

21

- First Order Logic
  - First Order Predicate Calculus
  - Consists of objects, predicates on objects, connectives and quantifiers
  - Predicates are the relations between objects or properties of the objects
  - Connectives and quantifiers allow for universal sentences
  - Relations between objects can be true or false
- Procedural Representation Model
  - This model of knowledge representation encodes facts along with the sequence of operations for manipulation and processing of the facts
  - Expert systems are based on this model
  - It works best when experts follow set of procedures for problem solving
  - Example: doctor making diagnosis

# Knowledge Models: Types

22

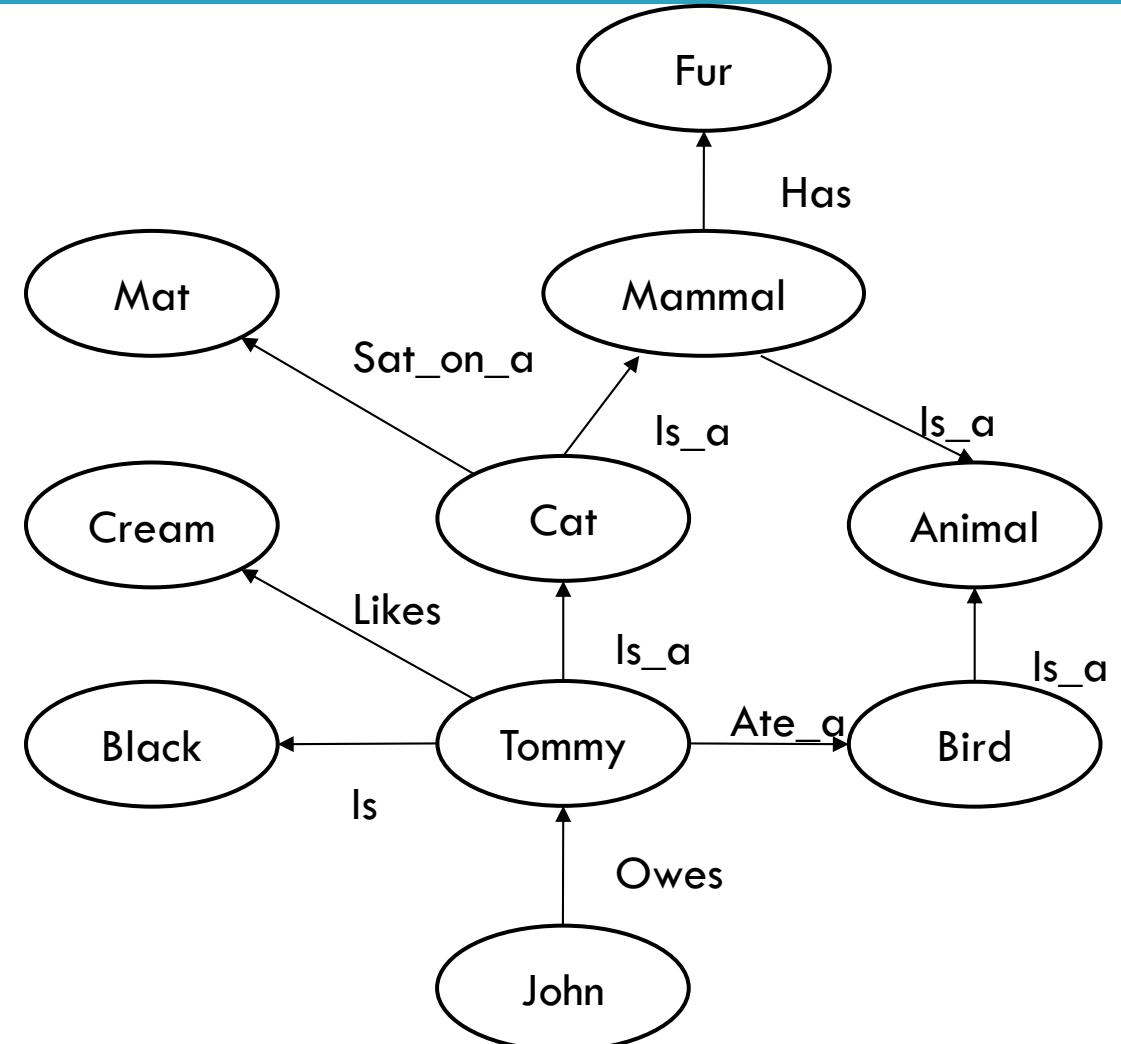
- Relational Representation Model
    - Collection of knowledge are stored in tabular form
    - Mostly used in commercial databases, relational databases
    - The information is manipulated with relational calculus use a language like SQL, Oracle, etc.
    - Its flexible way of storing information by not good for storing complex relationships
  - Problem arises when more than one subject area is attempted
  - A new knowledge base from scratch has to be built for each area of expertise
- 
- Hierarchical Representation Model
    - Based on inherited knowledge and the relationship and shared attributes between objects

# Knowledge Models: Types

23

## □ Semantic Nets

- Semantic networks are an alternative to predicate logic as a form of knowledge representation
- The idea is that we can store our knowledge in the form of graph with nodes representing objects in the world and are representing relationships between those objects



# Propositional Logic

24

- It is declarative sentences which can either be true or false but not both or neither
- A Very simple logic
- A Mathematical model that allows us to reason about the truth or falsehood of logical expressions
- There are sentences and connectives to describe an expression
- Its syntax defines allowable sentences
- Example:
  - Is it raining?
  - Is  $2+2=5$ ?
- Logical Connectives in Propositional Logic
  - $\wedge$  : Conjunction (and)
  - $\vee$  : Disjunction (or)
  - $\neg$  : Negation (not)
  - $\rightarrow \Rightarrow$  : Implication (if...then...)
  - $\Leftrightarrow \Leftarrow$  : Logical Equivalence (If and only If)

# Propositional Logic: Truth Tables

25

A	B	$A \wedge B$
F	F	F
F	T	F
T	F	F
T	T	T

A	B	$A \vee B$
F	F	F
F	T	T
T	F	T
T	T	T

A	$\neg A$
T	F
F	T

A	B	$A \Rightarrow B$
F	F	T
F	T	F
T	F	T
T	T	T

A	B	$A \Leftrightarrow B$
F	F	T
F	T	F
T	F	F
T	T	T

# Propositional Logic

26

- Sentence Properties
  - T or F itself is a sentence
  - Individual Proposition symbols are sentences  
eg. P, Q, ...
  - If s is a sentence, so is  $(s)$
  - If  $S_1$  and  $S_2$  are sentences, so are:  
 $\neg S_1$ ,  $\neg S_2$ ,  $S_1 \wedge S_2$ , etc.

- Order of Precedence
  - $\neg$ : Negation (not)
  - $\wedge$  : Conjunction (and)
  - $\vee$  : Disjunction (or)
  - $\Rightarrow \rightarrow$  : Implication  
(if...then...)
  - $\Leftrightarrow \leftrightarrow$  : Logical Equivalence (If and only If)

# Propositional Logic

27

- Atomic Sentences
  - Single sentence
  - T, F, P, Q,...  
where, each symbol stands for proposition that can be true or false.
  - Example: P=“Ram likes Rice”  
Q=“Sita is women”
- Complex Sentences
  - Sentences constructed from simple sentences using logical connectives
  - Example: P=“It is hot today”  
Q=“It is humid today”  
 $P \wedge Q$   
“It is hot and humid today”

# Propositional Logic

28

- Unsatisfiable (Contradiction)
  - If all the sentences or statements are always false
  - Example: “There will be a clear sky during rainy day”
- Satisfiable
  - If at least one sentence in the knowledge base is true

# Propositional Logic: Equivalence Laws

29

1.  $P \Rightarrow Q \equiv \neg P \vee Q$

2.  $P \Leftrightarrow Q \equiv (P \Rightarrow Q) \wedge (Q \Rightarrow P)$

3. Distributive Laws

$$A \wedge (B \vee C) \equiv (A \wedge B) \vee (A \wedge C)$$

$$A \vee (B \wedge C) \equiv (A \vee B) \wedge (A \vee C)$$

4. De-Morgan's Law

$$\neg(A \wedge B \wedge C) \equiv (\neg A) \vee (\neg B) \vee (\neg C)$$

$$\neg(A \vee B \vee C) \equiv (\neg A) \wedge (\neg B) \wedge (\neg C)$$

# Propositional Logic: Inference Rules

30

## 1. Modus Ponens Rule

Whenever any sentence of the form  $P \Rightarrow Q$  and P are given, then the sentence Q can be inferred

$$\frac{P \Rightarrow Q, P}{Q}$$

## 2. And Elimination

$$\frac{A \wedge B}{A | B}$$

sentence A or B can be inferred if A and B is given

## 3. And Introduction

$$\frac{A, B, \dots \dots \dots N}{A \wedge B \wedge \dots \wedge N}$$

## 4. Or Introduction

$$\frac{A, B, \dots \dots \dots N}{A \vee B \vee \dots \vee N}$$

## 5. Double Negation Elimination

$$\frac{\perp \perp P}{P}$$

# Propositional Logic: Inference Rules

31

6. Unit Resolution

$$\frac{A \vee B, \neg A}{B}$$

7. Modus Tollens

$$\frac{P \Rightarrow Q, \neg Q}{\neg P}$$

8. Resolution Chaining

$$\frac{P \Rightarrow Q, Q \Rightarrow R}{P \Rightarrow R}$$

$$\frac{\neg P \Rightarrow Q, Q \Rightarrow R}{\neg P \Rightarrow R}$$

# Propositional Logic

32

- The semantics defines the rules for determining the truth of sentences with respect to a particular model, i.e. semantic must specify how to compute the truth value of any sentence in a given model.

# Propositional Logic: BNF Grammar

33

- Backus Normal Form or Backus Naur Form
- It's a notation technique for context free grammars often used to describe the syntax of languages used in computing
- BNF can be used in two ways:
  - To generate strings belonging to the grammar
  - To recognize strings belonging to the grammar

# Normal Forms of Propositional Logic Sentences

34

1. Conjunctive (disjunction of conjunction of literals)

Normal Form

- In which a sentence is written as the conjunction of literals

$$(A \vee B) \Rightarrow Q$$

$$\equiv \neg(A \vee B) \vee Q$$

$$\equiv (\neg A \wedge \neg B) \vee Q$$

$$\equiv (\neg A \vee Q) \wedge (\neg B \vee Q)$$

2. Disjunctive (conjunction of disjunction of literals) Normal Form

- In which a sentence is written as the disjunction of literals

$$(A \wedge Q) \vee (B \wedge Q)$$

# First Order Predicate Logic (FOPL)

35

- Propositional logic assumes that the world or system being modelled can be described in terms of fixed, known set of propositions
- This assumption can make it awkward or even impossible to specify many pieces of knowledge
- Example:
  - Consider a general sentence “if a person is rich then they have a nice car”
  - In propositional logic, we can generate rule for each person as
    - $\text{Bob\_is\_rich} \rightarrow \text{Bob\_has\_a\_nice\_car}$
    - $\text{John\_is\_rich} \rightarrow \text{John\_has\_a\_nice\_car}$
  - This seems to be an impractical way to represent knowledge, hence, generalization to represent this type of knowledge is a must

# First Order Predicate Logic (FOPL)

36

- FOPL is a logic that gives us the ability to quantify over objects
- In FOPL, statements from a natural language like English are translated into symbolic structure composed of predicates, functions, variables, constants, quantifiers and logical connectives
- First Order Predicate Logic represents facts by separating classes and individuals and consider that world consists of different objects and relations between those objects

# FOPL: Syntax

37

Sentence	→	AtomicSentence   (Sentence Connective Sentence)   Quantifier Variable,...Sentence   $\neg$ Sentence
AtomicSentence	→	Predicate(Term,...)   Term = Term
Term	→	Function (Term,...)   Constant   Variable
Connective	→	$\neg$   $\vee$   $\wedge$   $\Rightarrow$   $\Leftarrow$
Quantifier	→	$\forall$   $\exists$
Constant	→	A   X   John   ...
Variable	→	a   x   s   ...
Predicate	→	Before   HasColor   Raining   ...
Function	→	Mother   Leftleg   ...

# FOPL: Syntax

38

- Constant Symbols are the strings that will be interpreted as representing objects
- Variable Symbols are used as place holders for quantifying over objects
- Predicate symbols are used to denote properties of objects and relationship among them
- Function Symbols map the specified number of input objects to objects
- Quantifiers are used to quantify objects
  - Universal Quantifier represents for all
  - Existential Quantifier represents the existence of an object

# FOPL: Variable Scope

39

- The scope of the variable is in the sentence to which the quantifier syntactically applies
- In a block structured programming language, a variable in a logical expression refers to the closest quantifier within whose scope it appears
- In a well formed formula all the variables should be properly introduced

# Relation Between Quantifiers

40

- $\forall x \neg P \equiv \neg \exists x P$
- $\neg \forall x P \equiv \exists x \neg P$
- $\forall x P \equiv \neg \exists x \neg P$
- $\exists x P \equiv \neg \forall x \neg P$
- $\forall x P(x) \cap Q(x) \equiv$   
 $\quad \forall x P(x) \cap \forall x Q(x)$
- $\exists x P(x) \cup Q(x) \equiv$   
 $\quad \exists x P(x) \cup \exists x Q(x)$

# Examples

41

- All birds can't fly  
 $\forall x \text{Bird}(x) \rightarrow \neg \text{Fly}(x)$   
OR  
 $\neg(\exists x (\text{Bird}(x) \cap \text{Fly}(x))$
- Not all birds can fly  
 $\neg(\forall x \text{Bird}(x))$   
OR  
Type equation here.
- If anyone can solve the problem then Raju can  
 $\exists x \text{Solves}(x, \text{problem}) \rightarrow \text{Solves}(\text{Raju}, \text{problem})$
- Try these
  - Nobody in electrical class is smarter than everyone in AI class
  - John hates all the people who don't hate themselves

# Equality

42

- Can include equality as a primitive predicate in the logic or require it to be introduced and axiomatized as the identity relation
- Useful in representing certain types of knowledge
  - Example: Sita owns two cars
$$\exists x \exists y (\text{Owns}(\text{Sita}, x) \cap \text{Owns}(\text{Sita}, y) \cap \text{Car}(x) \cap \text{Car}(y) \cap \neg(x = y))$$
- Try these:
  - There are exactly two purple flowers out of three
  - Everyone is married to exactly one person

**Every gardener likes the sun.**

$$\forall x \text{gardener}(x) \rightarrow \text{likes}(x, \text{Sun})$$

**You can fool some of the people all of the time.**

$$\exists x \forall t \text{person}(x) \wedge \text{time}(t) \rightarrow \text{can-fool}(x, t)$$

**You can fool all of the people some of the time.**

$$\forall x \exists t (\text{person}(x) \rightarrow \text{time}(t) \wedge \text{can-fool}(x, t))$$

$$\forall x (\text{person}(x) \rightarrow \exists t (\text{time}(t) \wedge \text{can-fool}(x, t)))$$

**All purple mushrooms are poisonous.**

$$\forall x (\text{mushroom}(x) \wedge \text{purple}(x)) \rightarrow \text{poisonous}(x)$$

**No purple mushroom is poisonous.**

$$\neg \exists x \text{ purple}(x) \wedge \text{mushroom}(x) \wedge \text{poisonous}(x)$$

$$\forall x (\text{mushroom}(x) \wedge \text{purple}(x)) \rightarrow \neg \text{poisonous}(x)$$

**There are exactly two purple mushrooms.**

$$\exists x \exists y \text{mushroom}(x) \wedge \text{purple}(x) \wedge \text{mushroom}(y) \wedge \text{purple}(y) \wedge \neg(x=y) \wedge \forall z (\text{mushroom}(z) \wedge \text{purple}(z)) \rightarrow ((x=z) \vee (y=z))$$

**Clinton is not tall.**

$$\neg \text{tall}(\text{Clinton})$$

**X is above Y iff X is on directly on top of Y or there is a pile of one or more other objects directly on top of one another starting with X and ending with Y.**

$$\forall x \forall y \text{above}(x,y) \leftrightarrow (\text{on}(x,y) \vee \exists z (\text{on}(x,z) \wedge \text{above}(z,y)))$$

# Try few more

45

- Ram likes all kinds of food
- Anything anyone eats and is not killed by is food
- Rita eats samosa and is still alive
- Gita eats everything Rita eats
- Someone who hates something owned by another person will not love that person
- There is a barber in the town who shaves all men in the town who don't shave themselves
- Everyone loves somebody
- No one likes everyone
- There is someone who is liked by everyone
- You can fool some of the people every time
- All employee earning Rs.200000/- or more per year pay taxes
- Some employee are sick today
- Nobody earns more than the chairman

# Horn Clause

46

- Disjunction of literals of which at most one is positive is Horn Clause

$$P_1 \cap P_2 \cap \dots \cap P_n \Rightarrow Q \equiv \neg P_1 \cup \neg P_2 \cup \dots \cup \neg P_n \cup Q$$

- Clause with exactly one positive literals giving definite clause (fact)
- Horn clause with no positive literals can be written as an implication whose conclusion is the literal false

$$\neg x_1 \cup \neg x_2 \equiv x_1 \cap x_2 \Rightarrow \text{False}$$

# Horn Clause

47

## Reason for its importance

- Every horn clause can be written as an implication whose premises is a conjunction of positive literals and whose conclusion is a single positive literal

Example:  $\neg L_1 \cup \neg L_2 \cup B$  can be written as  
 $L_1 \cap L_2 \Rightarrow B$

- Inference with horn clauses can be done with the forward chaining and backward chaining
- Deciding entailment with horn clauses can be done in time that is linear in the size of knowledge base

# Well Formed Formula

48

- A sentence that has all its variables properly introduced using quantifiers is a well formed formula
- Example:  
 $\forall xP(x, y)$  is not a well formed formula where x is bounded as universal quantifier and y is free  
 $\forall x\exists yQ(x, y)$  is a well formed formula where both x and y are bounded

## □ Notes:

- Predicate can't be quantifiers
- Constant can't be negative
- Letter cases must be well considered

# Inference in FOL

49

- If  $x$  is a parent of  $y$ , then  $x$  is older than  $y$
- If  $x$  is the mother of  $y$  then  $x$  is a parent of  $y$
- Devaki is the mother of Krishna
- Conclusion:  
Devaki is older than Krishna

## Mapping in FOL

- $\forall x \forall y \text{parent}(x, y) \Rightarrow \text{older}(x, y)$
- $\forall x \forall y \text{mother}(x, y) \Rightarrow \text{parent}(x, y)$
- $\text{mother}(\text{Devaki}, \text{Krishna})$
- Conclusion:  
 $\text{older}(\text{Devaki}, \text{Krishna})$

# Inference Rules in FOL

50

- Universal Instantiation
  - If a person is a student, studies in KEC and studies AI, then he/she is a third year student
  - $\forall x \text{student}(x) \cap \text{studiesin}(x, \text{KEC}) \cap \text{studies}(x, \text{AI}) \Rightarrow \text{thirdyearstudent}(x)$
- Existential Instantiation
  - There must be a topper in KEC
  - $\exists x \text{student}(x) \cap \text{studiesin}(x, \text{KEC}) \cap \text{topper}(x)$
- Propositionization
  - All people are kind  
 $\forall x \text{person}(x) \Rightarrow \text{kind}(x)$   
It can be inferred as  
 $\text{person}(\text{Ram}) \Rightarrow \text{kind}(\text{Ram})$

# Inference Rules in FOL

51

## □ Generalized Modus Ponens

- $\forall x \text{student}(x) \cap$   
 $\text{studieshard}(x) \Rightarrow$   
 $\text{good student}(x)$
- $\text{student}(\text{Arjun})$
- $\text{studieshard}(\text{Arjun})$
- Conclusion:  
 $\text{goodstudent}(\text{Arjun})$

## □ Unification

- $[\text{knows}(\text{Sita}, x), \text{knows}(\text{Sita}, \text{Rita})]$   
 $x = \text{Rita}$
- $[\text{knows}(\text{Sita}, x), \text{knows}(y, \text{Rita})]$   
 $\Rightarrow x = \text{Rita}, y = \text{Sita}$
- $[\text{knows}(\text{Sita}, x),$   
 $\text{knows}(y, \text{ismother}(y)) \Rightarrow y$   
 $= \text{Sita}, x = \text{mother}(\text{Sita})$
- $[\text{knows}(\text{Sita}, x), \text{knows}(x, \text{Rita})]$   
 $\Rightarrow \text{false}$

# Inference Rules in FOL

52

- Resolution
  - Produces proof by refutation (proof person or statement that is wrong)
  - Resolution can be applied to sentences in CNF (conjunctive normal form)
- Process of Resolution
  - Convert all sentences to CNF
  - Negate x
  - Add negate x to premises
  - Repeat until either a contradiction is detected or no progress is being made

# CNF Conversion Process

53

1. Elimination of all implications with equivalence symbols

- $P \rightarrow Q \equiv \neg P \cup Q$

- $P \Leftrightarrow Q \equiv (\neg P \cup Q) \cap (\neg Q \cup P)$

2. Move  $\neg$  inward (use De'Morgans law)

- $\neg(P \cap Q) \equiv \neg P \cup \neg Q$

- $\neg(P \cup Q) \equiv \neg P \cap \neg Q$

- $\forall x \neg P \equiv \neg \exists x P$

- $\neg \forall x P \equiv \exists x \neg P$

- $\forall x P \equiv \neg \exists x \neg P$

- $\exists x P \equiv \neg \forall x \neg P$

3. Standardize Variables

- Rename variables if necessary so that all quantifiers have different variable assignments

# CNF Conversion Process

54

## 4. Skolemization

- The process of eliminating the existential quantifiers through a substitution process
- The process requires that all such variables be replaced by short term functions, which can always assume a Skolem function, a correct value required for an existential quantifier variable

- If leftmost quantifier in an expression is existential quantifier ( $\exists$ ), replace all occurrence of the variables that quantifies with an arbitrary constant not appearing elsewhere in the expression and delete the quantifier
- Example:  $\exists x \exists y \forall z P(x, y, z) \cup Q(x, y) \equiv \forall z P(a, b, z) \cup Q(a, b)$

# CNF Conversion Process

55

## 4. Skolemization

- If existential quantifier ( $\exists$ ) is preceded by universal quantifier ( $\forall$ ), replace the existentially quantified variable by a function symbol whose arguments are variable appearing in those universal quantifiers

### □ Example:

- $$\begin{aligned} & \exists u \forall x \forall y \exists z P(f(u), x, y, z) \\ & \quad \cup Q(x, y, z) \\ \equiv & \forall x \forall y \exists z P(f(a), x, y, z) \\ & \quad \cup Q(x, y, z) \\ \equiv & \forall x \forall y P(f(a), x, y, f(x, y)) \\ & \quad \cup Q(x, y, f(x, y)) \end{aligned}$$
- 5. Drop all universal quantifiers
  - 6. Distribute  $\wedge$  over  $\vee$

# Example: Given Premises

56

1. If  $x$  is on top of  $y$ ,  $y$  supports  $x$
2. If  $x$  is above  $y$  and they are touching each other,  $x$  is on top of  $y$
3. Everything is on top of another thing
4. A cup is above a book
5. A cup is touching a book

Answer:

Is the book supporting the cup?

# Example: Solution

57

- $\forall x \forall y \text{ ontop}(x, y) \Rightarrow \text{supports}(y, x)$   
*Implication Elimination*  
 $\forall x \forall y \neg \text{ontop}(x, y)$   
 $\Rightarrow \text{supports}(y, x)$   
*Drop  $\forall x$  and  $\forall y$*   
 $\neg \text{ontop}(x, y)$   
 $\Rightarrow \text{supports}(y, x)$

- $\forall x, y \text{ above}(x, y) \cap \text{touch}(x, y) \Rightarrow \text{ontop}(x, y)$   
*Implication Elimination*  
 $\forall x, y \neg \text{above}(x, y)$   
 $\cup \neg \text{touch}(x, y)$   
 $\cup \text{ontop}(x, y)$   
*Drop  $\forall x, y$*   
 $\neg \text{above}(x, y) \cup \neg \text{touch}(x, y)$   
 $\cup \text{ontop}(x, y)$

# Example: Solution

58

- $\forall x, y \text{ ontop}(x, y)$

*Drop*  $\forall x, y$   
*ontop*( $x, y$ )

- *above*(*cup*, *book*)
- *touch*(*cup*, *book*)

# Solution

59

## Conclusion

- $\text{supports}(\text{book}, \text{cup})$

Let

$\neg\text{supports}(\text{book}, \text{cup})$

using second and fifth conditions

$\neg\text{above}(x, y) \cup \neg\text{touch}(x, y)$

$\cup \text{ontop}(x, y)$

$\text{touch}(\text{cup}, \text{book})$

$\neg\text{above}(x, y) \cup \text{ontop}(x, y)$

using fourth condition

$\text{above}(\text{cup}, \text{book})$

$\text{ontop}(x, y)$

using first condition

$\neg\text{ontop}(x, y) \cup \text{supports}(y, x)$

$\text{supports}(\text{book}, \text{cup})$

using assumed condition

$\neg\text{supports}(\text{book}, \text{cup})$

Empty Clause

Hence, the book is supporting the cup

# Try these

60

□ Every American who sells weapon to hostile nation is a criminal. The country Iraq is an enemy of America. All of the missiles in Iraq were sold by George. George is an American.

Prove:

George is a Criminal

□ All Pompeians are Romans. All Romans were either loyal to Caesor or hated him. Everyone is loyal to someone. People only try to assassinate rulers they are not loyal to. Marcus tried to assassinate Caesor. Marcus was a Pompeian.  
Conclude:  
Did Marcus hate Caesor?

# Forward Chaining

61

- One of the two main methods for reasoning using inference rules
- Can be described logically as repeated application of Modus Ponens
- It's a popular strategy of reasoning in expert system and production systems
- It starts with the available data and uses inference rules to extract more data until a goal is reached
- An inference engine using forward chaining searches the inference rules until it finds one where antecedent (If clause) is known to be true

# Forward Chaining

62

- When it found if clause it can conclude or infer the consequent (then clause) to its data resulting in the addition of new information
- Example: (Animal Identification System)  
If X croaks and eats flies then it's a frog  
If X chirps and sings then it's a canary
- If X is a frog then X is green  
If X is a canary then X is yellow  
goal: colour of pet given that it croaks and eat flies

# References

63

- Russell, S. and Norvig, P., 2011, Artificial Intelligence: A Modern Approach, Pearson, India.
- Rich, E. and Knight, K., 2004, Artificial Intelligence, Tata McGraw hill, India.

**64**

# Thank You

Any Queries?

Now, Search for yourself.

# UNIT 3

Knowledge Representation

# Outline

- Representations and Mappings
- Approaches to Knowledge Representation
- Issues in Knowledge Representation
- Semantics Net
- Frames
- Conceptual Dependencies
- Scripts

# Knowledge

- Data
- Information
- Knowledge
  
- Tacit Knowledge
- Recorded Knowledge
  
- Knowledge Engineering

# Representations and Mappings

- Solving Complex Problems is guided by requirement of large amount of knowledge and some mechanisms to manipulate that knowledge and create the solution to the Problem
- For that knowledge is to be represented for which the following points are to be considered
  - ▣ Facts: that we want to represent, i.e. the truth in the representing world
  - ▣ Representation: in some formal way

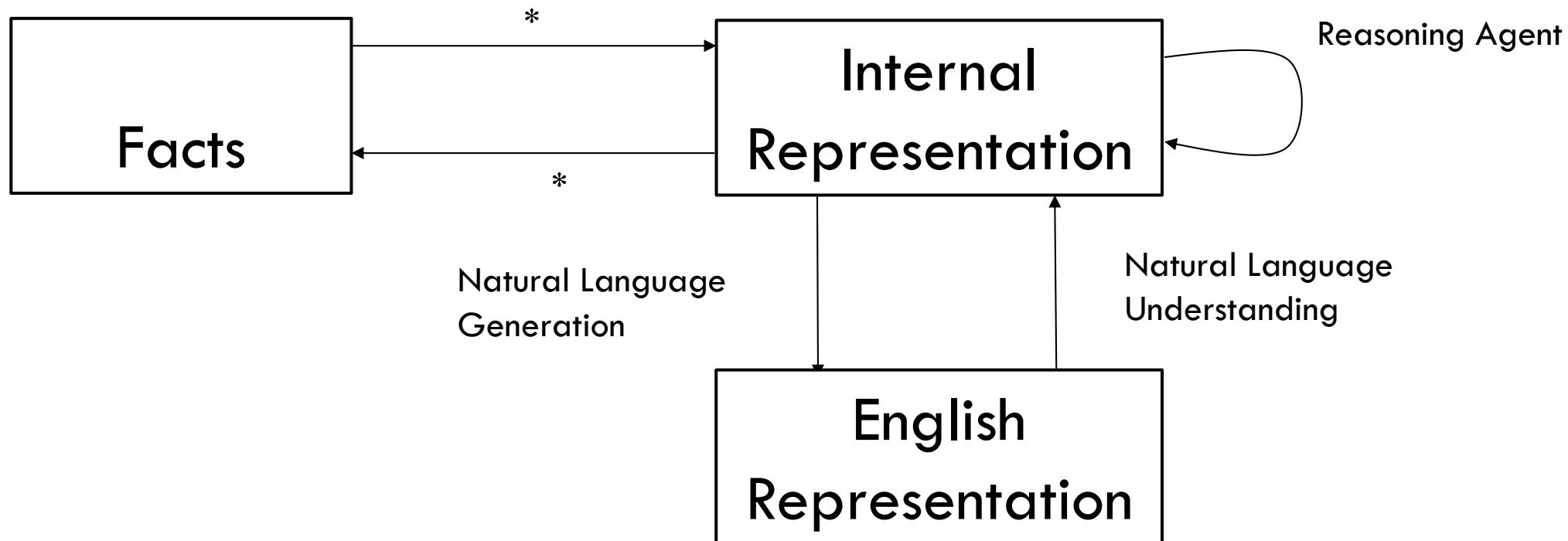
# Representations and Mappings

- For structuring these entities one way is to think at two levels:
  - The knowledge level
  - The symbol level
- At knowledge level facts are described
- At symbol level objects represented at knowledge level are defined in terms of symbols that can be manipulated by programs

# Representations and Mappings

## (Reviewing)

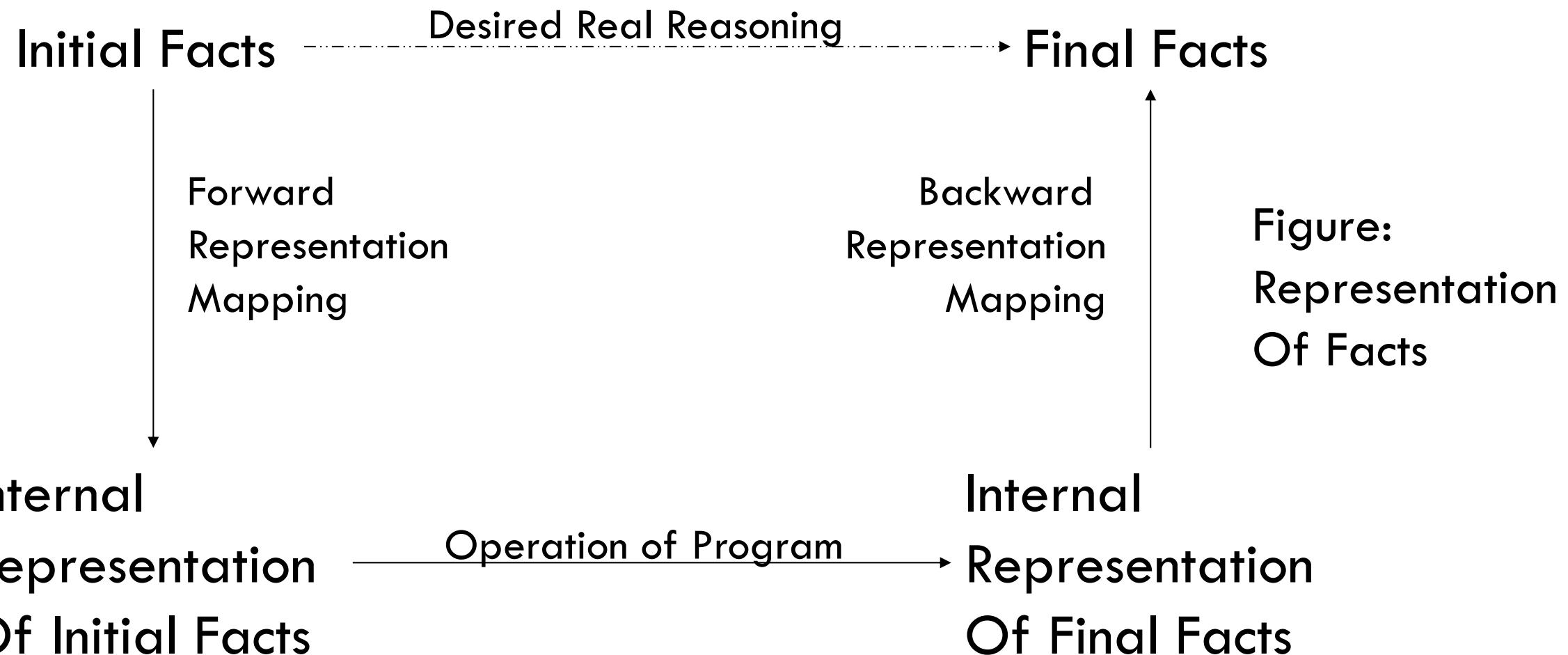
Figure: Mapping Facts and Representation



# Representations and Mappings

- Rather than thinking of one level on top of another, focusing on facts, representations and on the two way mappings that must exist between them is more important
- These links are called Representation Mappings
- Forward Representation Mapping maps facts to representations
- Backward Representation Mapping maps representations to facts

# Representations and Mappings



# Approaches to Knowledge Representation

For a good system following four properties are must

- Representational Adequacy: Ability to represent all kind of knowledge that are needed in the domain
- Inferential Adequacy: Ability to manipulate the representational structure in such a way as to derive new structures corresponding to new knowledge inferred from old
- Inferential Efficiency: Ability to incorporate into the knowledge structure additional information that can be used to focus the attention of the inference mechanism in the most promising direction
- Acquisitional Efficiency: Ability to acquire new information easily

# Knowledge Representation: Types

- Simple Relational Knowledge
  - The simplest way to represent declarative facts is as a set of relations of the same sort used in database system
- Inheritable Knowledge
  - Structure must be designed to correspond to the inference mechanism that are desired
- Inferential Knowledge
  - Represents knowledge as formal logic
  - Based on reasoning from facts or from other inferential knowledge
  - Useless unless there is also an inference procedure that can exploit it
- Procedural (Imperative) Knowledge
  - Knowledge exercised in the performance of some task
  - Processed by an intelligent agent

# Issues in Knowledge Representation

- Are any attributes of objects so basic that they have been occurred in almost every problem domain?
- Are there any important relationships that exist among attributes of objects
- At what level should knowledge be represented?
- How should sets of objects be represented?
- How can relevant parts be accessed when they are needed?

# Semantic Networks

- Other than descriptive logic, the major system designed to organise and for reasoning
- Evolved from existential graphs, called the logic of the future
- Existential graphs uses a graphical notation of nodes and arcs
- Semantic networks provide for certain kinds of sentences is often more convenient but if we strip away the human interface issues, the underlying concept persist with objects, relations, quantification and so on

# Semantic Networks

- Many variant of semantic net are available now a days
- Semantic nets are capable of representing individual objects, categories of objects and relationships among those objects
- A typical graphical notation displays object or categories names in ovals or boxes and connects them with labelled arcs
- Suitable for implementing inheritance and object oriented concepts
- Inverse Links: Example → Brother of (x, y) = Has brother (y, x)

# Semantic Networks

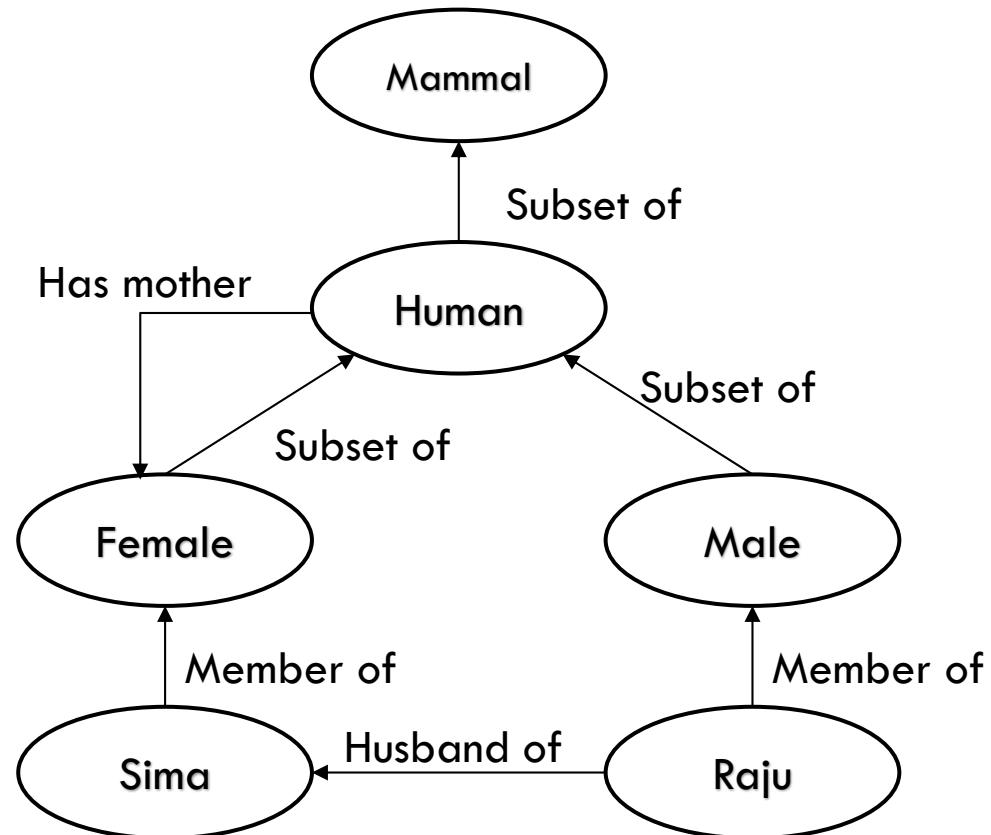


Figure: Semantic Network Example

# Frames

- A frame is a collection of attributes (slots) and associated values and possibly constraints on values that describe some entity in the world.
- Frame system is build on a set of frames that are connected to each other by the virtue of fact that the value of an attribute of one frame may be another frame
- Generally, Frames are based on set theory

# Frames

Example:

Human

Isa: Mammal

Cardinality: 6000000000

\*Legs: 2

Male

Isa: Human

Cardinality: 4000000000

\*Hair: Short

# Scripts

- A script is a structure that is used to describe the sequence of events in a particular context
- It consists of a set of slots
- Each slot is associated with some information describing the kind of values a slot may contain as well as a default value to be used if no other information is available
- Script seems to be similar to frames but these have more detailed information
- For example: refer to Page Number 286, Artificial Intelligence, Rich and Knight.

# References

- Russell, S. and Norvig, P., 2011, Artificial Intelligence: A Modern Approach, Pearson, India.
- Rich, E. and Knight, K., 2004, Artificial Intelligence, Tata McGraw hill, India.



**Thank You**

**Any Queries?**

# UNIT 4

Inference and Reasoning

# Contents

2

- Inference Theorems
- Deduction and Truth Maintenance
- Heuristic Search state-space Representation
- Game Playing
- Reasoning About Uncertainty Probability
- Bayesian Network
- Case Based Reasoning

# Heuristic Search State Space Representation

3

- 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

4

- 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

5

- **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

6

A restricted state-transition system is a triple

$\Sigma=(S, A, \gamma)$ , where:

- $S=\{s_1, s_2, \dots\}$  is a set of states;
- $A=\{a_1, a_2, \dots\}$  is a set of actions;
- $\gamma: S \times A \rightarrow S$  is a state transition function.

# Heuristic Search State Space Representation

7

## Search Problems:

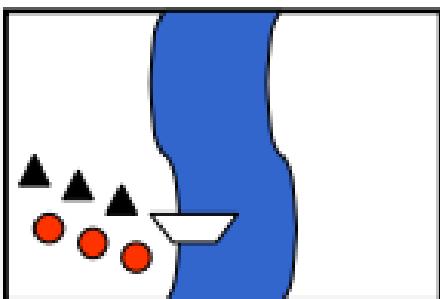
- initial state
- set of possible actions/applicability conditions
  - successor function:  $state \rightarrow$  set of  $\langle action, state \rangle$
  - successor function + initial state = state space
  - path (solution)
- goal
  - goal state or goal test function
- path cost function
  - for optimality
  - assumption: path cost = sum of step costs

# Heuristic Search State Space Representation

8

## 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

9

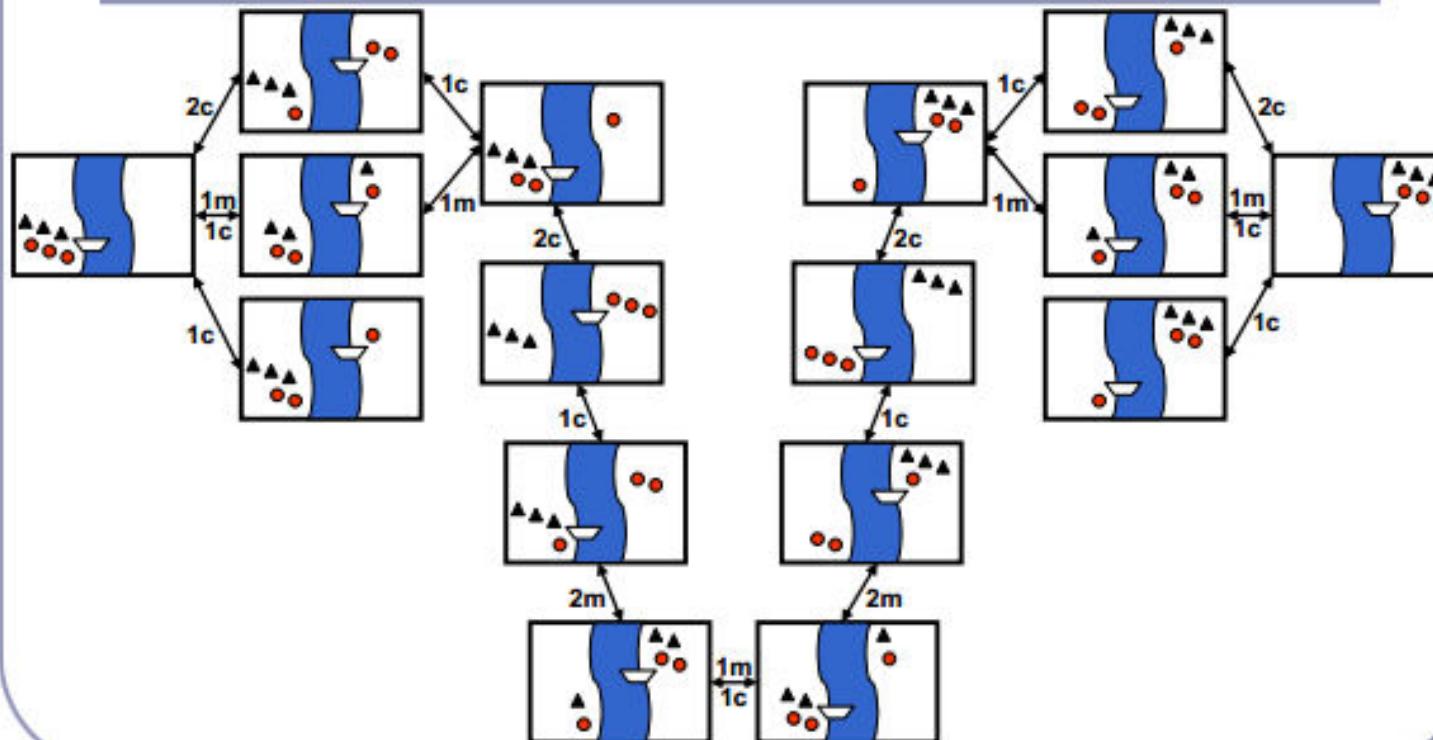
## Missionaries and Cannibals: Successor Function

<i>state</i>	set of <i>&lt;action, state&gt;</i>
$(L:3m, 3c, b - R:0m, 0c) \rightarrow$	$\{<2c, (L:3m, 1c - R:0m, 2c, b)>, <1m1c, (L:2m, 2c - R:1m, 1c, b)>, <1c, (L:3m, 2c - R:0m, 1c, b)>\}$
$(L:3m, 1c - R:0m, 2c, b) \rightarrow$	$\{<2c, (L:3m, 3c, b - R:0m, 0c)>, <1c, (L:3m, 2c, b - R:0m, 1c)>\}$
$(L:2m, 2c - R:1m, 1c, b) \rightarrow$	$\{<1m1c, (L:3m, 3c, b - R:0m, 0c)>, <1m, (L:3m, 2c, b - R:0m, 1c)>\}$
⋮	⋮

# Heuristic Search State Space Representation

10

## Missionaries and Cannibals: State Space

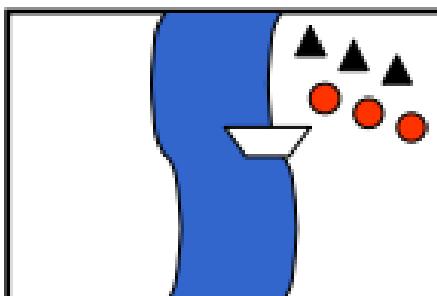


# Heuristic Search State Space Representation

11

## Missionaries and Cannibals: Goal State and Path Cost

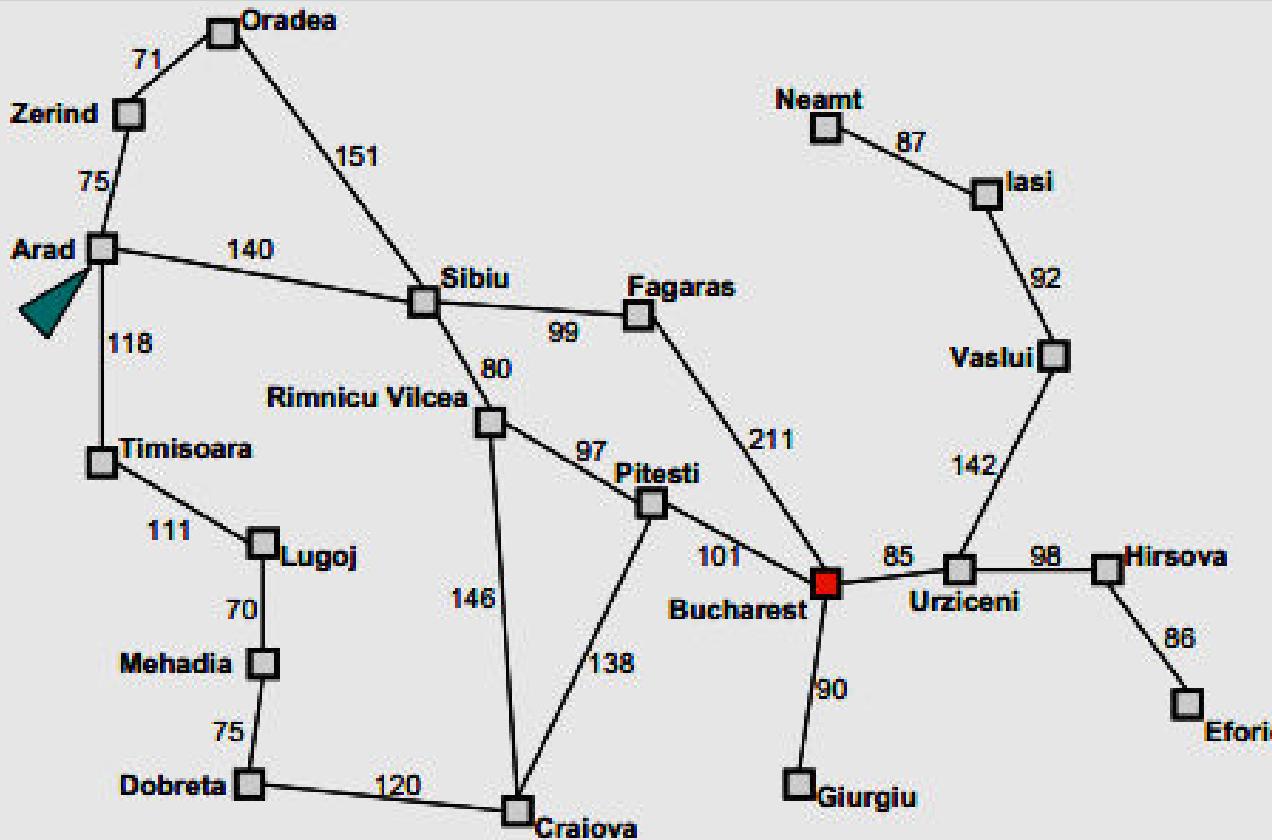
- goal state:
  - all missionaries, all cannibals, and the boat are on the right bank
- path cost
  - step cost: 1 for each crossing
  - path cost: number of crossings = length of path
- solution path:
  - 4 optimal solutions
  - cost: 11



# Heuristic Search State Space Representation

12

## Real-World Problem: Touring in Romania



# Heuristic Search State Space Representation

13

## Touring Romania: Search Problem Definition

---

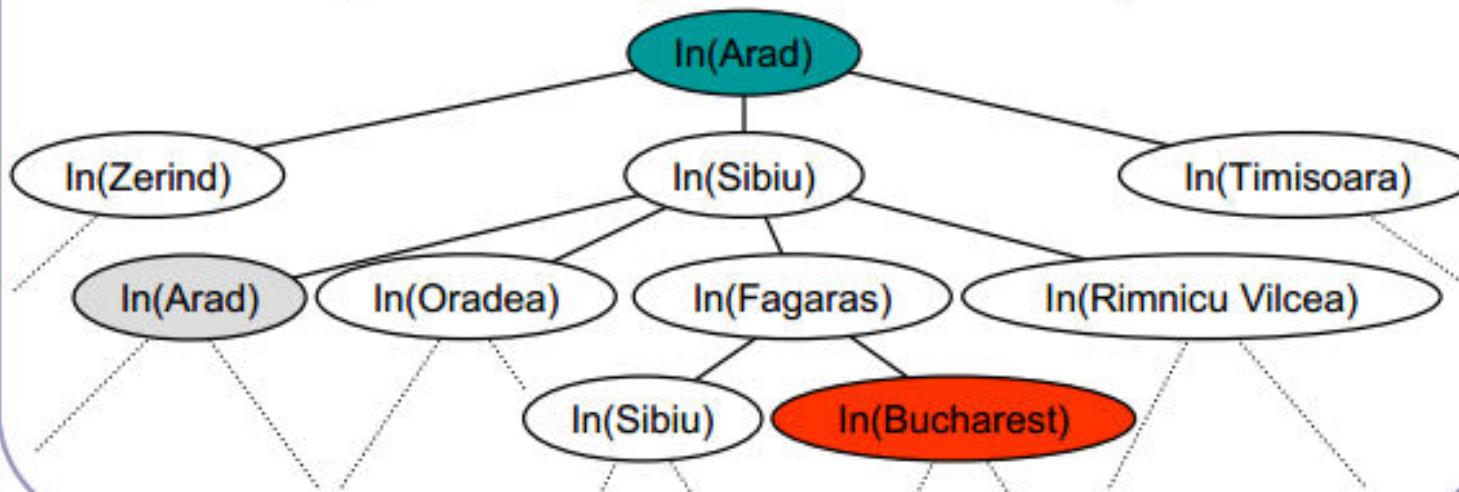
- initial state:
  - In(Arad)
- possible Actions:
  - DriveTo(Zerind), DriveTo(Sibiu), DriveTo(Timisoara), etc.
- goal state:
  - In(Bucharest)
- step cost:
  - distances between cities

# Heuristic Search State Space Representation

14

## Search Trees

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



# Heuristic Search State Space Representation

15

## Search Nodes

---

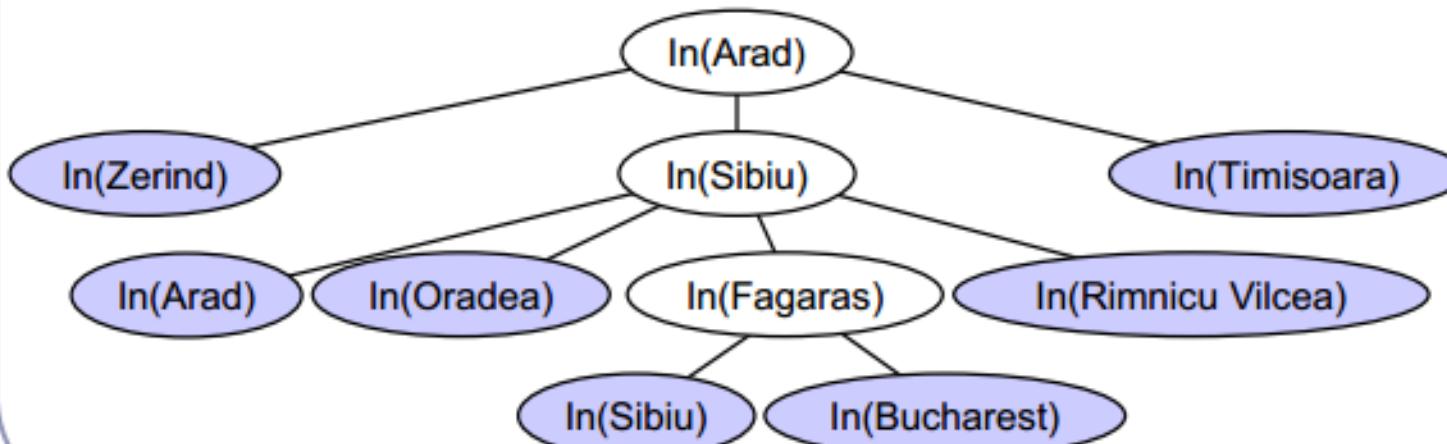
- search nodes: the nodes in the search tree
- data structure:
  - *state*: a state in the state space
  - *parent node*: the immediate predecessor in the search tree
  - *action*: the action that, performed in the parent node's state, leads to this node's state
  - *path cost*: the total cost of the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

# Heuristic Search State Space Representation

16

## Fringe Nodes in Touring Romania Example

fringe nodes: nodes that have not been expanded



# Heuristic Search State Space Representation

17

## **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

18

## General Tree Search Algorithm

```
function treeSearch(problem, strategy)
    fringe  $\leftarrow$  { new
        searchNode(problem.initialState) }

    loop
        if empty(fringe) then return failure
        node  $\leftarrow$  selectFrom(fringe, strategy)
        if problem.goalTest(node.state) then
            return pathTo(node)
        fringe  $\leftarrow$  fringe + expand(problem, node)
```

# Heuristic Search State Space Representation

19

## **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

20

## Best-First Search

- an instance of the general tree search or graph search algorithm
  - strategy: select next node based on an evaluation function  $f$ : 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

21

## 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

22

## 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

23

## Touring in Romania: Heuristic

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

Arad	366	Hirsova	151	Rimnicu	193
Bucharest	0	Iasi	226	Vilcea	
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

24

## 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:
$$f(n) = \text{estimated cost of the cheapest solution through } n$$
- A\* search is optimal if  $h(n)$  is admissible

# Heuristic Search State Space Representation

25

## Admissible Heuristics

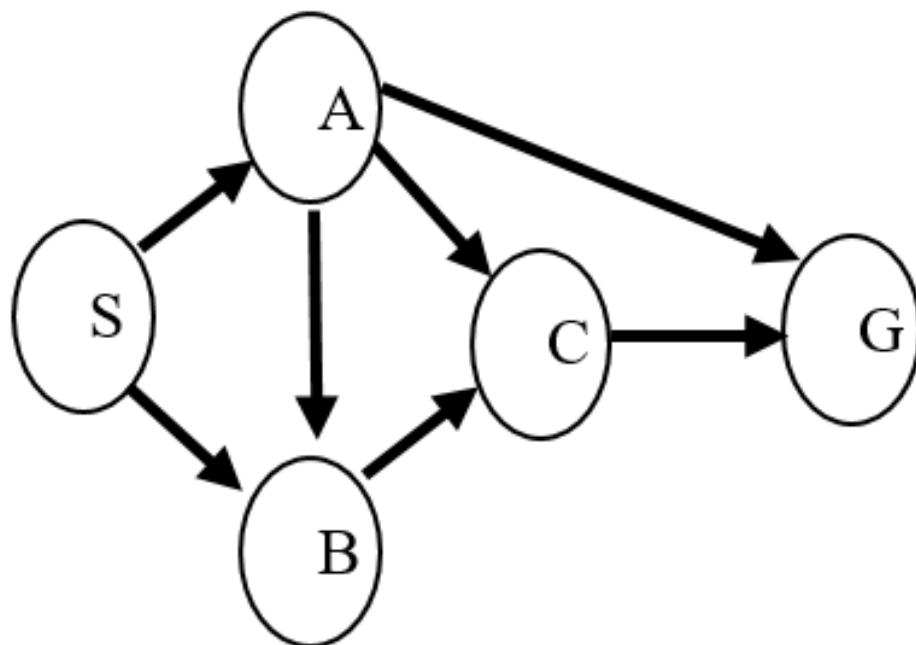
A heuristic  $h(n)$  is admissible if it *never overestimates* the distance from  $n$  to the nearest goal node.

- 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

26

## □ Example...



	S	A	B	C	G	State	H(n)
S		1	4				7
A			2	5	12		6
B				2			2
C					3		1
G							0

# Game Playing

27

- 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

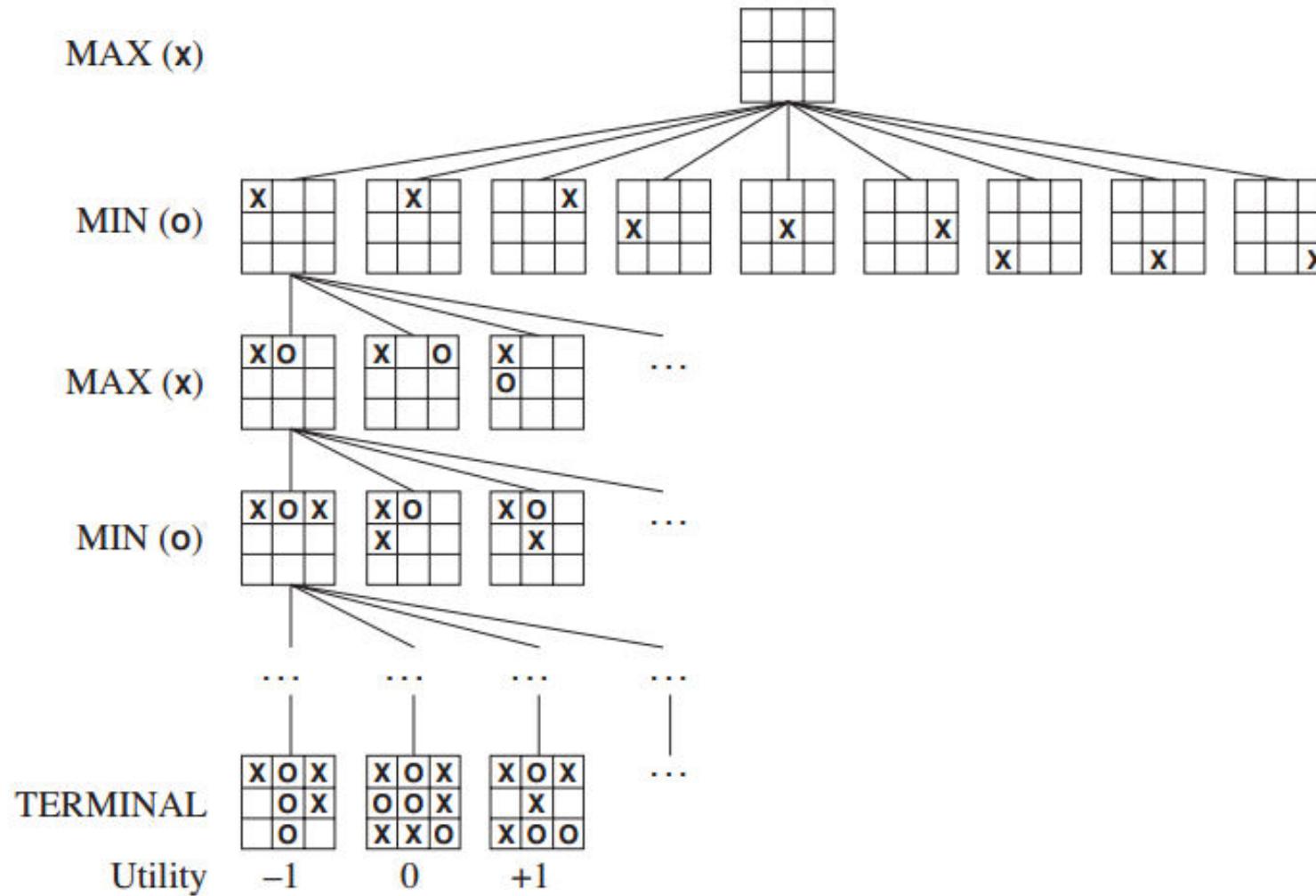
28

## 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

29



# Game Playing

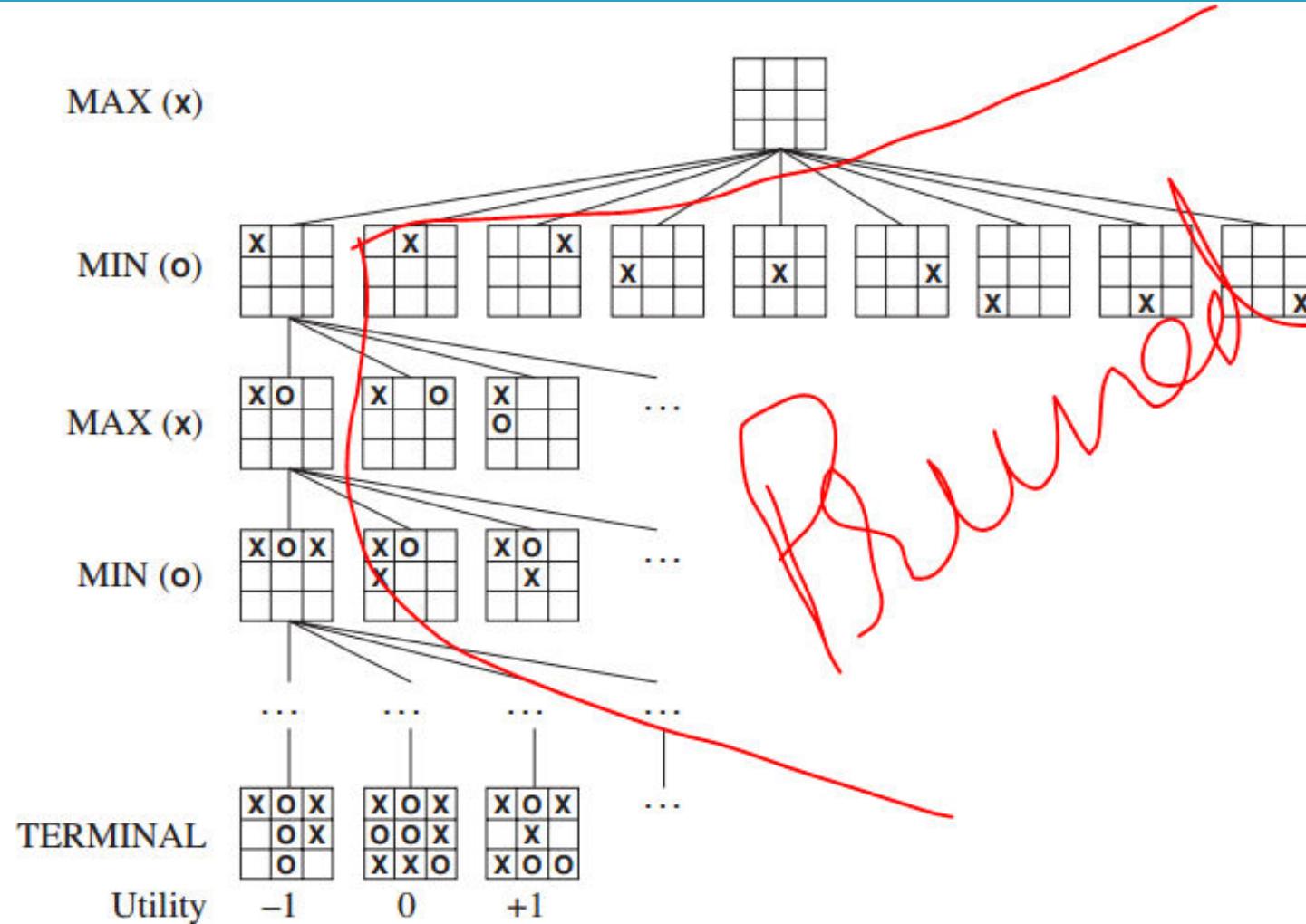
30

## 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

31



# Reasoning about Uncertainty Probability

32

- 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

33

## 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

34

## 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

35

- 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 least two possible outcomes: favorable outcome or success and unfavorable outcomes or failure

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

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

# Reasoning about Uncertainty Probability

36

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

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

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

and

$$p + q = 1$$

# Reasoning about Uncertainty Probability

37

- 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

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

Likewise, the probability of not getting 6 is

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

# Reasoning about Uncertainty Probability

38

- 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

39

- 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**

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

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

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

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

# Reasoning about Uncertainty Probability

40

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:

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

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

41

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

$$p(A \cap B_1) = p(A|B_1) \times p(B_1)$$

$$p(A \cap B_2) = p(A|B_2) \times p(B_2)$$

⋮

$$p(A \cap B_n) = p(A|B_n) \times p(B_n)$$

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

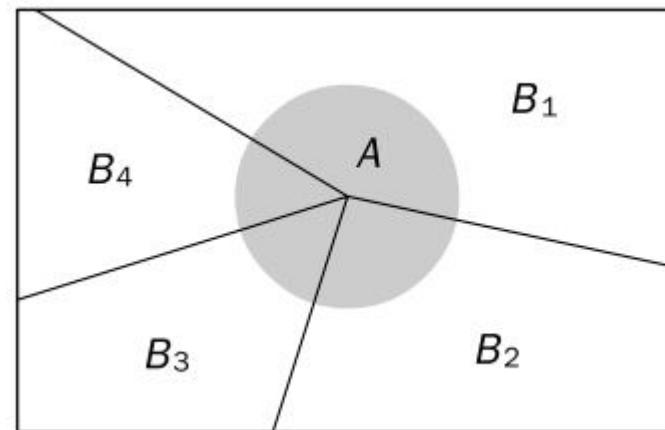
42

- Summed over an exhaustive list of events for  $B_i$ , 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

43

- 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

44

## 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

45

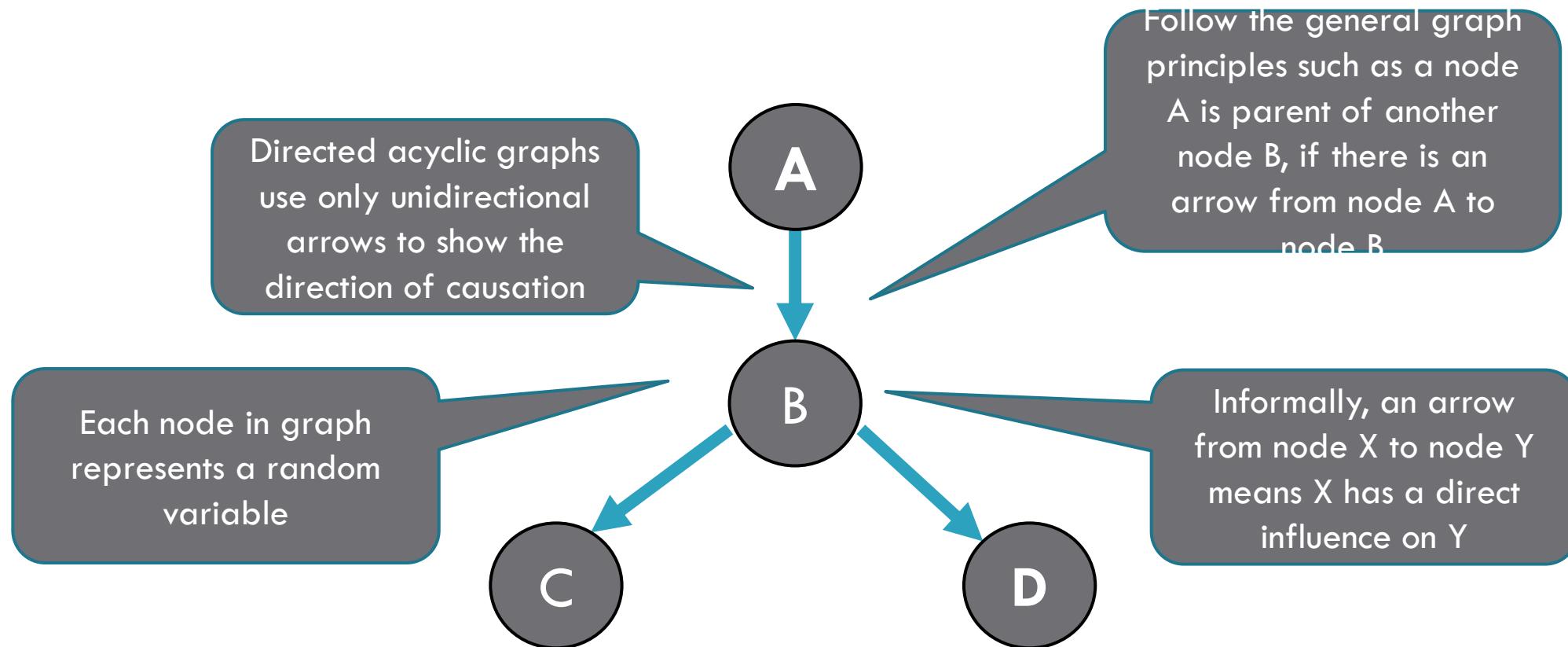
- Bayesian Network are also known as **Bayes Network, Belief Networks and Probabilistic Networks**
- A BN is defined by two parts, **Directed Acyclic Graph (DAG)** and **Conditional Probability Tables (CPT)**

Nodes → Random Variables

Arcs → Indicates Probabilistic dependencies between nodes

# Bayesian Networks

46



# Bayesian Networks

47

**A BN is a directed graph with the following properties:**

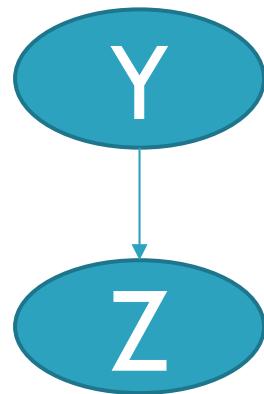
- **Nodes:** Set of Random Variables which may be discrete or continuous
- **Directed Links (Arcs) :** The real meaning od a link from node X to node Y is that X has a direct influence on Y
- Each node has a Conditional Probability Distribution  $P(X_i | Parents(X_i))$  that quantifies the effects that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

# Bayesian Networks

48

**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

49

## Incremental Network Construction:

1. **Nodes:** First determine the set of variables that are required to model the domain. Now order them,  $\{X_1, X_2, \dots, X_n\}$ . 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  $n$  do:
  1. Choose, from  $X_1, \dots, X_{i-1}$ , a minimal set of parents for  $X_i$  such that equation  $\mathbf{P}(X_i | X_{i-1}, \dots, X_1) = \mathbf{P}(X_i | \text{Parents}(X_i))$  is satisfied
  2. For each parent insert a link from the parent to  $X_i$
  3. CPTs: Write down the Conditional Probability Table,  $\mathbf{P}(X_i | \text{Parents}(X_i))$

# Bayesian Networks

50

Conditional Independence:

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

A BN represents Conditional Independence

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

# Bayesian Networks

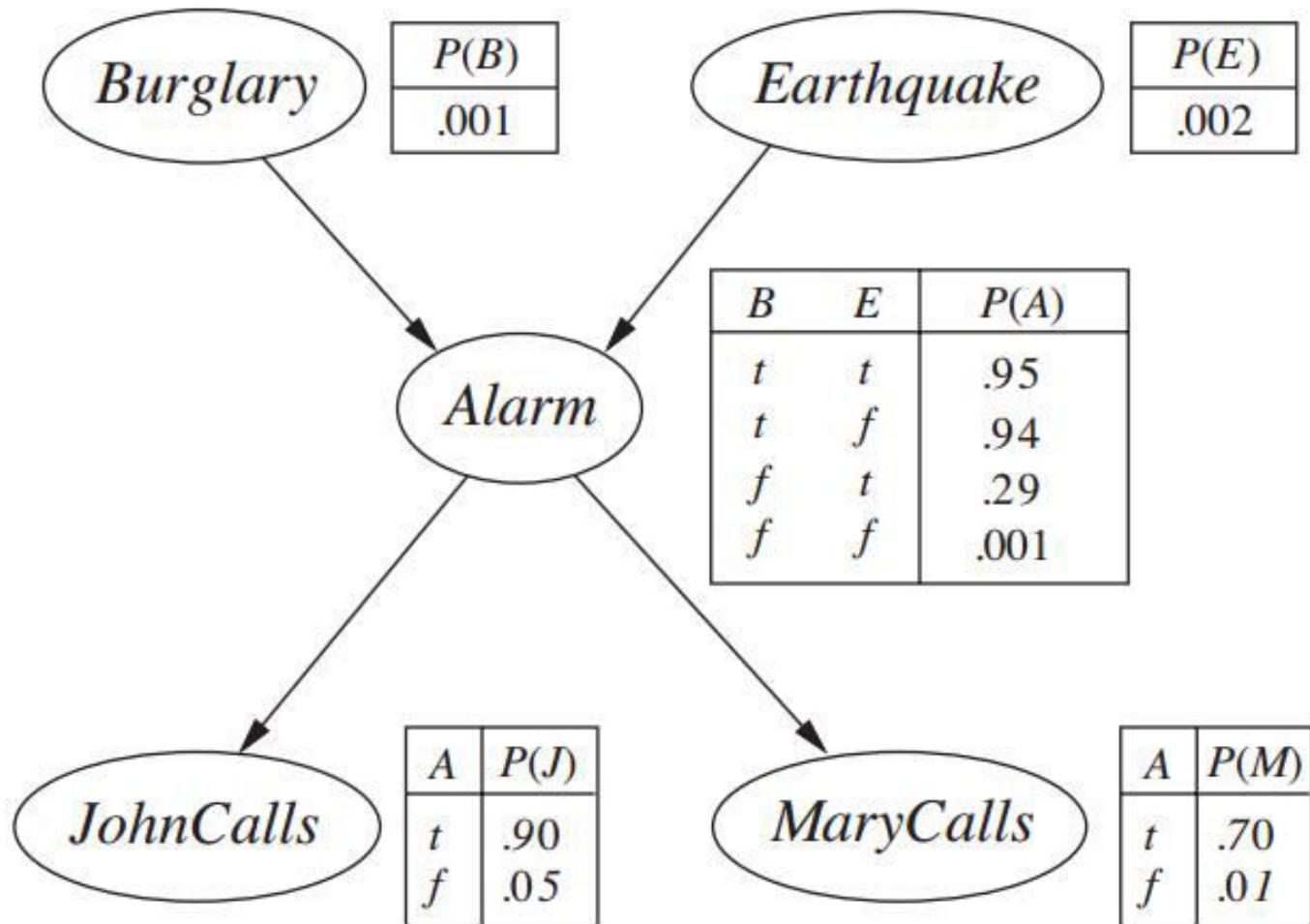
51

## 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

52



# Bayesian Networks

53

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

$$P(J | B) = ?$$

first calculate probability of Alarm ringing on burglary:

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

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

$$P(A | B) = 0.94$$

Now, Let us calculate  $P(J | B)$

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

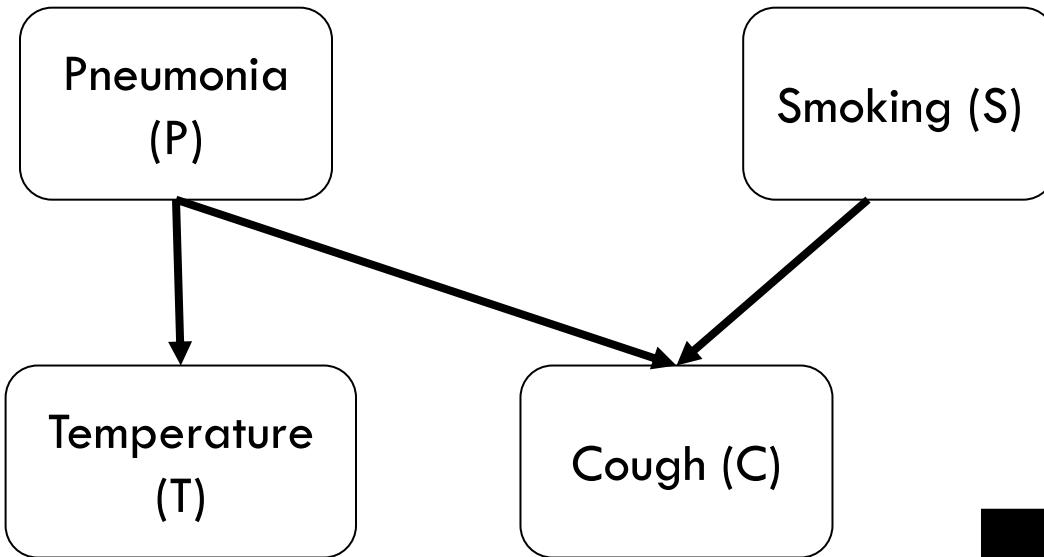
$$P(J | B) = (0.94) * (0.9) + (0.06) * (0.05) = 0.85$$

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

# Bayesian Network

54

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

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:

- $P(C | S \wedge P)$
- $P(S | C)$

# Bayesian Networks

55

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

56

- 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

57

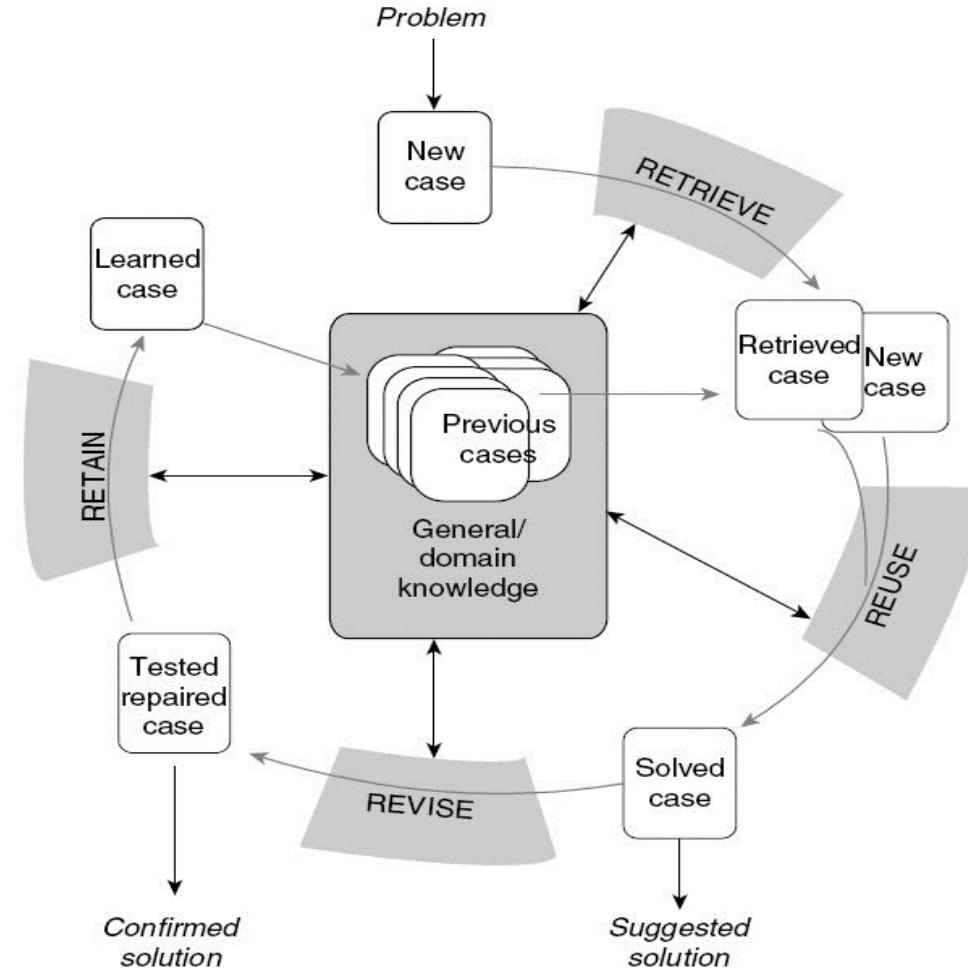
## 4 Re's

Retrieve

Reuse

Revise

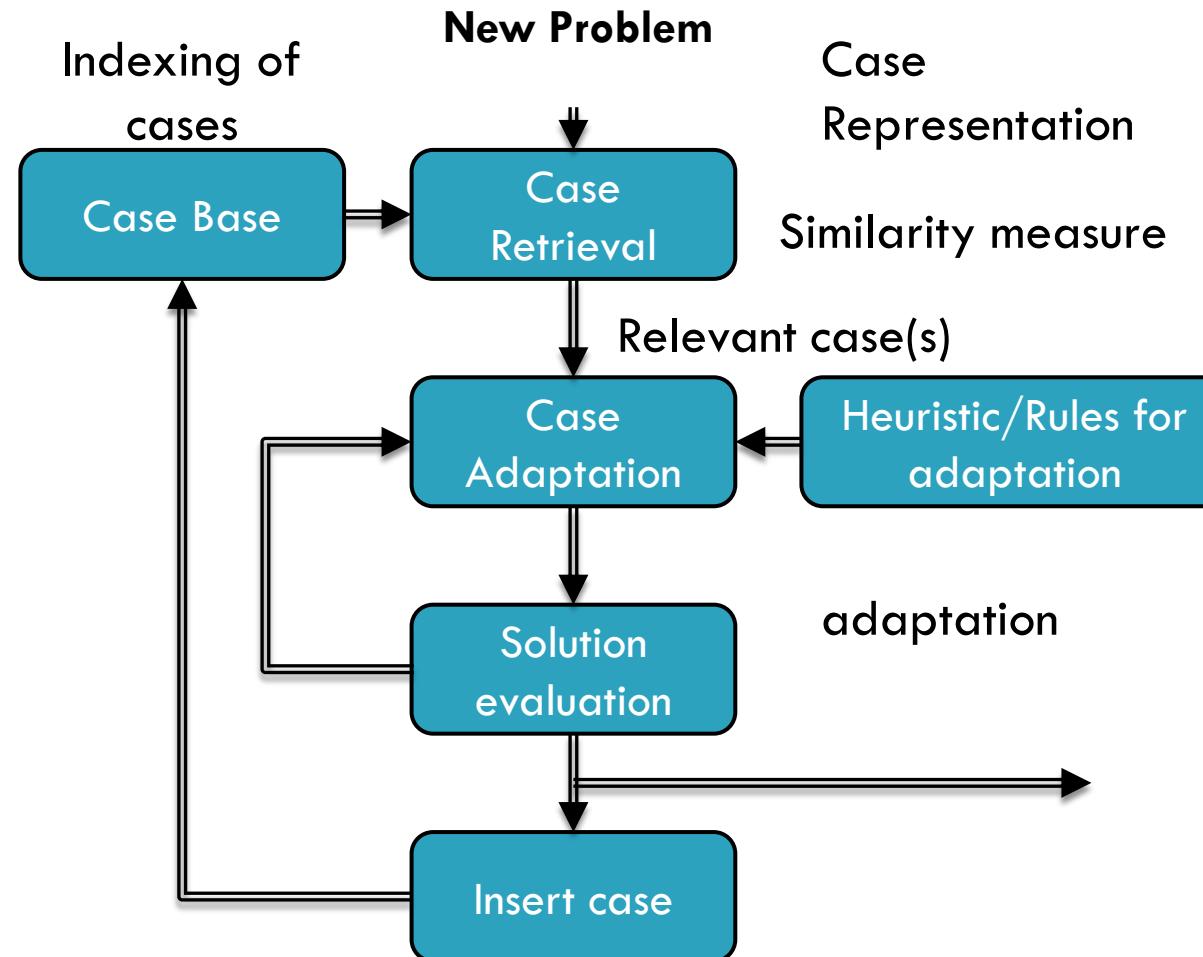
Retain



# Case Base Reasoning

58

Fig: Case Base Organization



# Case Base Reasoning

59

## 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

60

## 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

61

## 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

62

## 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

63

## 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

64

## 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

65

## 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

66

## CBR Applications

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

# References

67

- Rich and Knight
- Russel and Norvig

# UNIT 5

Machine Learning

# Contents

2

- Concepts of Learning (Based of Winston)
- Learning by analogy
- Inductive Learning
- Neural Network
- Genetic Algorithms
- Explanation based Learning
- Boltzmann Machines

# Concept of Learning

3

## Learning

- Acquiring new knowledge – Knowledge acquisition
- Modifying Old Knowledge, behavior, and skills- skills refinement
- May involve synthesizing different types of information
- Learning is also based on feedbacks

# Concept of Learning

4

## Types of Learning

- **Rote Learning**
  - ▣ Rote learning is a technique which focuses on memorization
  - ▣ It avoids understanding the inner complexities and inferences of the subject that is being learned and instead focuses on memorizing the material
  - ▣ The major practice involved in rote learning is repetition
- **Learning by examples**
  - ▣ Agent learns by seeing examples and classify the similar object to same class
- **Explanation based Learning**
  - ▣ Describes objects in brief
- **Learning by taking advice**
- **Learning by analogy**

# Concept of Learning

5

## Types of Learning

- Inductive Learning
  - Determine general pattern, rules and facts
  - Mainly example of experienced based learning
- Deductive Learning
  - Determine specific patterns, rules and facts
  - New rules are generated from old ones

# Concept of Learning

6

## Machine Learning

- Branch of AI that use algorithms to allow computer to evolve behaviours based on data collected from database or gathered through sensors
- Focuses on prediction, based on known properties learned from the training data
- The performance is usually evaluated with respect to the ability to reproduce known knowledge

# Concept of Learning

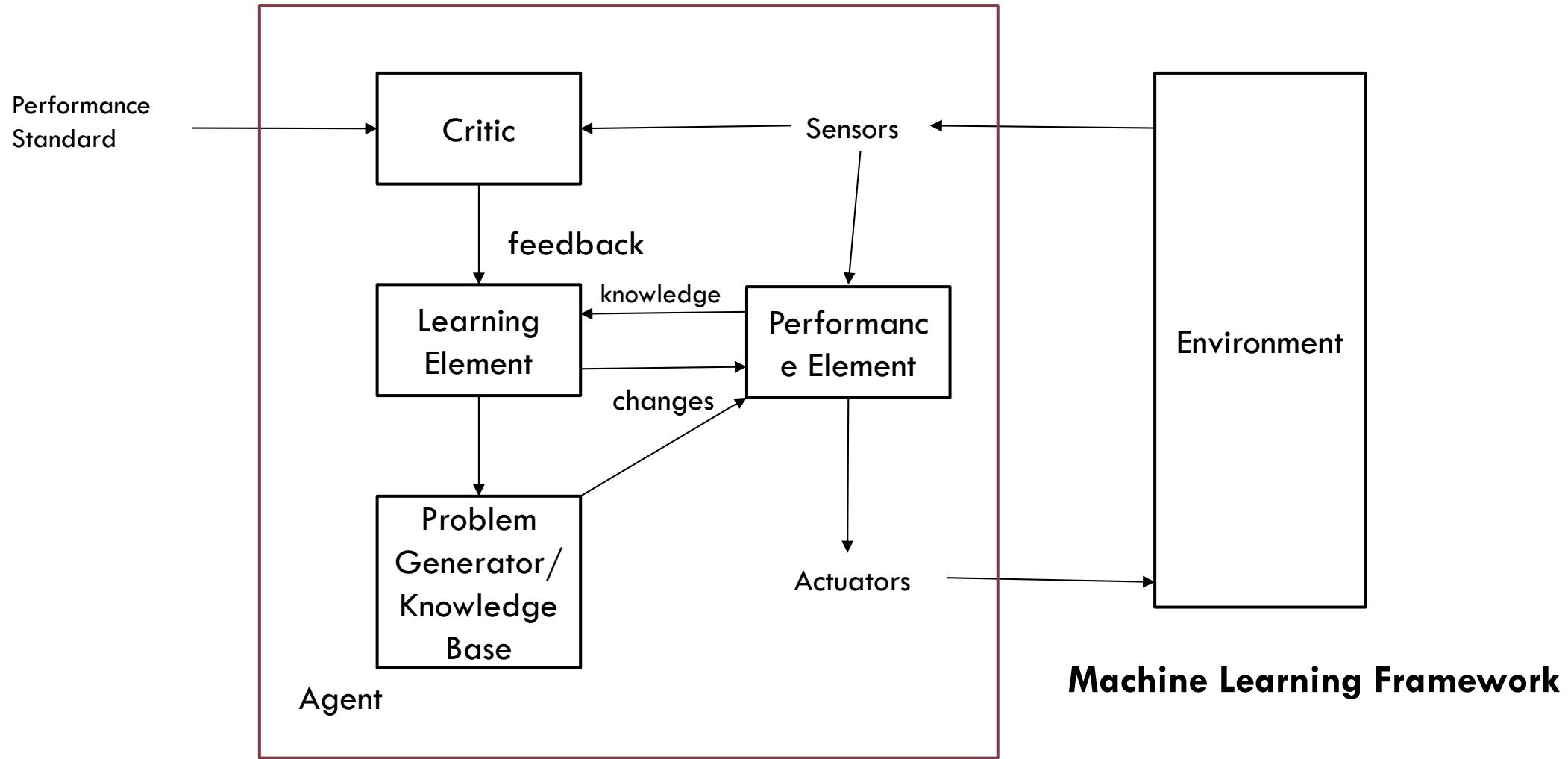
7

## Machine Learning

- Different cases of machine learning
  - ▣ Supervised learning: inputs and corresponding outputs are bind together
  - ▣ Unsupervised learning: only inputs are available for the learner
  - ▣ Reinforcement learning: learning based on reward or punishment

# Concept of Learning

8



# Concept of Learning

9

## Machine Learning Framework

- Environment
  - ▣ It refers to the nature and quality of the information given to the learning element
  - ▣ The nature of information depends on its level
    - High level: information is abstract and deals with broad class of problems
    - Low level: information is detailed and deals with single problem
  - ▣ Quality of information involves noise free, ordered and reliable information

# Concept of Learning

10

## Machine Learning Framework

- Learning Element
  - Acquire new knowledge through learning element
  - Learning may be of any type discussed above
- Problem Generator/ Knowledge Base
  - Stores the information about the problems and solutions are suggested
  - Knowledge Base should be
    - Expressive: Knowledge should be represented in easy and understandable way
    - Modifiable: Must be easy to update or add new data in the knowledge base
    - Extendibility: the knowledge base should have feature to change its structure that should be well defined

# Concept of Learning

11

## Machine Learning Framework

- Performance Element
  - ▣ This part analyses how complex the learning is and how learning is being performed
  - ▣ Complexity depends on the type of task to be performed
  - ▣ It must send feedback to the learning system as well so that evaluation of overall performance could be done
  - ▣ The learning element should have access to all internal actions of the performance elements
- Sensors and Actuators
  - ▣ Sensors collect information from environment
  - ▣ Actuators implement the suggested action

# Concept of Learning

12

## Winston's Learning or Semantic Nets Learning

- This program operated in simple blocks world domain
- Goal is to construct representations of the definitions of concepts in the block domain
- Eg. Learns the concepts of house, tents, and arches.
- Introduces near miss for each concept.
- Near miss is an object that is not an instance of the concept in question but that is very similar to such instances.

# Concept of Learning

13

## Winston's Learning or Semantic Nets Learning

- Begins with a structural description of one known instance of the concept
- Examine descriptions of other known instances of the concept
- Examine descriptions of near misses of the concept and restrict the definition to exclude this
- For example: refer to page number 459-461 of Artificial Intelligence, Rich and Knight

# Learning by Analogy

14

- Analogy is a powerful inference tool
- Our language and reasoning are laden with analogy
- Example:
  - Last month share market was a roller coaster.
  - Bill is like a fire engine
- So, AI must be able to grasp analogy for learning easily
- It is used in different learning strategies like learning by advice taking, learning in problem solving, etc.

# Learning by Analogy

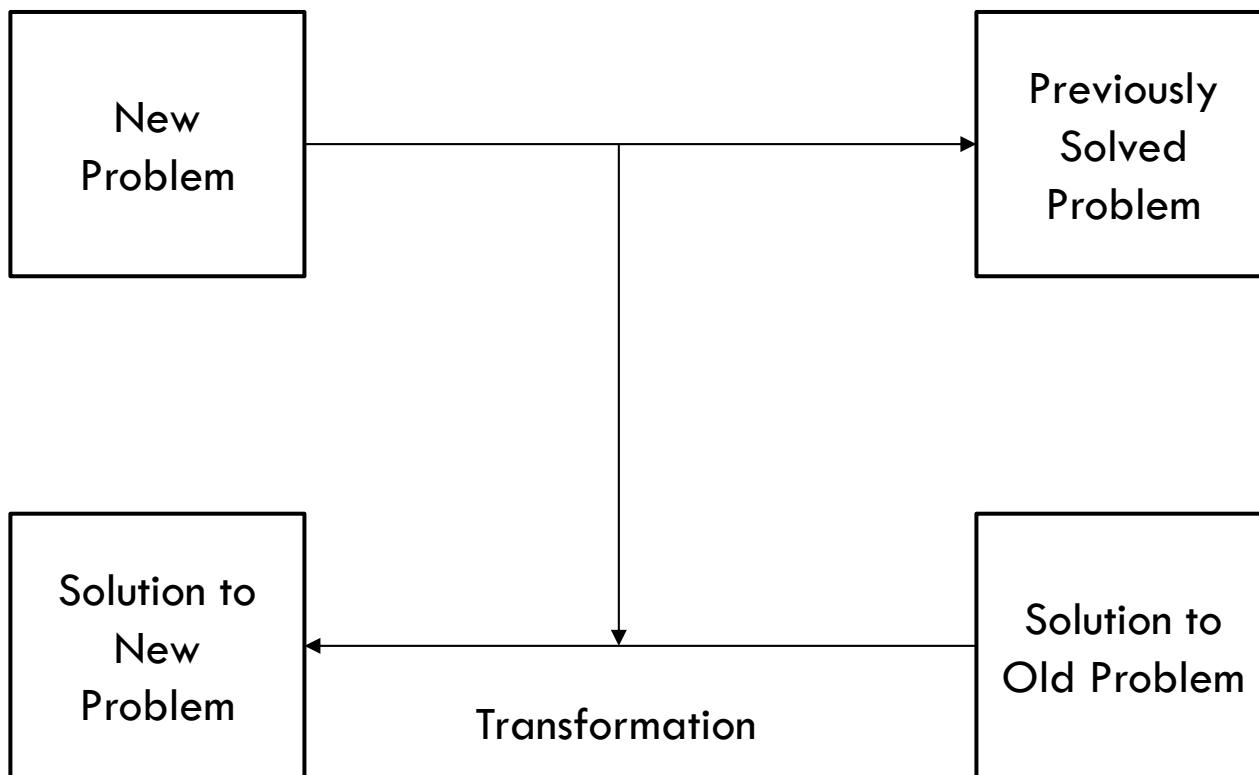
15

- Two methods of Analogy Problem solving are:
  - Transformational Analogy : focuses on final solution
  - Derivational Analogy: focuses on process of problem solving

# Learning by Analogy

16

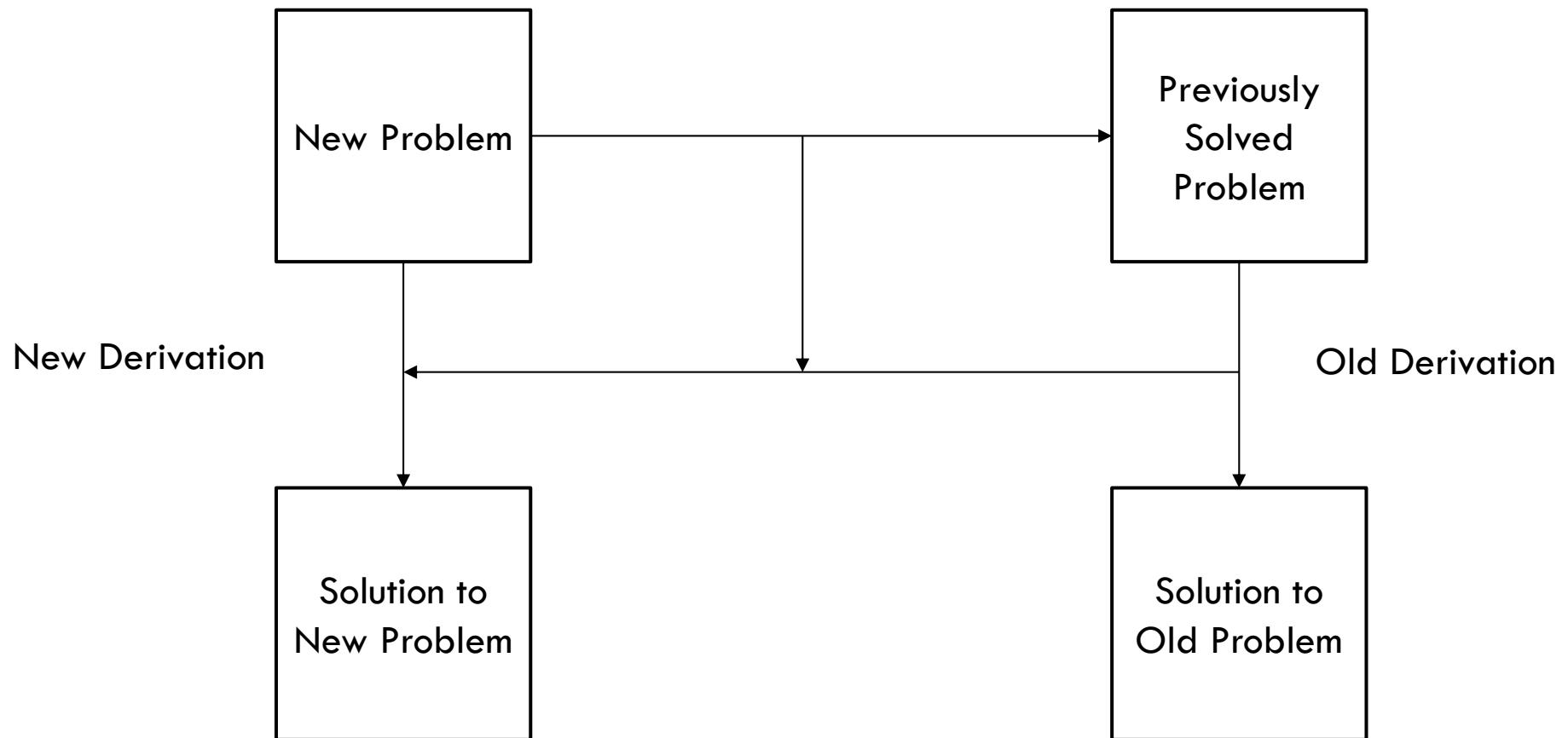
## Transformational Analogy



# Learning by Analogy

17

## Derivational Analogy



# Inductive Bias Learning

18

- Based on classification of problems prior to solving
- Based on induction of result
- Based on generic facts, rules and patterns
- Also called concept learning
- Approaches:
  - Winston's Learning Program
  - Version Space
  - Decision Tree

# Genetic Algorithm

- **Algorithm:** An algorithm is a sequence of instructions to solve a problem. Most of the algorithms are static.
- A **Genetic Algorithm(GA)** is adaptive (dynamic) a model of machine learning algorithm that derives its behavior from a metaphor of some of the mechanisms of evolution in nature.

# GA - Background

- On 1 July 1858, **Charles Darwin**, presented his **theory of evolution**. This day marks the beginning of a revolution in Biology.
- **Darwin's classical theory of evolution**, together with Weismann's **theory of natural selection** and Mandel's concept of **genetics**, now represent the **neo-Darwinism**
- **Neo=Darwinism** is based on process of reproduction, mutation, competition and selection.

# GA - Background

- Evolution can be seen as a process leading to the maintenance of a population's ability to survive and reproduce in a specific environment. This ability is called **evolutionary fitness**.
- Evolutionary fitness can also be viewed as a measure of the organism's ability to anticipate changes in its environment.
- The fitness, or the quantitative measure of the ability to predict environmental changes and respond adequately, can be considered as the quality that is optimised in natural life

# GA - Evolutionary Computation

- Evolutionary Computation stimulates evolution on a computer. The result of such simulations is a sense of optimisation algorithms
- Optimisation iteratively improves the quality of solutions until an optimal, or near-optimal, solution is found
- The evolutionary approach is based on computational models of natural selection and genetics. We call them **evolutionary computation**, an umbrella term that combines **genetic algorithms, evolution strategies and genetic programming**

# GA - Simulation of Natural Evolution

- All methods of evolutionary computation simulate natural evolution by creating a population of individuals, evaluating their fitness, generating a new population through genetic operations, and repeating this process a number of times.
- We focus on **Genetic Algorithm** as most of the other algorithms can be viewed as variations of genetic algorithms.

# Genetic Algorithm

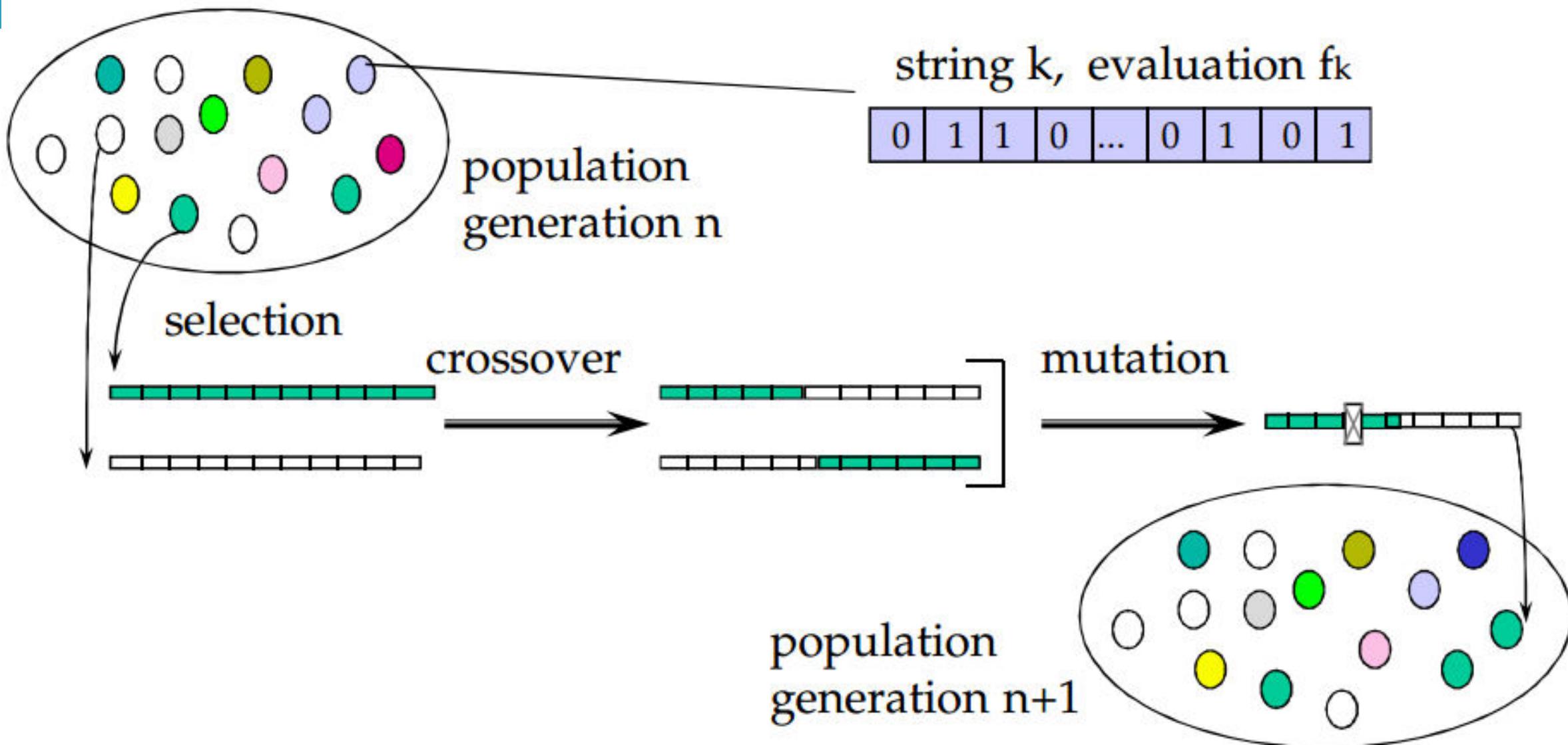
- In early 1970s John Holland introduced the concept of genetic algorithm
- His aim was to make computers do what nature does. Holland was concerned with algorithms that manipulate strings of binary digits
- Each artificial “chromosomes” consists of a number of “genes”, and each gene is represented by 0 or 1

1	0	1	1	0	1	0	0	0	0	0	1	0	1	0	1
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

# Genetic Algorithm

- Two mechanisms link a GA to the problem it is solving :  
**Encoding and Evaluation**
- The GA uses a measure of fitness of individual chromosomes to carry out reproduction. As reproduction takes place, the crossover operator exchanges part of two single chromosomes and the mutation operator changes the gene value in some randomly chosen location of the chromosome

# Basic Genetic Algorithm



# Genetic Algorithm

## GA Operators and Parameters

- **Fitness function:** The fitness function is defined over the genetic representation and measures the *quality* of the represented solution.
- **Selection Operator:** Selects parents for reproduction based on relative fitness of candidates in the population
  - Roulette Wheel Selection
  - Ranking Selection

# Genetic Algorithm

## □ Crossover Operator:

- Exchanges part of chromosome between two parent chromosomes with some crossover rate(probability), typically 0.4 – 0.8
- The main operator to provide exploitation in search building up good genes in chromosome
  - **One Point Crossover:** randomly chooses a crossover point where two parent chromosomes “break” and then exchanges the chromosome parts after that point. As a result, two new offspring is created

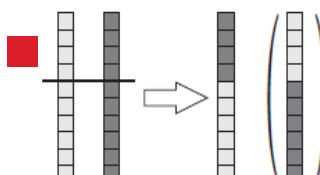


Fig. .a: Single-point Crossover (SPX).

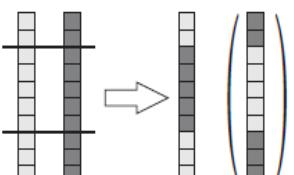


Fig. .b: Two-point Crossover (TPX).

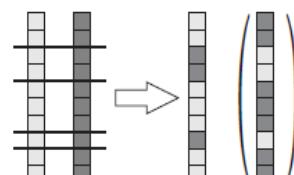


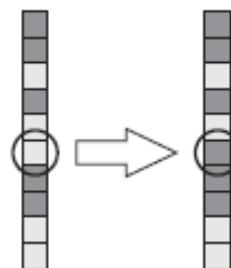
Fig. .c: Multi-point Crossover (MPX).

→ crossover points in two the chromosome parts spring are created.

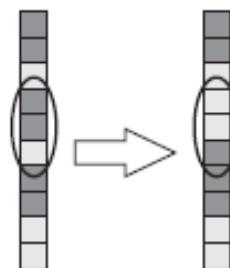
# Genetic Algorithm

## □ Mutation Operator:

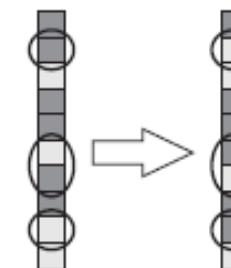
- Changes a randomly selected gene in the chromosome
- mimics random changes in genetic code
- Background operator to provide exploration in search to avoid being trapped on a local optimum
- Mutation probability is quiet small in nature and is kept low for GAs. typically in the range between [ 0.001 – 0.01 ] or



a: Single-gene mutation.



b: Multi-gene mutation



c: Multi-gene mutation

# Genetic Algorithm

- **Elitism Approach :** Saves the best individual in next generation
- **Basic GA Parameters:**
  - Population size
  - Crossover rate (Probability)
  - Mutation Rate (Probability)
  - Number of Generation ( a Stopping Criterion)

## Steps in Genetic Algorithm

1. Represent the problem variable as a chromosome of a fixed length, choose the size of a chromosome population  $N$ , the crossover probability  $pc$  and the mutation probability  $pm$ .
2. Define a fitness function to measure the fitness of an individual chromosome in the problem domain.
3. Randomly generate an initial population of chromosomes of size  $N$ :  $x_1, x_2, \dots, x_N$
4. Calculate the fitness of each individual chromosome:  $f(x_1), f(x_2), \dots, f(x_N)$
5. Select a pair of chromosomes for mating from the current population based on their fitness.
6. Create a pair of offspring chromosomes by applying the genetic operators – **crossover** and **mutation**.
7. Place the created offspring chromosomes in the new population.
8. Repeat Step 5 until the size of the new chromosome population becomes equal to the size of the initial population,  $N$ .
9. Replace the initial (parent) chromosome population with the new (offspring) population.
10. Go to Step 4, and repeat the process until the termination criterion is satisfied.

# Genetic Algorithm : Case Study

## Example of Selection

Evolutionary Algorithms is to maximize the function  $f(x) = x^2$  with  $x$  in the integer interval  $[0, 31]$ , i.e.,  $x = 0, 1, \dots, 30, 31$ .

1. The first step is encoding of chromosomes; use binary representation for integers; 5-bits are used to represent integers up to 31.
2. Assume that the population size is 4.
3. Generate initial population at random. They are chromosomes or genotypes; e.g., 01101, 11000, 01000, 10011.
4. Calculate fitness value for each individual.
  - (a) Decode the individual into an integer (called phenotypes),  
 $01101 \rightarrow 13; 11000 \rightarrow 24; 01000 \rightarrow 8; 10011 \rightarrow 19;$
  - (b) Evaluate the fitness according to  $f(x) = x^2$ ,  
 $13 \rightarrow 169; 24 \rightarrow 576; 8 \rightarrow 64; 19 \rightarrow 361.$
5. Select parents (two individuals) for crossover based on their fitness in  $p_i$ . Out of many methods for selecting the best chromosomes, if roulette-wheel selection is used, then the probability of the  $i^{th}$  string in the population is  $p_i = F_i / (\sum_{j=1}^n F_j)$ , where

$F_i$  is fitness for the string  $i$  in the population, expressed as  $f(x)$

$p_i$  is probability of the string  $i$  being selected,

$n$  is no of individuals in the population, is population size,  $n=4$

$n * p_i$  is expected count

String No	Initial Population	X value	Fitness $f(x) = x^2$	$p_i$	Expected count $N * Prob i$
1	0 1 1 0 1	13	169	0.14	0.58
2	1 1 0 0 0	24	576	0.49	1.97
3	0 1 0 0 0	8	64	0.06	0.22
4	1 0 0 1 1	19	361	0.31	1.23
<b>Sum</b>			<b>1170</b>	<b>1.00</b>	<b>4.00</b>
<b>Average</b>			293	0.25	1.00
<b>Max</b>			576	0.49	1.97

## Genetic Algorithm : Case Study

The string no 2 has maximum chance of selection.

6. Produce a new generation of solutions by picking from the existing pool of solutions with a preference for solutions which are better suited than others:

We divide the range into four bins, sized according to the relative fitness of the solutions which they represent.

Strings	Prob i	Associated Bin
0 1 1 0 1	0.14	0.0 ... 0.14
1 1 0 0 0	0.49	0.14 ... 0.63
0 1 0 0 0	0.06	0.63 ... 0.69
1 0 0 1 1	0.31	0.69 ... 1.00

By generating **4** uniform **(0, 1)** random values and seeing which bin they fall into we pick the four strings that will form the basis for the next generation.

Random No	Falls into bin	Chosen string
0.08	0.0 ... 0.14	0 1 1 0 1
0.24	0.14 ... 0.63	1 1 0 0 0
0.52	0.14 ... 0.63	1 1 0 0 0
0.87	0.69 ... 1.00	1 0 0 1 1

## Genetic Algorithm : Case Study

7. Randomly pair the members of the new generation  
Random number generator decides for us to mate the first two strings together and the second two strings together.
8. Within each pair swap parts of the members solutions to create offspring which are a mixture of the parents :

For the first pair of strings:      **0 1 1 0 1 , 1 1 0 0 0**

- We randomly select the crossover point to be after the fourth digit.

Crossing these two strings at that point yields:

**0 1 1 0 1**  $\Rightarrow$  **0 1 1 0 |1**  $\Rightarrow$  **0 1 1 0 0**

**1 1 0 0 0**  $\Rightarrow$  **1 1 0 0 |0**  $\Rightarrow$  **1 1 0 0 1**

For the second pair of strings:      **1 1 0 0 0 , 1 0 0 1 1**

- We randomly select the crossover point to be after the second digit.

Crossing these two strings at that point yields:

**1 1 0 0 0**  $\Rightarrow$  **1 1 |0 0 0**  $\Rightarrow$  **1 1 0 1 1**

**1 0 0 1 1**  $\Rightarrow$  **1 0 |0 1 1**  $\Rightarrow$  **1 0 0 0 0**

# Genetic Algorithm : Case Study

9. Randomly mutate a very small fraction of genes in the population :  
With a typical mutation probability of per bit it happens that none of the bits in our population are mutated.
  
10. Go back and re-evaluate fitness of the population (new generation) :  
This would be the first step in generating a new generation of solutions. However it is also useful in showing the way that a single iteration of the genetic algorithm has improved this sample.

<i>String No</i>	<i>Initial Population (chromosome)</i>	<i>X value (Pheno types)</i>	<i>Fitness <math>f(x) = x^2</math></i>	<i>Prob i (fraction of total)</i>	<i>Expected count</i>
1	0 1 1 0 0	12	144	0.082	0.328
2	1 1 0 0 1	25	625	0.356	1.424
3	1 1 0 1 1	27	729	0.415	1.660
4	1 0 0 0 0	16	256	0.145	0.580
<b>Total (sum)</b>			<b>1754</b>	<b>1.000</b>	<b>4.000</b>
<b>Average</b>			<b>439</b>	<b>0.250</b>	<b>1.000</b>
<b>Max</b>			<b>729</b>	<b>0.415</b>	<b>1.660</b>

Observe that :

1. Initial populations : At start step 5 were

0 1 1 0 1 , 1 1 0 0 0 , 0 1 0 0 0 , 1 0 0 1 1

After one cycle, new populations, at step 10 to act as initial population

0 1 1 0 0 , 1 1 0 0 1 , 1 1 0 1 1 , 1 0 0 0 0

2. The total fitness has gone from **1170** to **1754** in a single generation.
3. The algorithm has already come up with the string 11011 (i.e  $x = 27$ ) as a possible solution.

# Explanation Based Learning

36

- *Explanation-based learning (EBL)* uses a domain theory to construct an explanation of the training example, usually a proof that the example logically follows from the theory
- Using this proof the system filters the noise, selects only the relevant to the proof aspects of the domain theory, and organizes the training data into a systematic structure
- This makes the system more efficient in later attempts to deal with the same or similar examples.

# Explanation Based Learning

37

## ***Basic Concept of EBL***

- *Target concept.* The task of the learning system is to find an effective definition of this concept. Depending on the specific application the target concept could be a classification, theorem to be proven, a plan for achieving goal, or heuristic to make a problem solver more efficient.
- *Training example.* This is an instance of the target concept
- *Domain theory.* Usually this is a set of rules and facts representing domain knowledge. They are used to explain how the training example is an instance of the target concept.
- *Operationality criteria.* Some means to specify the form of the concept definition.

# Boltzmann Machine

38

- It's a network of symmetrically connected, neuron like units that make stochastic decision about whether to be on or off
- Has simple learning algorithm that allows it to discover interesting features that represent complex regularities in training data
- Very slow learning
- For searching, Boltzmann machine has fixed weights on the connection but for learning weights use small update

# Boltzmann Machine

39

- Output  $y$  is given by,

$$y = b_i + \sum x_j w_{ij}$$

- Where  $x_i=1$  for on state and 0 for off
- Probability of being on is

$$\text{Prob}(x_j = 1) = \frac{1}{1 + e^{-y}}$$

Thank You

# APPLICATION OF AI

Unit 6

# Contents

2

- Neural Networks
  - Network structure
  - Perceptron
  - Adaline networks
  - Madaline Networks
  - Multilayer Perceptron
  - Radial Basis Function
  - Kohonen Network
  - Elastic Net Model
  - Back Propagation

# Contents

3

- Expert System
  - Architecture of an Expert System
  - Knowledge acquisition
  - Induction
  - Knowledge Representation
  - Declarative Knowledge
  - Procedural Knowledge
  - Knowledge elicitation technique
  - Intelligent Editing Programs
  - Development of an Expert System
- Natural Language Processing
  - Levels of Analysis : Phonetic, Syntactic, Semantic, Pragmatic
  - Machine Vision, Bottom-up approach, Edge Detection, line detection, Need for top-down, Hypothesis-driven approaches

# Neural Networks

4

- ‘The computer hasn’t proved anything yet,’ angry Garry Kasparov, the world chess champion, said after his defeat in New York in May 1997. ‘If we were playing a real competitive match, I would tear down Deep Blue into pieces.’

# Network Structure

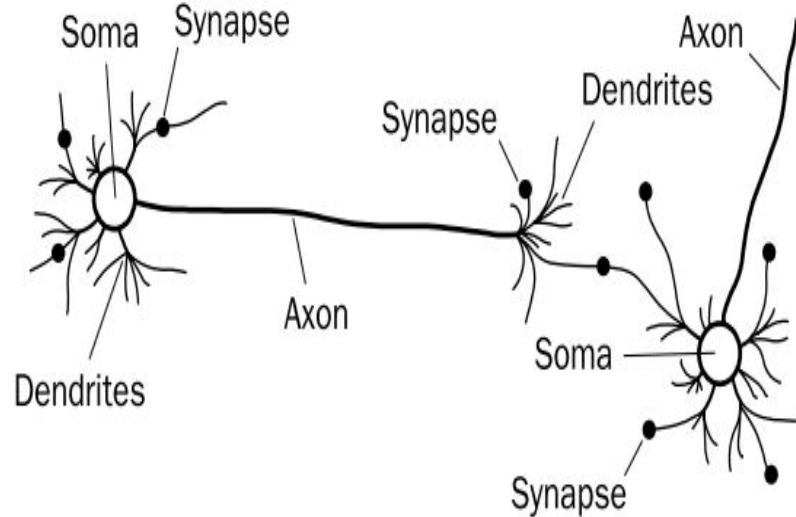
5

- A neural network can be defined as a model of reasoning based on the human brain. The brain consists of a densely interconnected set of nerve cells, or basic information-processing units, called **neurons**
- The human brain incorporates nearly 10 billion neurons and 60 trillion connections, **synapses**, between them. By using multiple neurons simultaneously, the brain can perform its functions much faster than the fastest computers in existence today.

# Network Structure

6

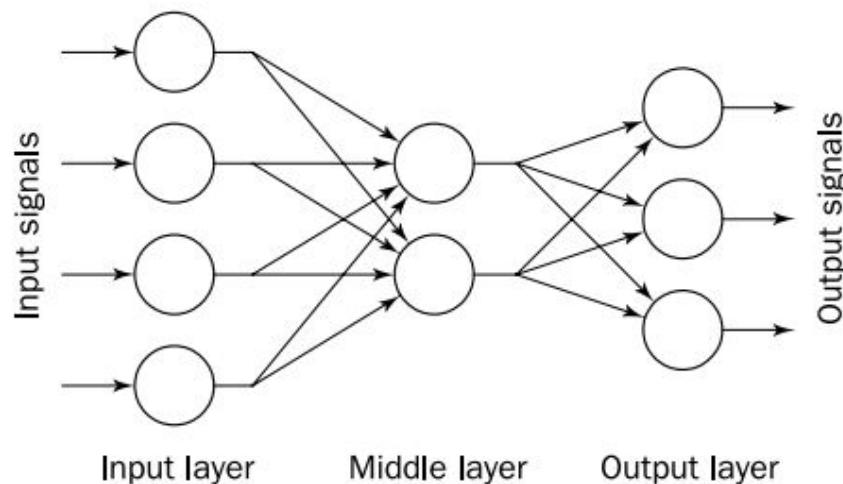
- Each neuron has a very simple structure, but an army of such elements constitutes a tremendous processing power
- **Neuron:** fundamental functional unit of all nervous system tissue
- **Soma:** cell body, contain nucleus
- **Dendrites:** a number of fibres, input
- **Axon:** single long fibre with many branches, output
- **Synapse:** junction of dendrites and axon, each neuron form synapse with 10 to 100000 other neurons



# Network Structure

7

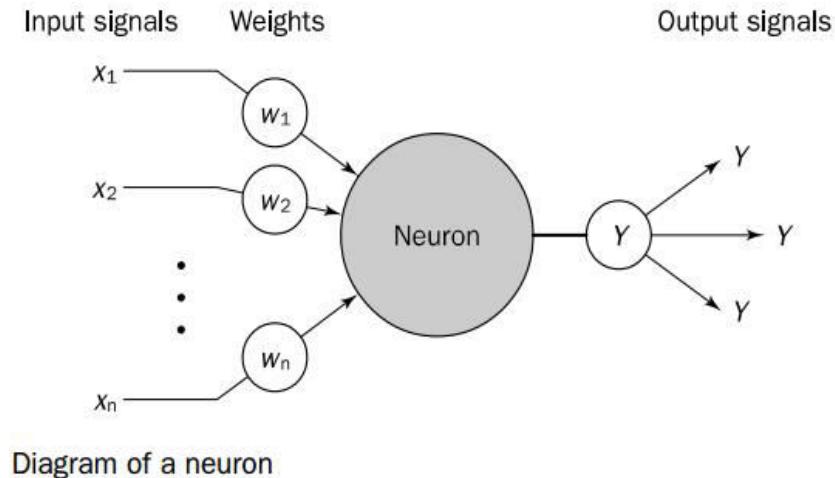
- Consists of a number of very simple and highly interconnected processors called neurons
- The neurons are connected by weighted links passing signals from one neuron to another.
- The output signal is transmitted through the neuron's outgoing connection. The outgoing connection splits into a number of branches that transmit the same signal. The outgoing branches terminate at the incoming connections of other neurons in the network



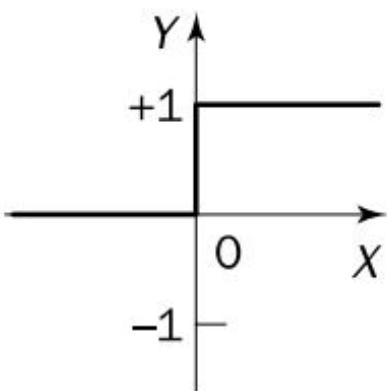
# The Neuron as a simple computing element: Diagram of a neuron

8

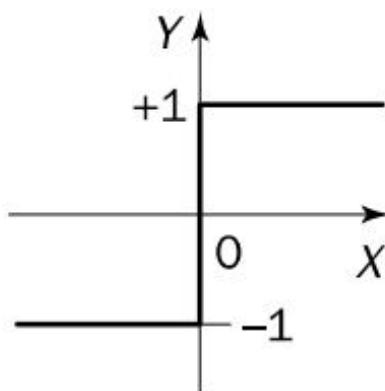
- ❑ The neuron computes the weighted sum of the input signals and compares the result with a **threshold value**,  $\theta$ . If the net input is less than the threshold, the neuron output is  $-1$ . But if the net input is greater than or equal to the threshold, the neuron becomes activated and its output attains a value  $+1$ .
- ❑ The neuron uses the following transfer or **activation function**
- ❑  $X = \sum_{i=1}^n x_i w_i \quad Y = \begin{cases} +1 & \text{if } X \geq \theta \\ -1 & \text{if } X \leq \theta \end{cases}$
- ❑ This type of activation function is called a **sign function**



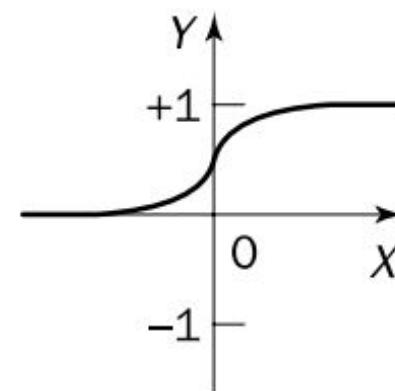
Step function



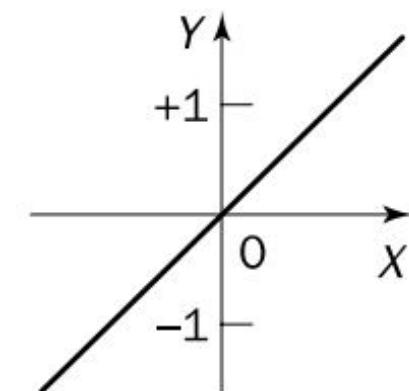
Sign function



Sigmoid function



Linear function



$$Y^{step} = \begin{cases} 1, & \text{if } X \geq 0 \\ 0, & \text{if } X < 0 \end{cases}$$

$$Y^{sign} = \begin{cases} +1, & \text{if } X \geq 0 \\ -1, & \text{if } X < 0 \end{cases}$$

$$Y^{sigmoid} = \frac{1}{1 + e^{-X}}$$

$$Y^{linear} = X$$

9

## Network Structure

Fig: Activation Functions of Neuron

# Perceptron

10

- In 1958, Frank Rosenblatt introduced a training algorithm that provided the first procedure for training a simple ANN : a **perceptron**
- The perceptron is the simplest form of a neural network. It consists of a single neuron with *adjustable* synaptic weights and a **hard limiter**
- The operation of Rosenblatt's perceptron is based on the McCulloch and Pitts neuron model. The model consists of a **linear combiner** followed by a hard limiter
- The weighted sum of the inputs is applied to the hard limiter, which produces an output equal to +1 if its input is positive and -1 if its input is negative

## Inputs

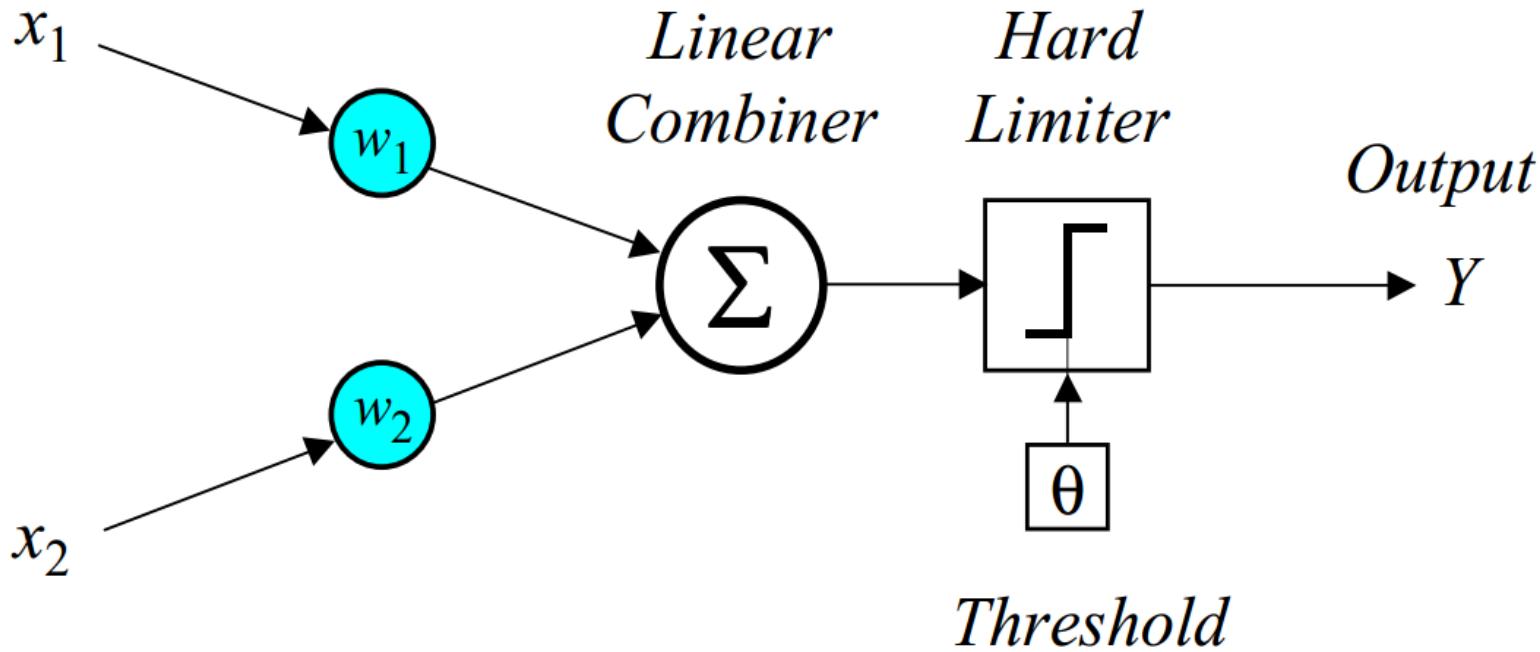


Fig: Single Layer two input Perceptron

# Perceptron

12

- The aim of the perceptron is to classify inputs,  $x_1, x_2, x_3, x_4, \dots, x_n$ . Into one of two classes, say  $A_1, A_2$ .
- In the case of an elementary perceptron, the n-dimensional space is divided into a *hyperplane* into two decision regions. The hyperplane is defined by ***linearly separable function***

$$\sum_{i=1}^n (x_i w_i - \theta) = 0$$

# Perceptron

13

## How does the perceptron learn its classification tasks?

- ❑ This is done by making small adjustment in the weights to reduce the difference between the actual and desired outputs of the perceptron. The initial weights are randomly assigned, usually in the range [-0.5, +0.5], and then updated to obtain the output consistent with the training examples.
- ❑ If at iteration  $p$ , the actual output is ,  $Y_{(p)}$  and the desired output is ,  $Y_{d(p)}$ , then the error is given by :

$$e_p = Y_{d(p)} - Y_{(p)} , \text{ where } p = 1, 2, 3, \dots$$

Iteration  $p$  here refers to the  $p^{th}$  training example presented to the perceptron

- ❑ If the error,  $e_p$  is positive, we need to increase perceptron output  $Y_{(p)}$ , but if it is negative, we need to decrease  $Y_{(p)}$

# Perceptron

14

## Perceptron Learning Rule

$$w_i(p+1) = w_i(p) + \alpha \times x_i(p) \times e(p)$$

where  $p = 1, 2, 3, \dots$

$\alpha$  is the **learning rate**, a positive constant less than unity

The perceptron learning rule was first proposed by Rosenblatt in 1960. Using this rule we can derive perceptron training algorithm for classification task

# Perceptron

15

## Perceptron Training Algorithm

### □ Step 1 : Initialization

Set initial weights  $w_1, w_2, \dots, w_n$  and threshold  $\theta$  to random numbers in the range [-0.5, +0.5].

If error  $e_p$  is positive, we need to increase perceptron output  $Y_p$ , but if it is negative, we need to decrease  $Y_p$

# Perceptron

16

## Perceptron Training Algorithm

### □ Step 2 : Activation

Activate the perceptron by applying inputs  $x_1(p), x_2(p), \dots, x_n(p)$  and desired output  $Y_d(p)$

Calculate the actual output at iteration  $p = 1$

$$Y(p) = \text{step} \left[ \sum_{i=1}^n x_i(p)w_i(p) - \theta \right]$$

Where  $n$  is the number of the perceptron inputs, and  $\text{step}$  is activation function

# Perceptron

17

## Perceptron Training Algorithm

- **Step 3: Weight Training**
- Update the weights of the perceptron  
 $w_i(p + 1) = w_i(p) + \Delta w_i(p)$
- Where  $\Delta w_i(p)$  is the weight correction at iteration  $p$ . The weight correction is computed by the delta rule:  
$$\Delta w_i(p) = \alpha \times x_i(p) \times e(p)$$
- **Step 4: Iteration**
- Increase iteration  $p$  by 1, go back to Step 2 and repeat the process until convergence

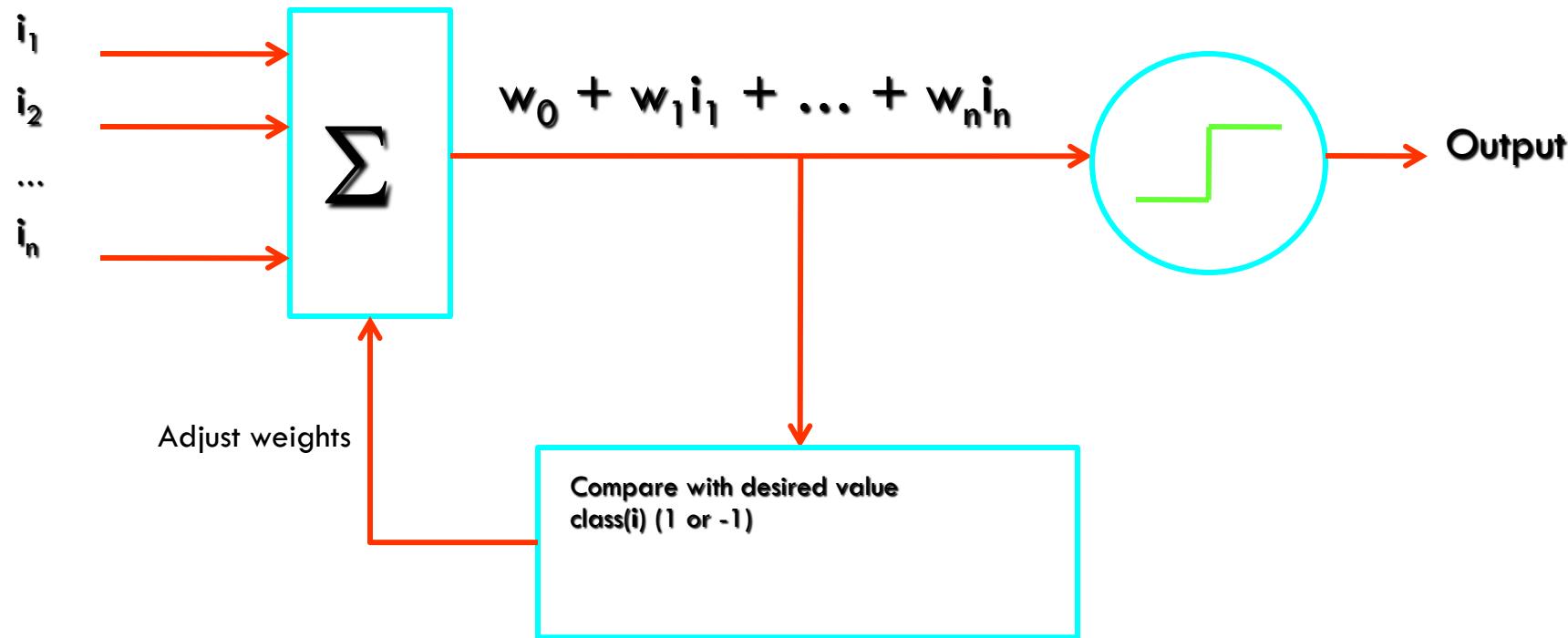
# Adaline

18

- Stands for **Adaptive Linear Element**
- It is a simple perceptron-like system that accomplishes classification by modifying weights in such a way as to diminish the Mean Square Error at every iteration. This can be accomplished using gradient Adaptive Linear Element [Adaline]
- Used in Neural network for
  - Adaptive filtering
  - Pattern Recognition

# Adaline

19



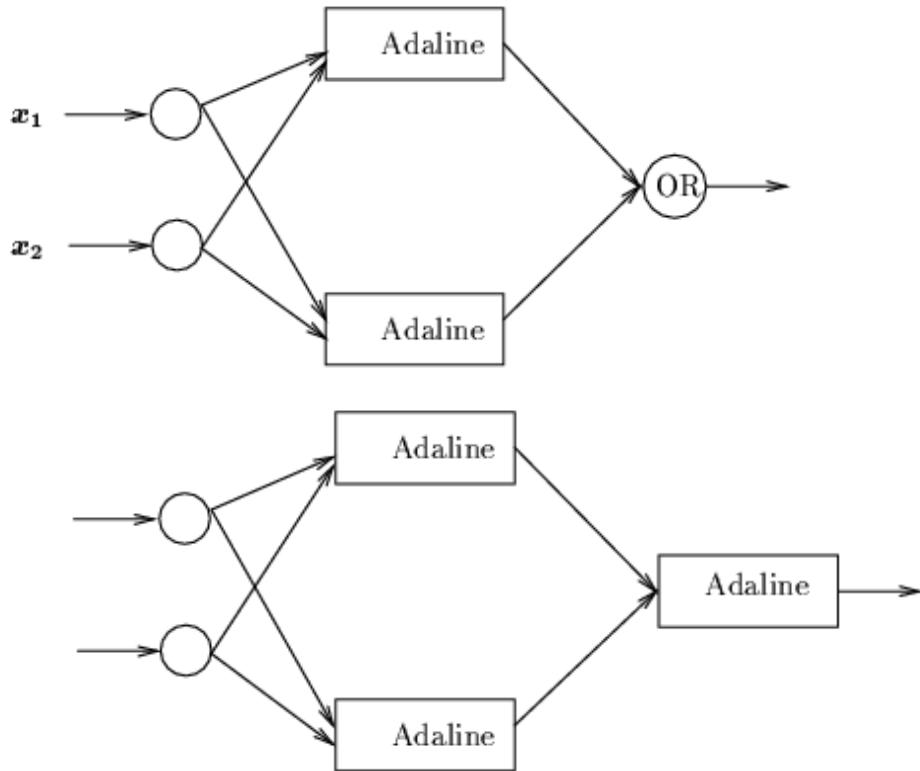
# Madaline

20

- **Architectures**
  - Hidden layers of adaline nodes
  - Output nodes differ
- **Learning**
  - Error driven, but not by gradient descent
  - Minimum disturbance: smaller change of weights is preferred, provided it can reduce the error
- **Three Madaline models**
  - Different node functions
  - Different learning rules (MR I, II, and III)
  - MR I and II developed in 60's, MR III much later (88)

# Madaline

21



**MRI net:**

Output nodes with logic function

**MRII net:**

Output nodes are adalines

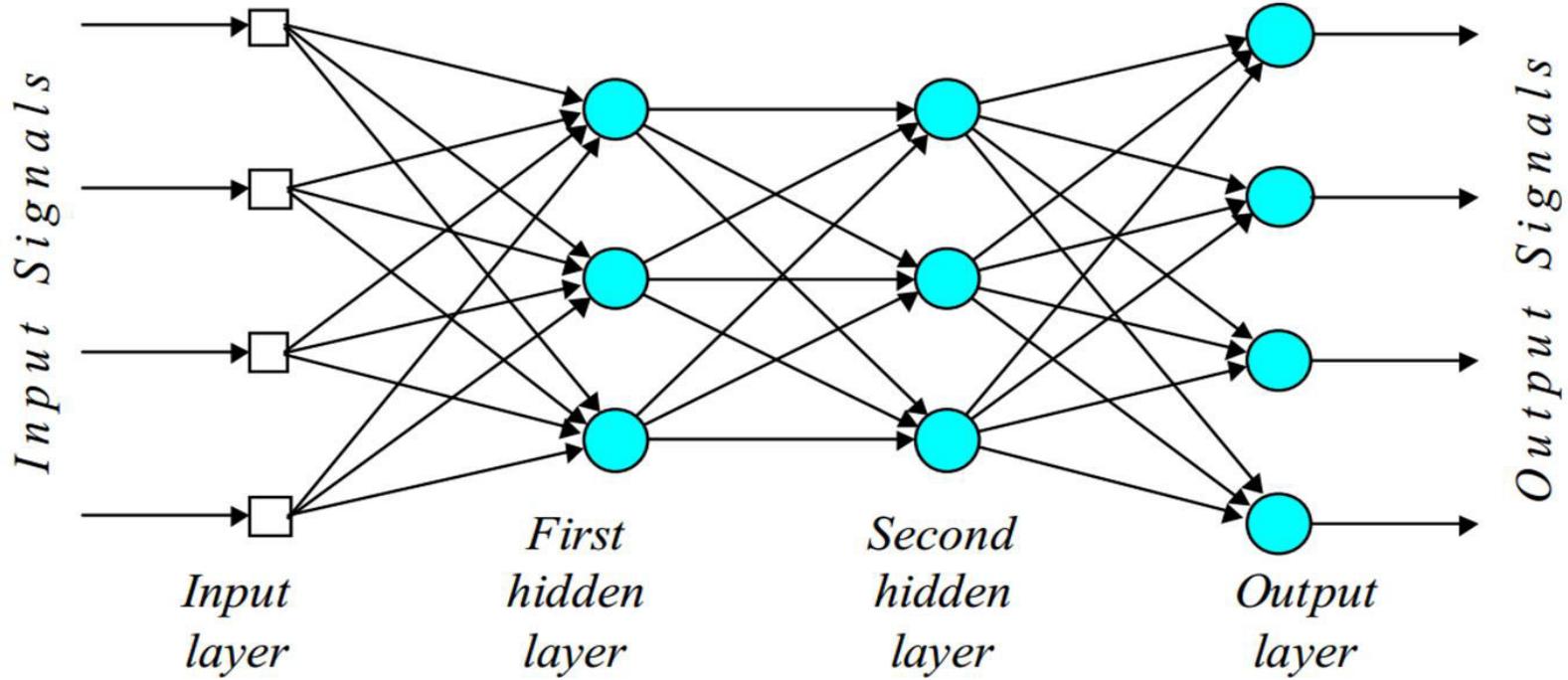
**MRIII net:**

Same as MRII, except the nodes with sigmoid function

# Multilayer Perceptron

22

- ❑ A multi layer perceptron is a feed forward neural network with one or more hidden layers
- ❑ The network consists of :
  - ❑ Input Layer
  - ❑ Hidden Layer
  - ❑ Output Layer
- ❑ The input signal is propagated in a forward direction in a layer-by-layer basis



## Multilayer Perceptron

Fig: Multilayer Perceptron with two hidden layers

# Multilayer Perceptron

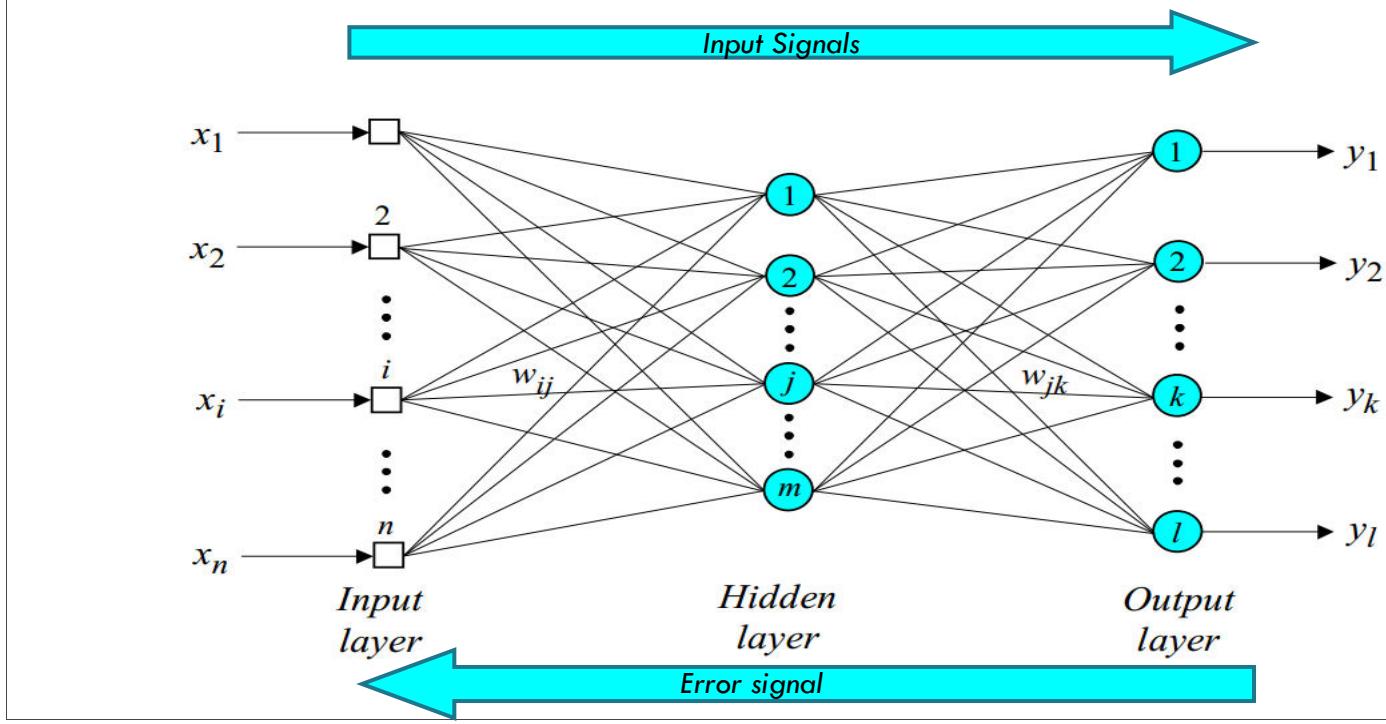
24

- ❑ The hidden layer “hides” its desired output. Neurons in the hidden layer can not be observed through the input/output behavior of the network. There is no obvious way to know what the desired output of the hidden layer should be.
- ❑ Commercial ANNs incorporate three and sometimes four layers, including one or two hidden layers. Each layer can contain from 10 to 1000 neurons. Experimental neural networks may have five or six layers, including three or four hidden layers, and utilize millions of neurons.

# Back Propagation

25

- ❑ Learning in a multilayer network proceeds the same way as for a perceptron
- ❑ A training set of input patterns is presented to the network
- ❑ The network computes its output pattern, and if there is an error –or other word difference between actual and desired output pattern – the weight are adjusted to reduce the error
- ❑ In a back-propagation neural network, the learning algorithm has two phases
- ❑ First, a training input pattern is presented to the network input layer. The network propagates the input pattern from layer to layer until the output pattern is generated by the out layer
- ❑ If this pattern is different from the desired output, an error is calculated and then propagated backwards through the network from the output layer to the input layer. The weights are modified as the error is propagated



# Back Propagation Training Algorithm

27

- **Step 1: Initialization**
- Set all the weights and threshold levels of the network to random numbers uniformly distributed inside a small range:
- $\left( -\frac{2.4}{F_i}, +\frac{2.4}{F_i} \right)$
- where  $F_i$  is the total number of inputs of neuron  $i$  in the network.  
The weight initialization is done on a neuron-by-neuron basis.

# Back Propagation Training Algorithm

- **Step 2: Activation**
- Activate the back-propagation neural network by applying inputs  $x_1(p), x_2(p), \dots, x_n(p)$  and desired outputs  $y_{d,1}(p), y_{d,2}(p), \dots, y_{d,n}(p)$
- A) Calculate the actual output of the neurons in the hidden layers:

$$y_j(p) = \text{sigmoid} \left[ \sum_{i=1}^n x_i(p) \cdot w_{ij}(p) - \theta_j \right]$$

- Where  $n$  is the number of inputs of neuron  $j$  in the hidden layer, and *sigmoid* is the *sigmoid* activation function

# Back Propagation Training Algorithm

29

- **Step 2: Activation(contd...)**
- b) Calculate the actual outputs of the neurons in the output layer:

$$y_k(p) = \text{sigmoid} \left[ \sum_{j=1}^m x_{jk}(p) \cdot w_{jk}(p) - \theta_k \right]$$

- Where  $m$  is the number of inputs of neuron  $k$  in the output layer.

# Back Propagation Training Algorithm

30

- **Step 3: weight Training**
- Update the weights in the back-propagation network propagating backward the errors associated with output neurons.
- (a) Calculate the error gradient for the neurons in the output layer:

$$\delta_k(p) = y_k(p) \cdot [1 - y_k(p)] \cdot e_k(p)$$

where  $e_k(p) = y_{d,k}(p) - y_k(p)$

Calculate the weight corrections:

$$\Delta w_{jk}(p) = \alpha \cdot y_j(p) \cdot \delta_k(p)$$

Update the weights at the output neurons:

$$w_{jk}(p+1) = w_{jk}(p) + \Delta w_{jk}(p)$$

# Back Propagation Training Algorithm

31

- **Step 3: weight Training(contd...)**
- (b) Calculate the error gradient for the neurons in the hidden layer:

$$\delta_j(p) = y_j(p) \cdot [1 - y_j(p)] \cdot \sum_{k=1}^l \delta_k(p) w_{jk}(p)$$

Calculate the weight corrections:

$$\Delta w_{ij}(p) = \alpha \cdot x_i(p) \cdot \delta_j(p)$$

Update the weights at the hidden neurons:

$$w_{ij}(p+1) = w_{ij}(p) + \Delta w_{ij}(p)$$

# Back Propagation Training Algorithm

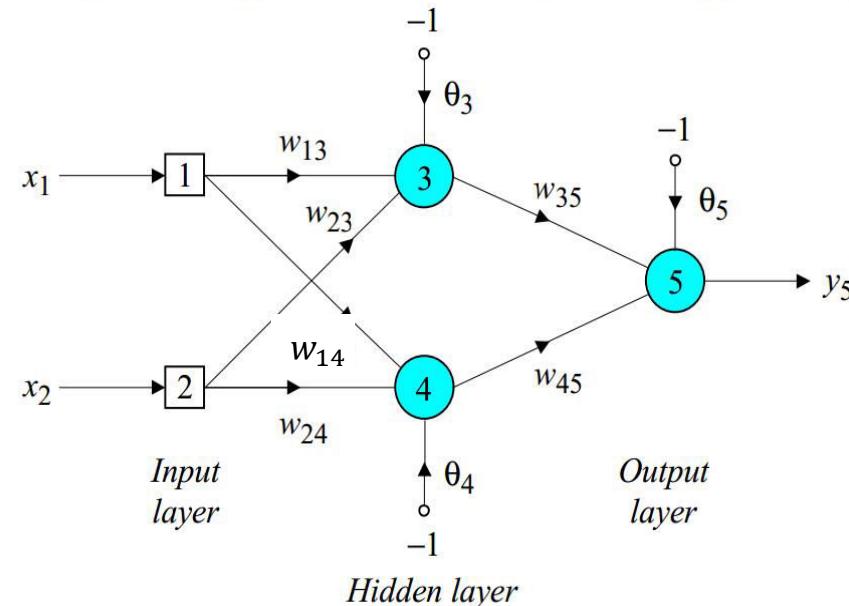
32

- **Step 3: Iteration**
- Increase iteration  $p$  by one, go back to Step 2 and repeat the process until the selected error criterion is satisfied.
- As an example, we may consider the three layer back-propagation network. Suppose that the network is required to perform logical operation Exclusive-OR. Recall that a single-layer perceptron could not do this operation. Now we will apply the three layer net.

# Example: Three-layer network for solving the Exclusive-OR operation

- The effect of the threshold applied to a neuron in the hidden layer is represented by its weight,  $\theta$ , connected to a fixed input equal to -1
- The initial weights and threshold levels are set randomly as follows:

$$w_{13} = 0.5, w_{14} = 0.9, w_{23} = 0.4, w_{24} = 1.0, w_{35} = -1.2, \\ w_{45} = 1.1, \theta_3 = 0.8, \theta_4 = -0.1 \text{ and } \theta_5 = 0.3.$$



# Example: Three-layer network for solving the Exclusive-OR operation

- We consider a training set where inputs  $x_1$  and  $x_2$  are equal to 1 and desired output  $y_{d,5}$  is 0. The actual output of neurons 3 and 4 in the hidden layers are calculated as:

$$y_3 = \text{sigmoid}(x_1 w_{13} + x_2 w_{23} - \theta_3) = 1 / \left[ 1 + e^{-(1 \cdot 0.5 + 1 \cdot 0.4 - 1 \cdot 0.8)} \right] = 0.5250$$
$$y_4 = \text{sigmoid}(x_1 w_{14} + x_2 w_{24} - \theta_4) = 1 / \left[ 1 + e^{-(1 \cdot 0.9 + 1 \cdot 1.0 + 1 \cdot 0.1)} \right] = 0.8808$$

- Now the actual output of neuron 5 in the output layer is determined as:

$$y_5 = \text{sigmoid}(y_3 w_{35} + y_4 w_{45} - \theta_5) = 1 / \left[ 1 + e^{(-0.5250 \cdot 1.2 + 0.8808 \cdot 1.1 - 1 \cdot 0.3)} \right] = 0.5097$$

Thus, the following error is obtained:

$$e = y_{d,5} - y_5 = 0 - 0.5097 = -0.5097$$

# Example: Three-layer network for solving the Exclusive-OR operation

- The next step is weight training. To update the weights and threshold levels in our network, we propagate the error,  $e$ , from the output layer backward to the input layer.
- First, we calculate the error gradient for neuron 5 in the output layer

$$\delta_5 = y_5(1-y_5)e = 0.5097 \cdot (1-0.5097) \cdot (-0.5097) = -0.1274$$

- Then we determine the weight corrections assuming that the learning rate parameter,  $\alpha$ , is equal to 0.1

$$\Delta w_{35} = \alpha \cdot y_3 \cdot \delta_5 = 0.1 \cdot 0.5250 \cdot (-0.1274) = -0.0067$$

$$\Delta w_{45} = \alpha \cdot y_4 \cdot \delta_5 = 0.1 \cdot 0.8808 \cdot (-0.1274) = -0.0112$$

$$\Delta \theta_5 = \alpha \cdot (-1) \cdot \delta_5 = 0.1 \cdot (-1) \cdot (-0.1274) = -0.0127$$

# Example: Three-layer network for solving the Exclusive-OR operation

- Now we calculate the error gradients for neuron 3 and 4 in the hidden layer:

$$\delta_3 = y_3(1-y_3) \cdot \delta_5 \cdot w_{35} = 0.5250 \cdot (1-0.5250) \cdot (-0.1274) \cdot (-1.2) = 0.0381$$

$$\delta_4 = y_4(1-y_4) \cdot \delta_5 \cdot w_{45} = 0.8808 \cdot (1-0.8808) \cdot (-0.1274) \cdot 1.1 = -0.0147$$

- We, then, determine the weight corrections:

$$\Delta w_{13} = \alpha \cdot x_1 \cdot \delta_3 = 0.1 \cdot 1 \cdot 0.0381 = 0.0038$$

$$\Delta w_{23} = \alpha \cdot x_2 \cdot \delta_3 = 0.1 \cdot 1 \cdot 0.0381 = 0.0038$$

$$\Delta \theta_3 = \alpha \cdot (-1) \cdot \delta_3 = 0.1 \cdot (-1) \cdot 0.0381 = -0.0038$$

$$\Delta w_{14} = \alpha \cdot x_1 \cdot \delta_4 = 0.1 \cdot 1 \cdot (-0.0147) = -0.0015$$

$$\Delta w_{24} = \alpha \cdot x_2 \cdot \delta_4 = 0.1 \cdot 1 \cdot (-0.0147) = -0.0015$$

$$\Delta \theta_4 = \alpha \cdot (-1) \cdot \delta_4 = 0.1 \cdot (-1) \cdot (-0.0147) = 0.0015$$

# Example: Three-layer network for solving the Exclusive-OR operation

- ❑ At last, we update all weights and thresholds
- ❑ The training process is updated till the sum of squared error is less than 0.001

$$w_{13} = w_{13} + \Delta w_{13} = 0.5 + 0.0038 = 0.5038$$

$$w_{14} = w_{14} + \Delta w_{14} = 0.9 - 0.0015 = 0.8985$$

$$w_{23} = w_{23} + \Delta w_{23} = 0.4 + 0.0038 = 0.4038$$

$$w_{24} = w_{24} + \Delta w_{24} = 1.0 - 0.0015 = 0.9985$$

$$w_{35} = w_{35} + \Delta w_{35} = -1.2 - 0.0067 = -1.2067$$

$$w_{45} = w_{45} + \Delta w_{45} = 1.1 - 0.0112 = 1.0888$$

$$\theta_3 = \theta_3 + \Delta \theta_3 = 0.8 - 0.0038 = 0.7962$$

$$\theta_4 = \theta_4 + \Delta \theta_4 = -0.1 + 0.0015 = -0.0985$$

$$\theta_5 = \theta_5 + \Delta \theta_5 = 0.3 + 0.0127 = 0.3127$$

# Example: Three-layer network for solving the Exclusive-OR operation

- The Final results of three layer network learning is:

Inputs		Desired output	Actual output	Error	Sum of squared errors
$x_1$	$x_2$	$y_d$	$y_5$	$e$	
1	1	0	0.0155	-0.0155	0.0010
0	1	1	0.9849	0.0151	
1	0	1	0.9849	0.0151	
0	0	0	0.0175	-0.0175	

# Gradient Descent

- ❑ Gradient descent is an iterative minimisation method. The gradient of the error function always shows in the direction of the steepest ascent of the error function.
- ❑ It is determined as the derivative of the activation function multiplied by the error at the neuron output

For Neuron  $k$  in the output layer

$$\delta_k(p) = \frac{\partial y_k(p)}{\partial X_k(p)} \times e_k(p)$$

Where,  $y_k(p)$  is the output of neuron  $k$  at iteration  $p$ , and  $X_k(p)$  is the net weighted input of neuron  $k$  at same iteration.

# Hopfield Network

40

- neural networks with feedback – Hopfield networks
- presence of such loops has a profound impact on the learning capability of the network
- After applying a new input, the network output is calculated and feedback to adjust the input. Then the output is calculated again, and the process is repeated until the output becomes constant.[Working mechanism of Recurrent network]

# Hopfield Network

41

- Refer Hopfield Network training algorithm in Negnevitskey's Book [page 212]

# Hopfield Network

42

- Storage memory, Energy based model
- Composed of binary threshold units with recurrent connections between them
- Recurrent network of non-linear units are generally hard to analyze. They behave in different ways:
  - ▣ Settle to stable state
  - ▣ Oscillates
  - ▣ Follows chaotic trajectories that can't be predicted far into future [input might not assure solution]

# Hopfield Network

43

- Hopfield realized the connections are symmetric, there is global energy function
  - ▣ Binary configuration of whole network has an energy
- Binary threshold decision rule cause the network to settle to minimum of this energy [if we apply downhill energy rule]

# Hopfield Network

44

- Global energy is the sum of many contributions. Each contribution depends on “one connection weight” and binary state of two neurons
- $E = -\sum_i s_i b_i - \sum_{i < j} s_i s_j \cdot w_i w_i$
- This quadratic energy function makes it possible for each unit to compute locally how its state affects the global energy
- Energy gap =  $\Delta E_i = E(s_i = 0) - E(s_i = 1) = b_i - \sum_j s_j w_{ij}$

# Hopfield Network

45

- Example:

# Kohonen Network

46

- self-organized maps ( Kohonen, 1982)
- Learning of neural network without the presence of teacher
- Competitive learning

# Kohonen Network

47

- In competitive learning, neurons compete among themselves to be activated
- While in Hebbian learning, several output neurons can be activated simultaneously, in competitive learning, only a single output neuron is active at any time.
- The output neuron that wins the “competition” is called the **winner-takes-all neuron**.

# Kohonen Network

48

- The basic idea of competitive learning was introduced in the early 1970s.
- In the late 1980s, Teuvo Kohonen introduced a special class of artificial neural networks called **self-organizing feature maps**. These maps are based on competitive learning.

# Kohonen Network

49

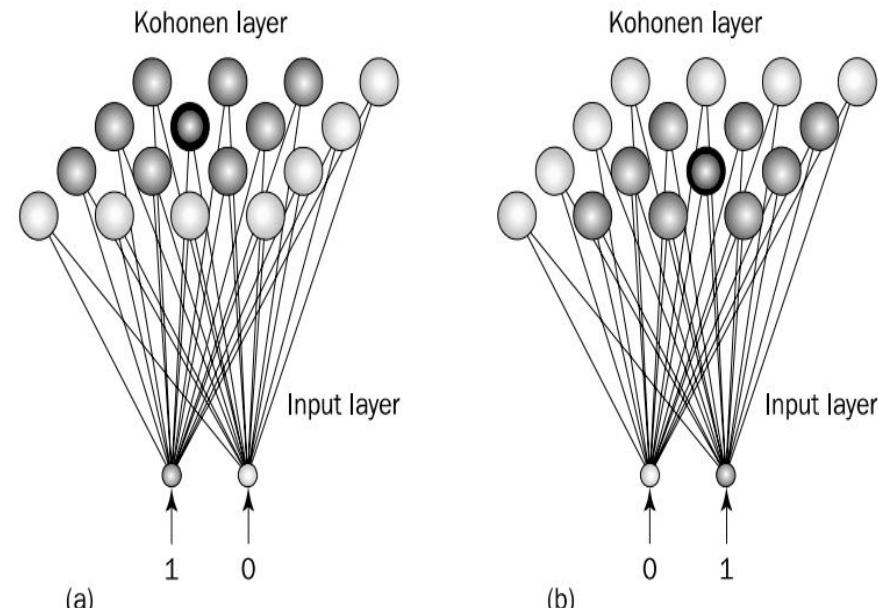
## What is self organizing feature map?

Our brain is dominated by the cerebral cortex, a very complex structure of billions of neurons and hundreds of billions of synapses. The cortex includes areas that are responsible for different human activities (motor, visual, auditory, somatosensory, etc.), and associated with different sensory inputs. We can say that each sensory input is mapped into a corresponding area of the cerebral cortex. **The cortex is a self-organizing computational map in the human brain.**

# Kohonen Network

50

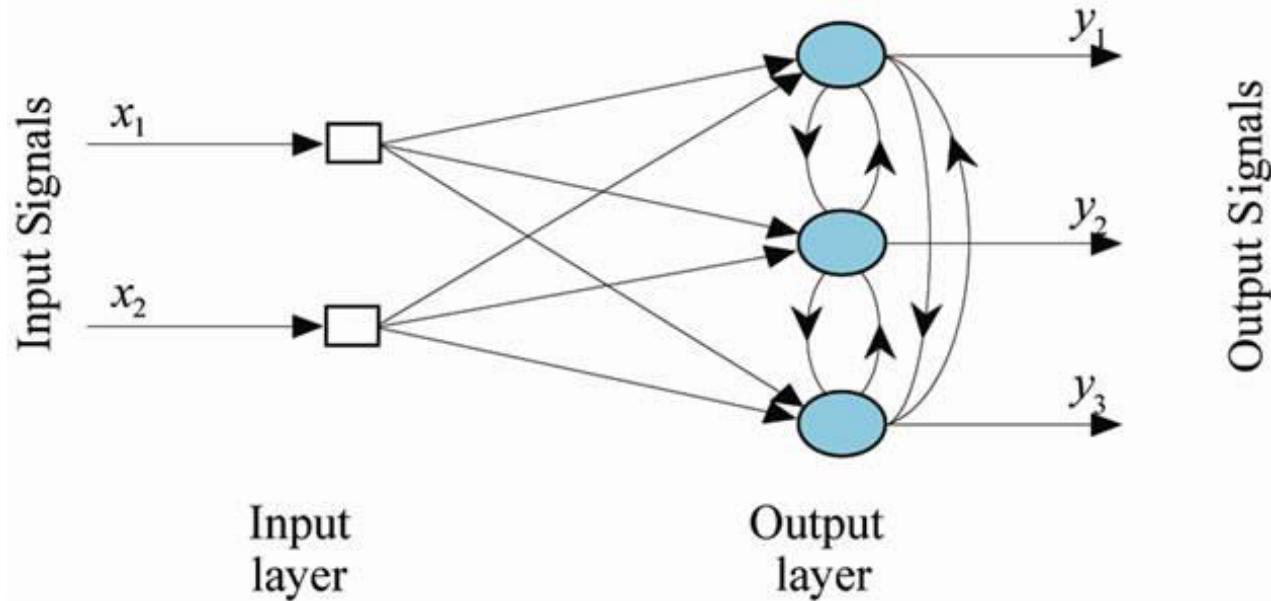
- The Kohonen model provides a topological mapping. It places a fixed number of input patterns from the input layer into a higher dimension output or Kohonen Layer
- Training in Kohonen Network begins with the winner's neighborhood of a fairly large size. Then, as training proceeds, the neighborhood size gradually decreases



# Kohonen Network

51

Architecture  
of the  
Kohonen  
Network



# Kohonen Network

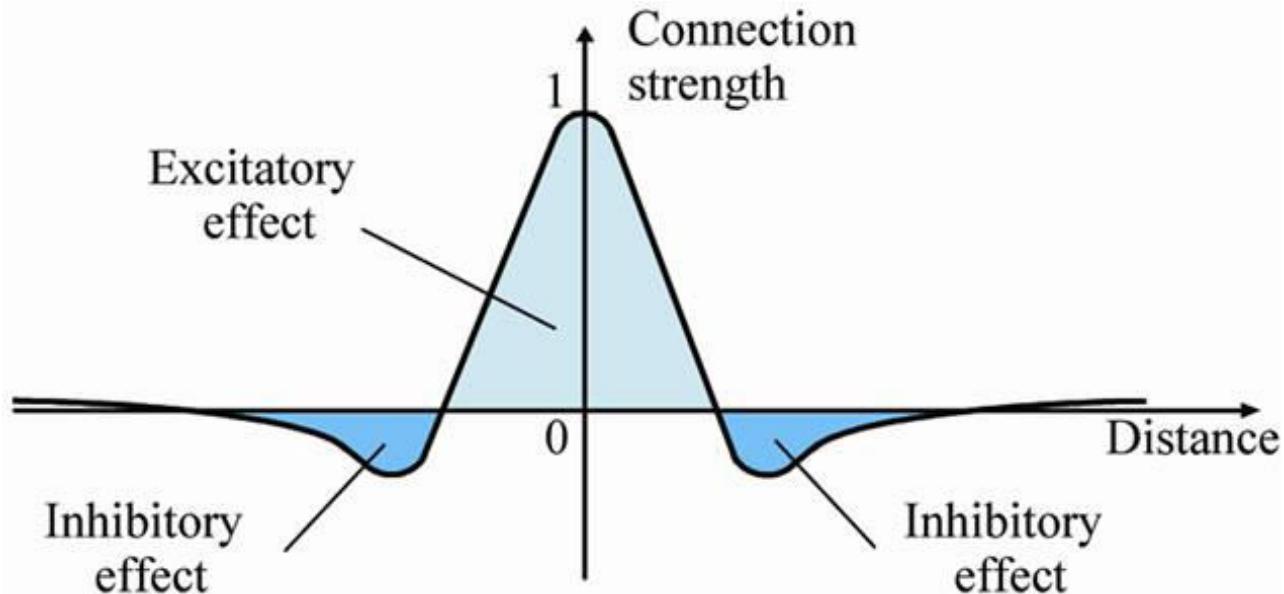
52

- The lateral connections are used to create a competition between neurons. The neuron with the largest activation level among all neurons in the output layer becomes the winner. This neuron is the only neuron that produces an output signal. The activity of all other neurons is suppressed in the competition.
- The lateral feedback connections produce excitatory or inhibitory effects, depending on the distance from the winning neuron. This is achieved by the use of a **Mexican hat function** which describes synaptic weights between neurons in the Kohonen layer.

# Kohonen Network

53

Mexican hat  
function



# Kohonen Network

54

- Mexican hat function represents the relationship between the distance from the winner-takes-all neuron and the strength of the connection within the Kohonen layer
- According to this function, the **near neighborhood** (a short-range lateral excitation area) has a strong excitatory effect, **remote neighborhood** (an inhibitory penumbra) has a mild inhibitory effect and **very remote neighborhood** (an area surrounding the inhibitory penumbra) has a weak excitatory effect, which is usually neglected

# Kohonen Network

55

- In the Kohonen network, a neuron learns by shifting its weight from inactive connections to active ones. Only the winning neuron and its neighborhood are allowed to learn. If a neuron does not respond to a given input pattern, then learning can not occur in that particular neuron
- The competitive learning rule defines the change in  $\Delta w_{ij}$  applied to synaptic weight  $w_{ij}$  as

$$\Delta w_{ij} = \begin{cases} \alpha(x_i - w_{ij}), & \text{if neuron } j \text{ wins the competition} \\ 0, & \text{if neuron } j \text{ loses the competition} \end{cases}$$

- Where  $x_i$  is the input signal and  $\alpha$  is the learning rate parameter

# Kohonen Network

- The overall effect of the competitive learning rule resides in moving the synaptic weight vector  $W_i$  of the winning neuron  $j$  towards the input pattern  $X$ . the matching Euclidean Distance between vectors.
- The Euclidean Distance between pair of n-by-1 vectors  $X$  and  $W_i$  is defined by

$$d = \|X - W_j\| = \left[ \sum_{i=1}^n (x_i - w_{ij})^2 \right]^{1/2}$$

- Where  $x_i$  and  $w_j$  are ith elements of the vectors  $X$  and  $W_i$  respectively

# Kohonen Network

57

- To identify the winning neuron ,  $j_X$  , that best matches the input vector  $X$ , we can apply following condition (Haykin, 1999)

$$j_X = \min_j \|X - W_j\|, \quad j = 1, 2, \dots, m$$

where  $m$  is the number of neurons in the Kohonen layer

# Kohonen Network

EX: Suppose that the two dimensional input vector  $X$  is presented to 3-dimensional Kohonen Network

$$X = \begin{bmatrix} 0.52 \\ 0.12 \end{bmatrix}$$

The initial weight vectors,  $\mathbf{W}_j$ , are given by

$$\mathbf{W}_1 = \begin{bmatrix} 0.27 \\ 0.81 \end{bmatrix} \quad \mathbf{W}_2 = \begin{bmatrix} 0.42 \\ 0.70 \end{bmatrix} \quad \mathbf{W}_3 = \begin{bmatrix} 0.43 \\ 0.21 \end{bmatrix}$$

# Kohonen Network

We find the winning (best-matching) neuron  $j_X$  using the minimum-distance Euclidean criterion:

$$d_1 = \sqrt{(x_1 - w_{11})^2 + (x_2 - w_{21})^2} = \sqrt{(0.52 - 0.27)^2 + (0.12 - 0.81)^2} = 0.73$$

$$d_2 = \sqrt{(x_1 - w_{12})^2 + (x_2 - w_{22})^2} = \sqrt{(0.52 - 0.42)^2 + (0.12 - 0.70)^2} = 0.59$$

$$d_3 = \sqrt{(x_1 - w_{13})^2 + (x_2 - w_{23})^2} = \sqrt{(0.52 - 0.43)^2 + (0.12 - 0.21)^2} = 0.13$$

Neuron 3 is the winner and its weight vector  $\mathbf{W}_3$  is updated according to the competitive learning rule.

$$\Delta w_{13} = \alpha (x_1 - w_{13}) = 0.1 (0.52 - 0.43) = 0.01$$

$$\Delta w_{23} = \alpha (x_2 - w_{23}) = 0.1 (0.12 - 0.21) = -0.01$$

# Kohonen Network

60

- The updated weight if vector  $\mathbf{W}_3$  at  $(p+1)$  is given by

$$\mathbf{W}_3(p+1) = \mathbf{W}_3(p) + \Delta\mathbf{W}_3(p) = \begin{bmatrix} 0.43 \\ 0.21 \end{bmatrix} + \begin{bmatrix} 0.01 \\ -0.01 \end{bmatrix} = \begin{bmatrix} 0.44 \\ 0.20 \end{bmatrix}$$

The weight vector  $\mathbf{W}_3$  of the winning neuron 3 becomes closer to the input vector  $\mathbf{X}$  with each iteration.

# Kohonen Network

61

- Competitive Learning Algorithm
  - Refer Page 209 of Negnevitsky

- “Radial Basis Function and Elastic Net Model” on your own

# Characteristics of ANN

- ❑ Adaptive learning
- ❑ Self-organization
- ❑ Error tolerance
- ❑ Real-time operation
- ❑ Parallel information processing

# Benefits and Limitations of ANN

Benefits	Limitations
Ability to tackle new kind of problems	Performs less well at tasks humans tend to find difficult
Robustness	Lack of explanation facilities
	Require large amount of test data

# Expert System

Definition of Expert System:

An Expert System is a collection of programmes or Computer Software that solves problems in the domain of interest. It is called system because it consists of both problem solving component and a support component.

The process of building Expert System is called knowledge engineering and is done by knowledge Engineer

# Expert System

66

**Expert systems provide the following important features:**

- Facility for non-expert personnel to solve problems that require some expertise
- Speedy solution
- Reliable solution
- Cost reduction
- Power to manage without human experts
- Wider areas of knowledge

**Use of expert systems is specially recommended when:**

- Human experts are difficult to find
- Human experts are expensive
- Knowledge improvement is important
- The available information is poor, partial, incomplete
- Problems are incompletely defined
- There is lack of knowledge among all those who need it
- The problem is rapidly changing legal rules and codes

# Architecture of an Expert System

67

Architecture of expert systems reflects the knowledge engineers' understanding of the methods representing knowledge and how to perform intelligent decision making task with the support of computer-based systems

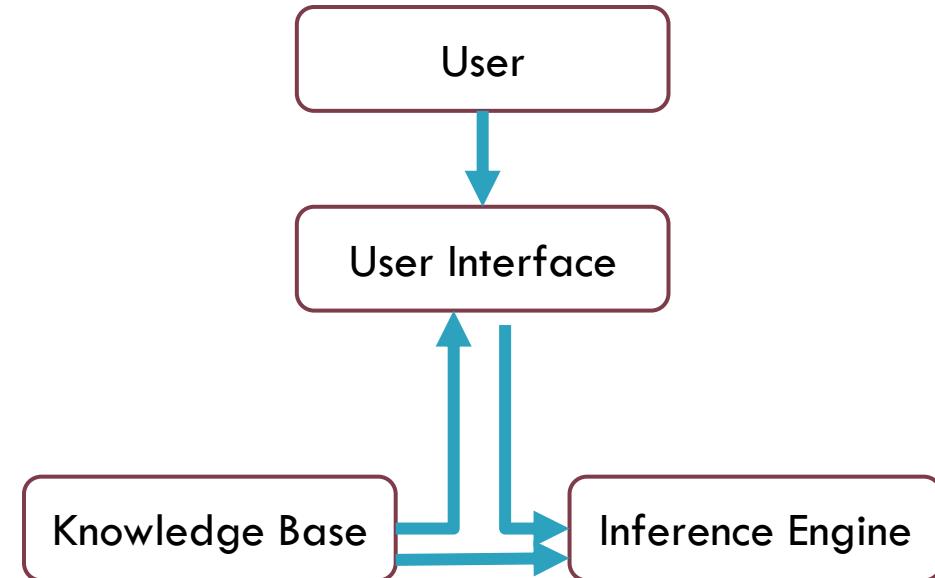


Fig: Architecture of Expert System

# Expert System

68

## □ Declarative Knowledge

□ descriptive representation of knowledge. It tells us facts: what things are. It is expressed in a factual statement, such as “There is a positive association between smoking and cancer.” Domain experts tell us about truths and associations. This type of knowledge is considered shallow, or surface-level, information that experts can verbalize. Declarative knowledge is especially important in the initial stage of knowledge acquisition.

# Expert System

69

- Procedural Knowledge
  - considers the manner in which things work under different sets of circumstances
  - The following is an example :“Compute the ratio between the price of a share and the earnings per share. If the ratio is larger than 12, stop your investigation. Your investment is too risky. If the ratio is less than 12, check the balance sheet.” Thus, procedural knowledge includes step-by-step sequences and how-to types of instructions; it may also include explanations.
  - Procedural knowledge involves automatic responses to stimuli.
  - It may also tell us how to use declarative knowledge and how to make inferences.

# Expert System

70

## Declarative Knowledge

- Declarative knowledge relates to a specific object.
- It includes information about the meaning, roles, environment, resources, activities, associations, and outcomes of the object

## Procedural Knowledge

- Procedural knowledge relates to the procedures used in the problem-solving process (e.g., information about problem definition, data gathering, the solution process, evaluation criteria).

# Development of an Expert System

71

Expert system design and development must be carefully programmed in success is desired.

The following are the main steps to be followed while developing expert system

a.	Outline Statement	b.	Knowledge acquisition
c.	Knowledge representation	d.	Prototype Development
e.	Testing	f.	Main Knowledge Acquisition
f.	Specification with detailed Information	g.	System Development
h.	Implementation	i.	Maintenance

# Expert System: Features

- Goal Driven (Backward Chaining) or Data Driven (Forward Chaining) Reasoning
- Coping with uncertainty
- Data Representation
- User Interface
- Explanations (ability to explain solution w.r.t. problem specification)
- Use knowledge rather than data
- Use symbolic representation for knowledge
- Should have meta knowledge

# Expert System: Advantages

- Provide consistent answers for repetitive decisions, processors and tasks
- Hold and maintain significant levels of information
- Encourage organization to clarify the logic of their decision making
- Ask question like human expertise

# Expert System: Disadvantages

- Lack of common sense needed in same decision making
- Can't make creative responses as human expert would in unusual circumstances
- Not able to explain their logic and reasoning
- Error may occur in the knowledge base and lead to wrong decision making
- Can't adapt to changing environment, unless knowledge base is changed

# Expert System: Example → DENDRAL

- First ES developed in late 1960s
- Designed to analyse mass spectra
- Based on the mass of fragments seen in the spectra, it would be possible to make inference as the nature of molecule tested, identifying functional groups or even the entire molecule
- DENDRAL used heuristic knowledge obtained from experienced chemists
- Uses forward chaining for reasoning
- Discovered number of structures previously unknown to climate experts

# Expert System - MYCIN

76

- An expert system for treating blood infections
- MYCIN would attempt to diagnose patients based on reported symptoms and medical test results
- Could ask some more information and lab test results for diagnosis
- It would recommend a course of treatment, if requested MYCIN would explain the reasoning that lead to its diagnosis and recommendations
- Uses about 500 production rules
- MYCIN operated at roughly the same level of competence as human specialists in blood infections
- Uses backward chaining for reasoning

# Natural Language Processing

# NLP

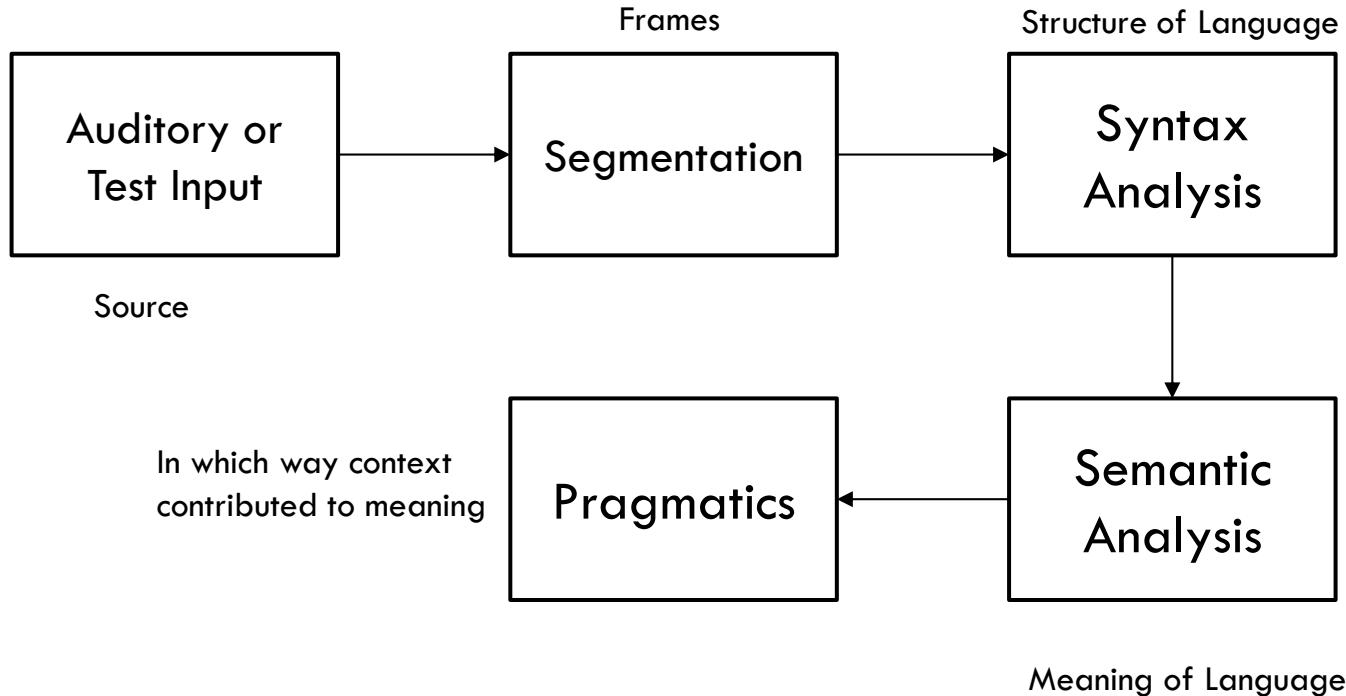
78

- Machine is considered intelligent if it can understand and manipulate Natural Language
- The language spoken by people : Natural Language

# Natural Language Processing

- NLP is one of the field of AI that processes or analyses written or spoken language
- NLP=NLG+NLU, where NLG is about generation and NLU is about understanding the natural language
- Understanding language requires a lot of knowledge

# NLP: Processes



# NLP

81

- Machine is considered intelligent if it can understand and manipulate Natural Language
- The language spoken by people : Natural Language
- NLP → AI method of communicating with an intelligent systems using Natural Language
- NLP is required when an intelligent system like robot to perform as per your instructions
- EX: getting result from Dialog Based clinical ES

# NLP

82

- NLP involves making computers to perform useful tasks with the natural languages human use
- I/O encompasses
  - Speech
  - Written Text

# NLP [Components]

83

- Natural Language Understanding
  - Understanding involves the following tasks
    - Mapping the given input in natural language into useful representations.
    - Analyzing different aspects of the language

# NLP [Component]

84

- Natural Language Generation
  - It is the process of producing meaningful phrases and sentences in the form of natural language from some internal representation.  
It involves :-
    - **Text planning** – It includes retrieving the relevant content from knowledge base.
    - **Sentence planning** – It includes choosing required words, forming meaningful phrases, setting tone of the sentence.
    - **Text Realization** – It is mapping sentence plan into sentence structure.  
The NLU is harder than NLG.

# NLP [Difficulties in NLU]

85

- **Lexical ambiguity** – It is at very primitive level such as word-level.  
For example, treating the word “board” as noun or verb?
- **Syntax Level ambiguity** – A sentence can be parsed in different ways.  
For example, “He lifted the beetle with red cap.” – Did he use cap to lift  
the beetle or he lifted a beetle that had red cap?
- **Referential ambiguity** – Referring to something using pronouns. For  
example, Rima went to Gauri. She said, “I am tired.” – Exactly who is tired?
- One input can mean different meanings.
- Many inputs can mean the same thing

# NLP [Steps]

86

- **Lexical Analysis** – It involves identifying and analyzing the structure of words. Lexicon of a language means the collection of words and phrases in a language. Lexical analysis is dividing the whole chunk of text into paragraphs, sentences, and words.
- **Syntactic Analysis Parsing** – It involves analysis of words in the sentence for grammar and arranging words in a manner that shows the relationship among the words. The sentence such as “The school goes to boy” is rejected by English syntactic analyzer.

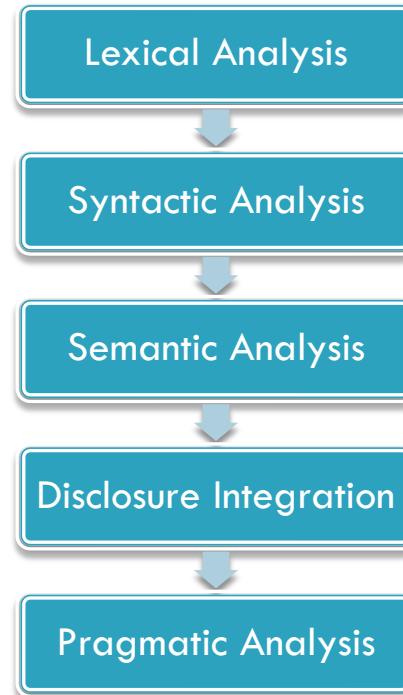
# NLP [Steps]

87

- **Semantic Analysis** – It draws the exact meaning or the dictionary meaning from the text. The text is checked for meaningfulness. It is done by mapping syntactic structures and objects in the task domain. The semantic analyzer disregards sentence such as “hot icecream”.
- **Discourse Integration** – The meaning of any sentence depends upon the meaning of the sentence just before it. In addition, it also brings about the meaning of immediately succeeding sentence.
- **Pragmatic Analysis** – During this, what was said is re-interpreted on what it actually meant. It involves deriving those aspects of language which require real world knowledge.

# NLP [Steps]

88

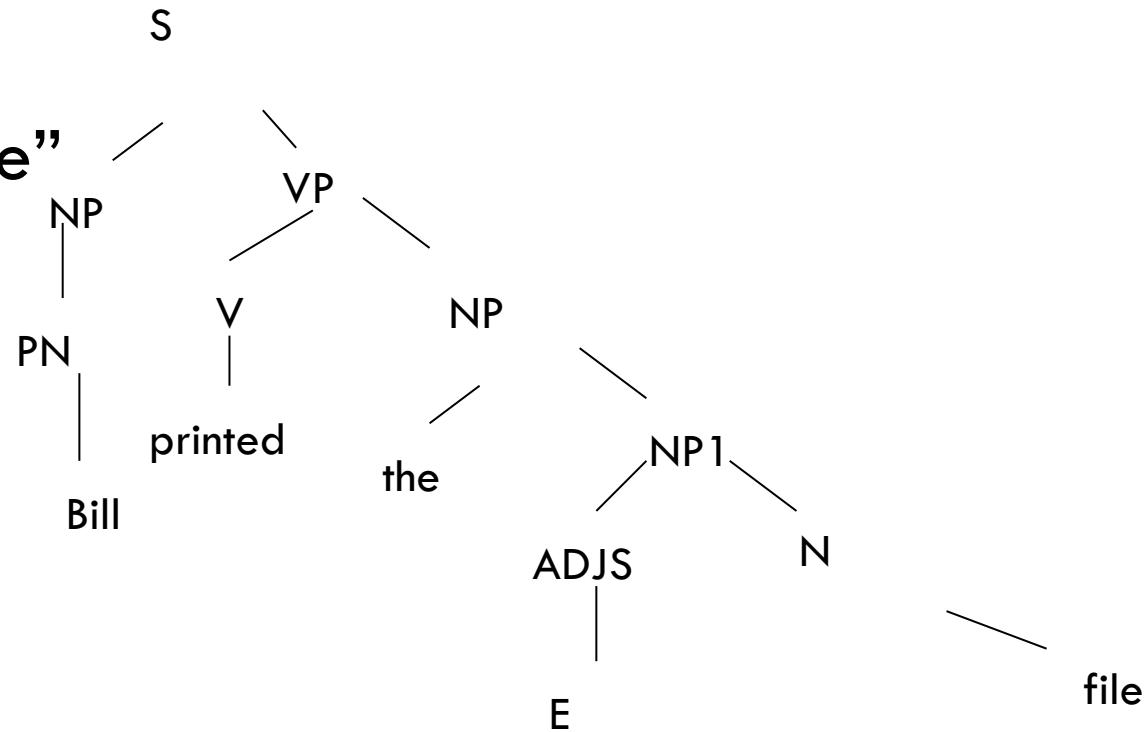


# NLP

89

## □ Parse Tree

“Bill Printed the file”



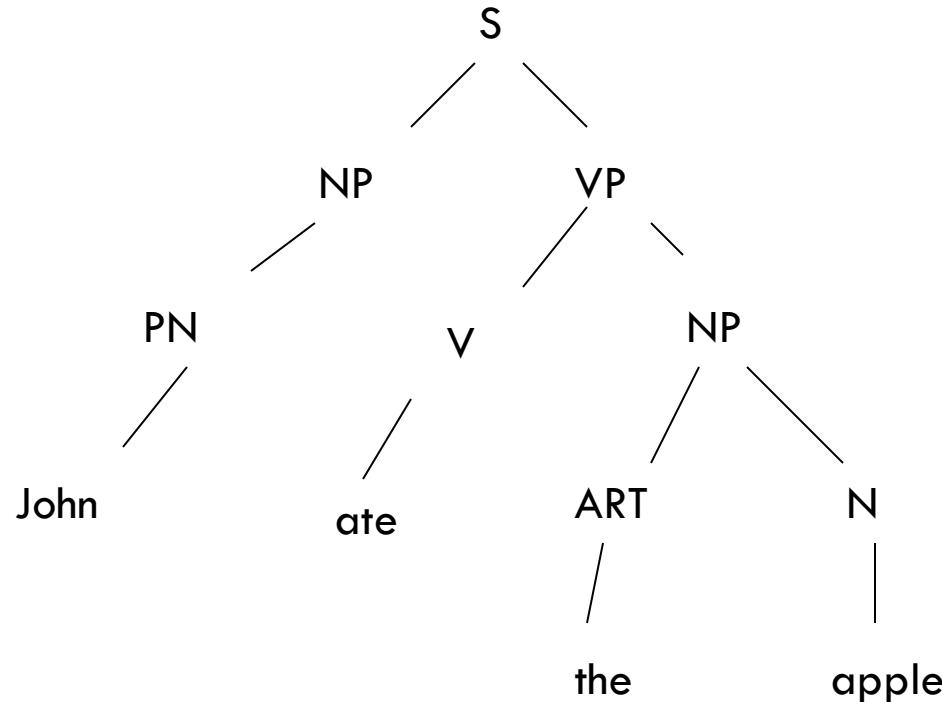
# NLP

90

## □ A parse tree :

John ate the apple.

1.  $S \rightarrow NP\ VP$
2.  $VP \rightarrow V\ NP$
3.  $NP \rightarrow NAME$
4.  $NP \rightarrow ART\ N$
5.  $NAME \rightarrow John$
6.  $V \rightarrow ate$
7.  $ART \rightarrow the$
8.  $N \rightarrow apple$



# References

- Russell, S. and Norvig, P., 2011, Artificial Intelligence: A Modern Approach, Pearson, India.
- Rich, E. and Knight, K., 2004, Artificial Intelligence, Tata McGraw hill, India.