CHAPTER : ONE
Introduction to Artificial Intelligence (AI)

Contents

- Introduction
- AI Vs. KNB
- Views of defining AI
- Strong AI Vs. Weak AI
- History of AI
- State of the Art
- Application of AI

Introduction

- What is intelligence ?

➢ Artificial Intelligence is composed of two words Artificial and Intelligence.

➢ Artificial defines "man-made," and intelligence defines "thinking power", or “the ability to learn and solve problems” hence Artificial Intelligence means "a man-made thinking power."

  – “The capacity to learn and solve problems” (Webster dictionary)
  – The ability to think and act rationally
  • Intelligence is the ability of ***observing, learning, remembering*** **and** ***reasoning*** something.

- This course focuses on ***computational intelligence***:
  – Trying to understand the computational principles behind intelligent behavior
- Top languages for AI
  – Python
  – Prolog
  – R
  – Lisp
  – Java

Cont….

- AI is the effort to develop computer based system(intelligent entities) that behave as human.
- It enable computers behave/act like human and emulate the reasoning power of humans
  - in order to do tasks that require human intelligence.

– Which task requires intelligence?
  – Complex arithmetic operations
    – For instance, Solving $2^{20} * 3^{50}$?
  – Every day/Mundane tasks
    – Example, Natural language understanding; face recognition
  – Expert tasks: which require specialists knowledge
    – Example, Medical diagnosis; computer maintenance

AI Foundations/Academic Disciplines important to AI.

- AI is a synergy/cooperation of d/t disciplines

A. Psychology/Cognitive Science: how do people behave, perceive, process information, represent knowledge.

- How do humans and animals think and act?
- Behaviorism, measuring of the percepts (or stimulus) given to an animal and its resulting actions (or response).
- Cognitive psychology, which views the brain as an information-processing device.
  - ✓ For example, Craik clearly specified the three key steps of designing good knowledge-based agent:-

    (1) The stimulus must be translated into an internal representation,

    (2) The representation is manipulated by cognitive processes to derive new internal representations, and

    (3) These are in turn retranslated back into action.

B. Neuroscience: Neurons as information processing units

- How do brains process information?
- Neuroscience is the study of the nervous system, particularly the brain.

Cont.

C. Philosophy: Logic, methods of reasoning, foundations of learning & rationality.

- Can formal rules be used to draw valid conclusions?
- How does the mind arise from a physical brain?
- Where does knowledge come from?
- How does knowledge lead to action?

D. Mathematics: Formal representation and proof, algorithms, computation, probability.

- What are the formal rules to draw valid conclusions?
- What can be computed?
- How do we reason with uncertain information?

E. Linguistics: knowledge representation, grammar

- How does language relate to thought?
- Natural Language Processing is a hybrid field of modern linguistic and AI.

F. Computer engineering: building fast computers.

- How can we build an efficient computer?
- For artificial intelligence to succeed, we need two things: intelligence and an artifact.
  - ✓ The computer has been the artifact of choice.
- The modern digital electronic computer passed various historical innovations, like:-
  - ✓ Operational Computer
  - ✓ Programmable Computer
  - ✓ Electronic Computer
- AI also owes a debt to the software side of computer science, which has supplied the operating systems, programming languages, and tools needed to write modern programs.

Computers vs. Human Brain

	Supercomputer	Personal Computer	Human Brain
Computational units	10^4 CPUs, 10^{12} transistors	4 CPUs, 10^9 transistors	10^{11} neurons
Storage units	10^{14} bits RAM	10^{11} bits RAM	10^{11} neurons
	10^{15} bits disk	10^{13} bits disk	10^{14} synapses
Cycle time	10^{-9} sec	10^{-9} sec	10^{-3} sec
Operations/sec	10^{15}	10^{10}	10^{17}
Memory updates/sec	10^{14}	10^{10}	10^{14}

- ✓ Figure is a crude comparison of the raw computational resources available to the IBM BLUE GENE supercomputer, a typical personal computer of 2008, and the human brain.
- ✓ The brain's numbers are essentially fixed, whereas the supercomputer's numbers have been increasing by a factor of 10 every 5 years or so, allowing it to achieve rough parity with the brain.
- ✓ The personal computer lags behind on all metrics except cycle time.

- ✓ Brains and digital computers have somewhat different properties.
- ✓ Previous figure shows that computers have a cycle time that is a million times faster than a brain.
- ✓ The brain makes up for that with far more storage and interconnection than even a high-end personal computer, although the largest supercomputers have a capacity that is similar to the brain's.
- ✓ (It should be noted, however, that the brain does not seem to use all of its neurons simultaneously.)
- ✓ Futurists make much of these numbers, pointing to an approaching **singularity** at which computers reach a superhuman level of performance (Vinge, 1993; Kurzweil, 2005), but the raw comparisons are not especially informative.
- ✓ Even with a computer of virtually unlimited capacity, we still would not know how to achieve the brain's level of intelligence.

Different people approach Al with different goals in mind,
Two important questions to ask are:

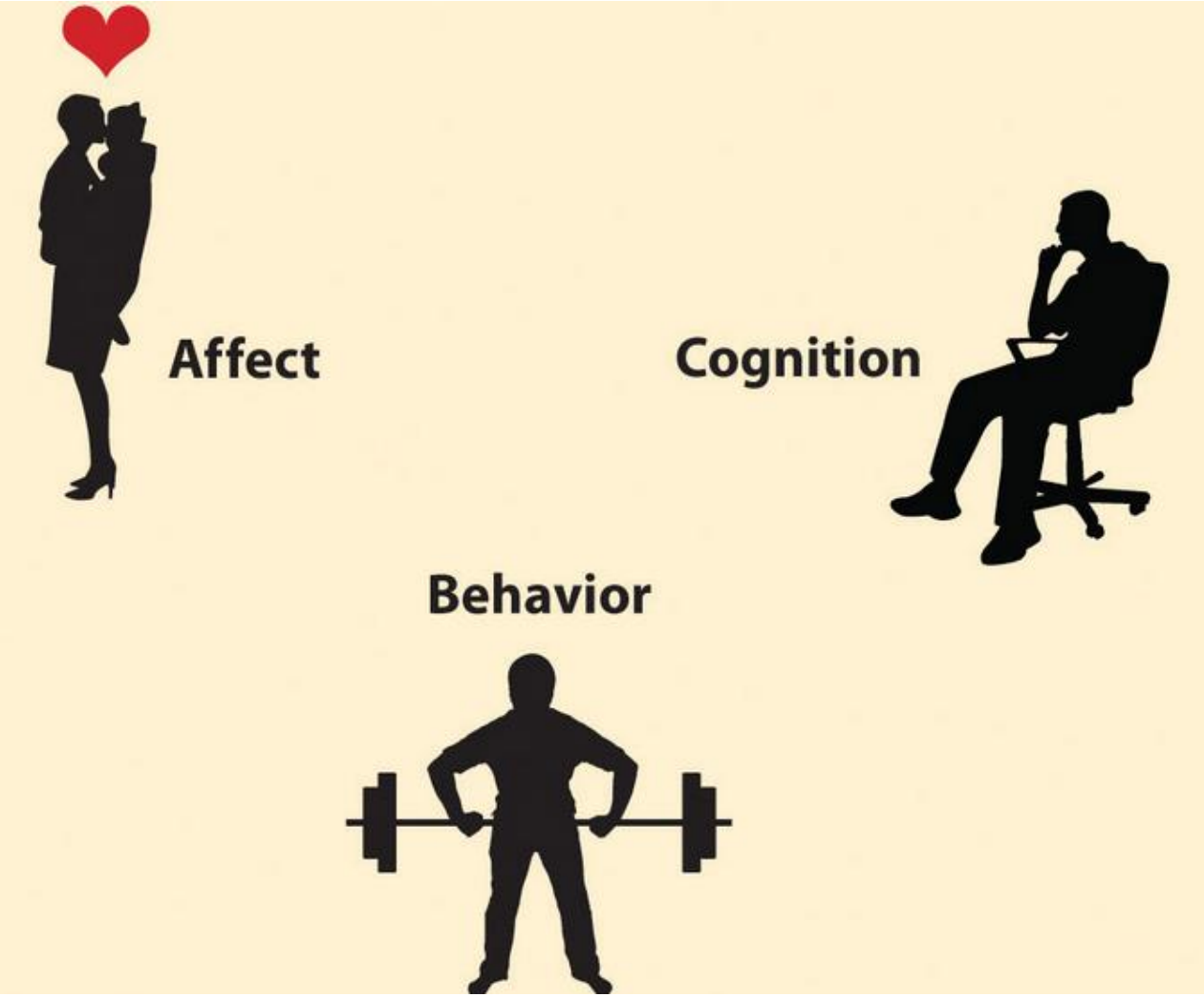
- Are we concerned with thinking or behavior?
- Do we want to model humans or work from an ideal standard?

✓ We adopt the view that intelligence is concerned mainly with rational action.

✓ Ideally, an intelligent agent takes the best possible action in a situation.

✓ We study the problem of building agents that are intelligent in this sense

- we rely on these three basic and interrelated human capacities:
  - Affect (feelings)
  - Behavior (interactions)
  - Cognition (thought)


Affect
Cognition
Behavior

Characteristics of Intelligent system:

- Use vast amount of knowledge
- Learn from experience and adopt to changing environment
- Interact with human using natural language and speech
- Respond in real time
- Tolerate error and ambiguity in communication

Artificial Intelligence vs. KBS

- Knowledge based system is part of Artificial Intelligence
- AI also requires extensive knowledge of the subject at hand.
  - AI program should have knowledge base
  - Knowledge representation is one of the most important and most active areas in AI.
  - AI programs should be learning in nature and update its knowledge accordingly.

What is Knowledge

- Knowledge includes facts about the real world entities and the relationship between them
  - It is an understanding gained through experience
  - Familiarity with the way to perform a task
  - An accumulation of facts, procedural rules, or heuristics
- Characteristics of Knowledge:
  - It is voluminous/big in nature and requires proper structuring.
  - It may be incomplete/partial and imprecise/indefinite.
  - It may keep on changing (dynamic).

Knowledge base

- Knowledge base is used to store facts and rules.
- In order to solve problems, the computer needs an internal model of the world.
  - This model contains, for example, the description of relevant objects and the relations between these objects.
  - All information must be stored in such a way that it is readily accessible.
- Various methods have been used for KR, such as logic, semantic networks, frames, scripts, etc...

Knowledge base systems (KBSs)

- is a computer program that reasons and uses a knowledge base to solve complex problem

- KBS - is a form of artificial intelligence (AI) which captures the **knowledge** of human experts to support decision-making

- Use **inference** to solve problems on a computer.

  Inference- is a conclusion reached on the basis of evidence and reasoning.

  – Ability to make the workings of the human mind understandable and executable on a computer.

Views of AI/Some possible definitions of AI are:-

- Four AI Definitions by Russell + Norvig
- AI is found on the premise that:
  - workings of human mind can be explained in terms of computation, and
  - computers can do the right thing given correct premises/properties and reasoning rules to achieve a specified goal.

Views of AI fall into four categories:

Thinking humanly	Thinking rationally
Acting humanly	Acting rationally

Views of defining AI

What is AI ?

Different scholars define AI differently

(A) AI as a system that think humanly

(B) AI as a system that think rationally

Concerned with thought processing and reasoning

Concerned with behaviors of agents

(C) AI as a system that Act humanly

(D) AI as a system that Act rationally

Views of defining AI

measures success of AI in terms of human being performance

(A) AI as a system that think humanly

(C) AI as a system that Act humanly

measures success of AI in terms of ideal concept of intelligence (rationality)

(B) AI as a system that think rationally

(D) AI as a system that Act rationally

I. Thinking humanly: The Cognitive Modeling

- Cognitive Science
  - ✓ The brain as an information processing machine.
  - ✓ Requires scientific theories of how the brain works
- How to understand cognition as a computational process?
  - ✓ Introspection, try to think about how we think
  - ✓ Predict and test behavior of human subjects
  - ✓ Image the brain, examine neurological data

- Reasons like humans do
  - Programs that behave like humans
  - Is concerned with modeling human thinking processes
- Requires understanding of the internal activities of the brain
  - see how humans behave in certain situations and see if you could make computers behave in that same way.
  - **Protocol analysis** (think aloud) can help in understanding how human beings solve a given problem

Example. write a program that plays chess.

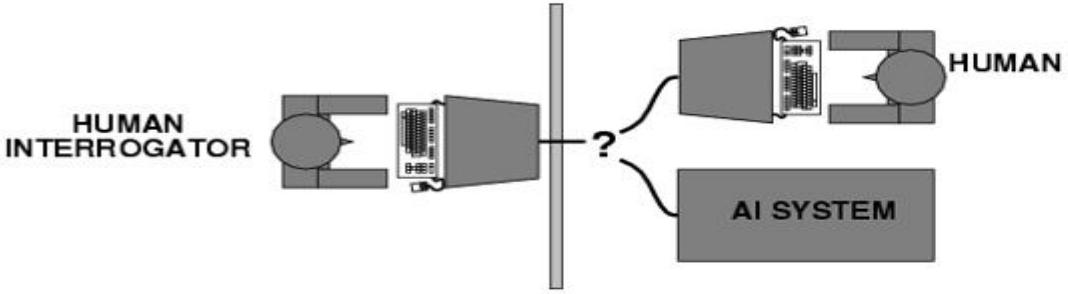
- Instead of making the best possible chess-playing program, you would make one that play chess like people do.

II. Acting humanly: The Turing Test

Can machines act like human do? Can machines behave intelligently?

- Acting like humans requires AI programs to interact with people

- Turing Test:
- What capabilities would a computer need to have to pass the Turing Test?
  - ✓ Natural Language Processing
  - ✓ Knowledge Representation
  - ✓ Automated Reasoning
  - ✓ Machine Learning



- ***Turing Test Criticisms***
  - ✓ There are some potential problems with the Turing Test, for example:-
    - Some human behavior is not intelligent
    - Some intelligent behavior may not be human
    - Human observers may be easy to fool.
    - The *"Chinese Room Argument"*, that is one may simulate intelligence without having true intelligence
  - ✓ Is passing the Turing test a good scientific goal?

III. Thinking Rationally: The Laws of Thought

- Idealized or "right" way of thinking.
- **Logic**
  - ✓ Patterns of argument that always yield correct conclusions when supplied with correct premises

    "Socrates is a man;

    All men are mortal;

    Therefore Socrates is mortal."
- Beginning with Aristotle, philosophers and mathematicians have attempted to formalize the rules of logical thought.

Logical Approach to AI

- ✓ Describe problem in formal logical notation and apply general deduction procedures to solve it.
- ✓ Problems with the logical approach are:-
    - Computational complexity of finding the solution.
    - Describing real-world problems and knowledge in logical notation.
    - A lot of intelligent or “rational” behavior has nothing to do with logic.

IV. Acting rationally: The rational agent

- A rational agent is one that acts to achieve the best outcome.
  - ✓ Goals are application-dependent and are expressed in terms of the utility of outcomes.
  - ✓ Being rational means maximizing your expected utility.
- This definition of rationality only concerns the decisions/actions that are made, not the cognitive process behind them.

•Central to our approach to AI

- **Doing the right thing** so as to achieve one's goal, given one's beliefs.
  - **AI** is the study and construction of rational agents (an agent that perceives and acts)
- Rational action requires the ability to represent knowledge and reason with it so as to reach good decision.
  - Learning for better understanding of how the world works

The State of The Art

- Robotic Vehicles
- Speech Recognition
- Autonomous Planning and Scheduling
- Game Playing
- Spam Fighting
- Fraud Detection
- Robotics
- Machine Translation
- Vision
- Drones
- Medical diagnosis

What do we get from using AI technology instead of previous reactive technology?

Following are some main advantages of Artificial Intelligence:

A. High Accuracy with fewer errors as it made decisions as pre-experience or information.

B. High-Speed: AI systems can be of very high-speed and fast-decision making

C. High reliability: AI machines are highly reliable and can perform the same action multiple times with high accuracy.

D. **Useful for risky areas:** AI machines can be helpful in situations such as defusing a bomb, exploring the ocean floor, where to employ a human can be risky.

E. Digital Assistant: AI technology is currently used by various E-commerce websites to show the products as per customer requirements.

F. Useful as a public utility: Such as a self driving car ,facial/message recognition for security purposes, Natural language processing (for search engines, for spelling checker etc.

Some of the disadvantages of AI are:

A. High Cost: The hardware and software requirement of AI is very costly as it requires lots of resource to meet current world requirements.

B. Can't think out of the box: Even we are making smarter machines with AI, but still they cannot work out of the box, as the robot will only do that work for which they are **trained**, or programmed.

C. No feelings and emotions: AI machines can be an outstanding performer, but still it does not have the feeling so it cannot make any kind of emotional attachment with humans, and may sometime be **harmful** for users if the proper care is not taken.

D. Increase dependence on machines: With the increment of technology, people are getting more dependent on devices and hence they are losing their mental capabilities.

E. No Original Creativity: As humans are so creative and can imagine some new ideas but still AI machines cannot beat this power of human intelligence and cannot be creative and imaginative.

History of AI

- Artificial Intelligence is not a new word and not a new technology for researchers.
- This technology is much older than you would imagine.
- Even there are the myths of Mechanical men in Ancient Greek and Egyptian Myths.
- The following are some milestones in the history of AI which define the journey from the AI generation to till date development (see Figure 3.3).

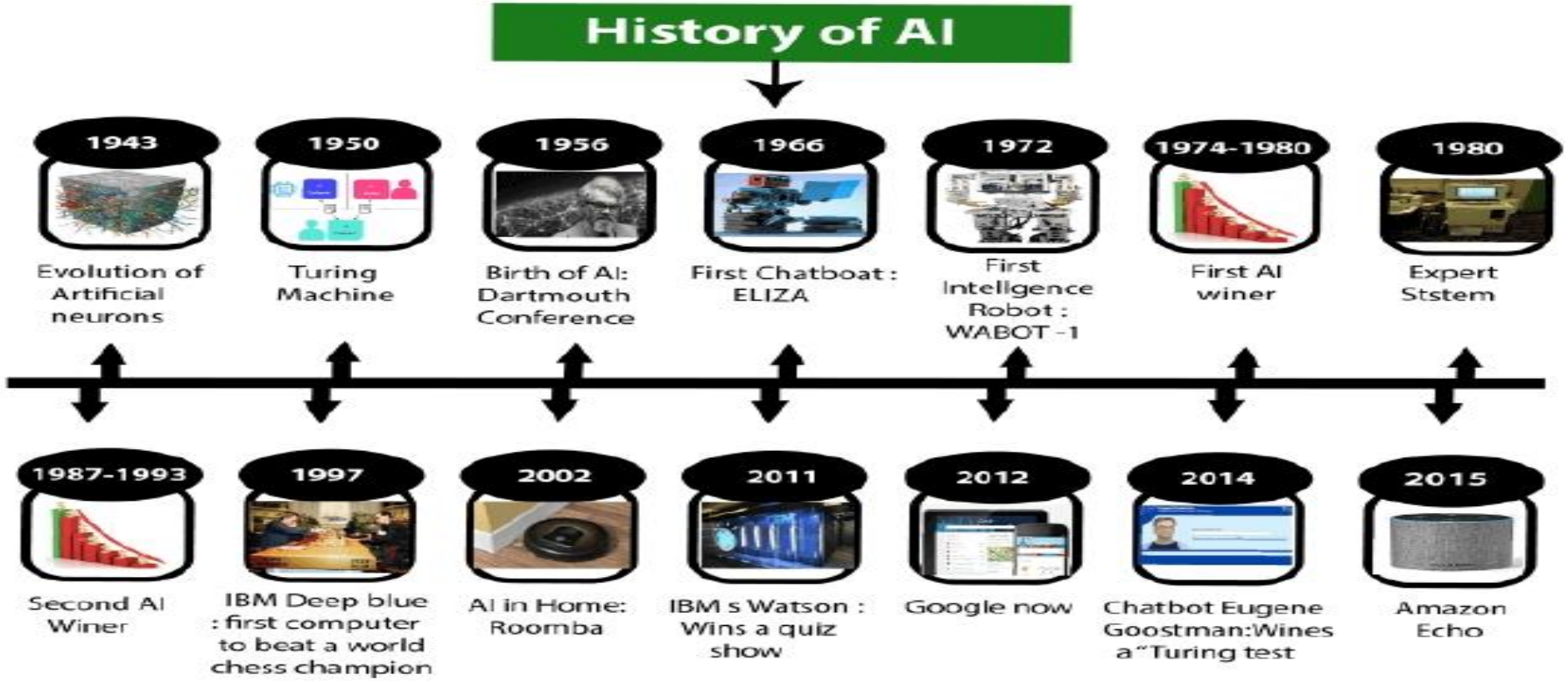


Figure 3.3 History of Artificial Intelligence (AI)

A. Maturation of Artificial Intelligence (1943-1952)

- **The year 1943**: Warren McCulloch and Walter pits are **proposed a model of artificial neurons**.
- **The year 1949**: Donald Hebb modifying the **connection strength between neurons.** His rule is now called Hebbian learning.
- **The year 1950**: Alan Turing proposed a test can check the machine's ability to exhibit intelligent behavior **equivalent to human intelligence**, called a **Turing test**.

B. The birth of Artificial Intelligence (1952-1956)

- **The year 1955**: An Allen Newell and Herbert A. Simon created the "**first artificial intelligence program**" Which was named "Logic Theorist".
- **The year 1956**: John McCarthy ,AI coined as an academic field. At that time high-level computer languages such as **FORTRAN, LISP, or COBOL** were invented.

C. The golden years-Early enthusiasm (1956-1974)

➢ **The year 1966**: Joseph Weizenbaum created the **first chatbot** in 1966, which was named as **ELIZA**.

➢ **The year 1972**: The first intelligent **humanoid robot** was built in Japan which was named **WABOT-1**.

D. The first AI winter (1974-1980)

➢ AI winter refers to the time period where computer scientists dealt with a severe **shortage of funding** from the **government** for **AI researches**.

➢ During AI winters, an interest in publicity/promotion on artificial intelligence was decreased.

E. A boom of AI (1980-1987)

➢ **The year 1980**: AI came back with "**Expert System**". Expert systems were programmed that emulate the decision-making ability of a human expert.

➢ In the Year 1980, the first national conference of the American Association of Artificial Intelligence was held at Stanford University.

F. The second AI winter (1987-1993)

➢ The duration between the years 1987 to 1993 was the second AI Winter duration.

➢ Again, **Investors and government stopped in funding for AI research** due to high cost but not efficient results.

➢ The expert system such as XCON was very cost-effective.

➢ (**XCON** /Expert configure ,automatically selecting the computer **system** components based on the customer's requirements

- Examples are: MYCIN: It could identify various **bacteria** that could cause **acute infections**.
- DENDRAL: Used for chemical analysis to predict molecular structure.
- PXDES: Used to **predict** the degree and type of **lung cancer**.

G. The emergence/rise of intelligent agents (1993-2011)

➢ **The year 1997**: In the year 1997, IBM Deep Blue beats world chess champion, Gary Kasparov, and became the first computer to beat a world chess champion.

➢ **The year 2002**: for the first time, AI entered the **home** in the form of **Roomba**, a **vacuum cleaner**.

➢ **The year 2006**: AI came into the Business world until the year 2006. Companies like **Facebook, Twitter, and Netflix** also started using AI.

H. **Deep learning**, **big data and artificial general intelligence** (2011-present)

➢ **The year 2011**: In the year 2011, IBM's Watson won jeopardy, a **quiz** show, where it had to solve complex questions as well as challenges.

Watson had proved that it could understand natural language and can solve tricky questions quickly.

➢ **The year 2012**: Google has launched an Android app feature "**Google now**", which was able to provide information to the user as a **prediction**.

➢ **The year 2014**: In the year 2014, **Chatbot** "Eugene Goostman" won a competition in the infamous "**Turing test."**

➢ **The year 2018**: The "Project Debater" from IBM debated on complex topics with two master debaters and also performed extremely well.

- ✓ Nowadays companies like Google, Facebook, IBM, and **Amazon** are working with AI and creating amazing devices.
- ✓ The future of Artificial Intelligence is inspiring and will come with high intelligence.

How to make computers act like humans?

The following sub-fields are emerged

- **Natural Language processing** (enable computers communicate in human language, English, Amharic, ..)
- **Knowledge representation** (schemes to store information, both facts and inferences, before and during interrogation/interview/questioning)
- **Automated reasoning** (use stored information to answer questions and to draw new conclusions)
- **Machine learning** (adapt to new circumstances/conditions and accumulate/collect knowledge)
- **Computer vision** (recognize/identify objects based on patterns/designs in the same way as the human visual system does)
- **Robotics** (produce mechanical device capable of controlled motion; which enable computers to see, hear & take actions)

- Is AI equals human intelligence? Can we create a KBS called mind?

Types of AI

- Artificial Intelligence can be divided into two categorization which are based on capabilities and based on functionally of AI.

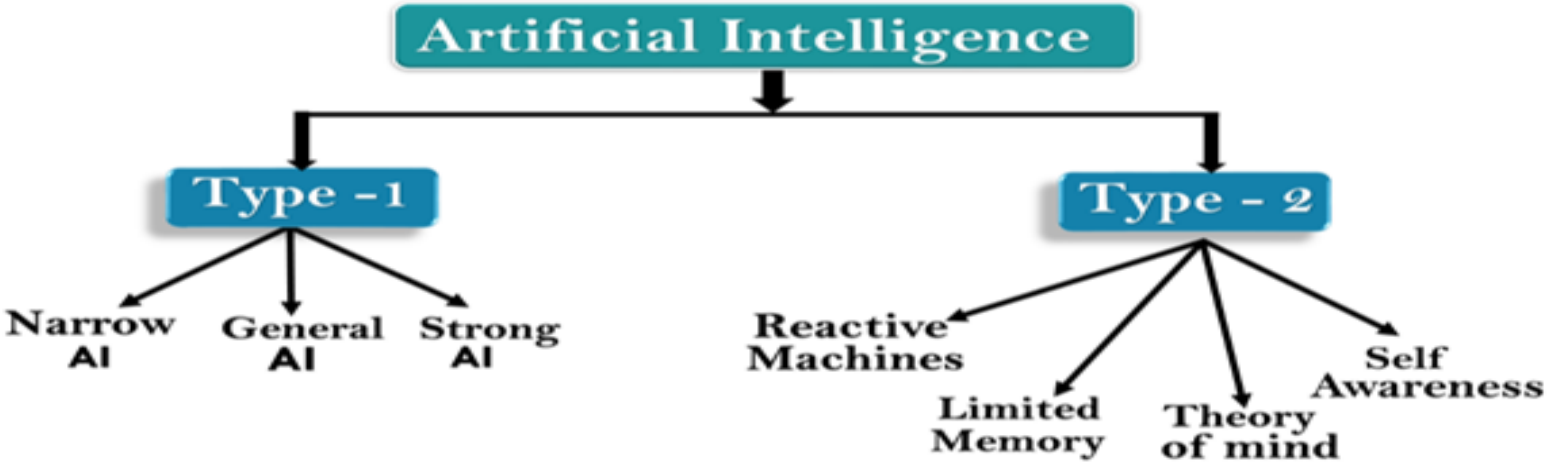


Figure types of Artificial Intelligence (AI)

A. **Based on Capabilities**

1. ***Weak AI or Narrow AI:*** is a type of AI which is able to perform a dedicated task with intelligence.

- Narrow AI , it is only trained for one specific task.
- Some Examples of Narrow AI are Google translate, playing chess, self-driving cars, speech recognition, and image recognition.

2. General AI: is a type of intelligence that could perform any intellectual task with efficiency like a human.

- The idea behind the general AI to make such a system that could be smarter and think like a human on its own.
- Currently, there is no such system exists which could come under general AI and can perform any task as perfect as a human.
- The worldwide researchers are now focused on developing machines with General AI.
- As systems with general AI **are still under research**, and it will take lots of effort and time to develop such systems.

3. Super AI: is a level of Intelligence of Systems at which machines could surpass human intelligence, and can perform any task **better than** a human with cognitive properties.

- Some key characteristics of strong AI include capability include the ability to think, to reason solve the puzzle, make judgments, plan, learn, and communicate on its own.

B. Based on the functionality

1. Reactive Machines

- Purely reactive machines are the most basic types of Artificial Intelligence.
- Such AI systems **do not store memories** or past experiences for future actions.
- These machines only focus on current scenarios and react on it as per possible best action.
- IBM's Deep Blue system is an example of reactive machines.
- Google's AlphaGo is also an example of reactive machines

2. Limited Memory

- Limited memory machines **can store past experiences** or some data for a short period of time.
- These machines can use stored data for a limited time period only.
- Self-driving cars are one of the best examples of Limited Memory systems.
- These cars can store the recent speed of nearby cars, the distance of other cars, speed limits, and other information to navigate the road.

3. Theory of Mind

- Theory of Mind AI should understand **human emotions**, people, beliefs, and be able to interact socially **like humans**.
- This type of AI machines is still not developed, but researchers are making lots of efforts and improvement for developing such AI machines.

4. Self-Awareness

- Self-awareness AI is the **future of Artificial** Intelligence.
- These machines will be super intelligent and will have **their own consciousness**, sentiments, and self-awareness.
- These machines will be **smarter than the human** mind.

- Self-Awareness AI does not exist in reality still and it is a hypothetical concept.

Applications of AI

- AI used to solve complex problems in efficient way in multiple industries, such as Healthcare, entertainment, finance, education, etc.
- AI is making our daily life more comfortable and faster.

A. AI in agriculture:- Agriculture is an area that requires various resources, labor, money, and time for the best result.

- ✓ Now a day's agriculture is becoming digital, and AI is emerging in this field.
  - For example agriculture robotics, crop monitoring, and predictive analysis.

B. AI in Healthcare: In the last, five to ten years, AI becoming more advantageous for the healthcare industry and going to have a significant impact on this industry to make a better and faster diagnosis than humans.

C. AI in education:

- ✓ AI can automate grading so that the tutor can have more time to teach.
- ✓ AI chatbot can communicate with students as a teaching assistant.
- ✓ AI in the future can be work as a personal virtual tutor for students, which will be accessible easily at any time and any place.

D. AI in Finance and E-commerce

- ✓ AI and finance industries are the best matches for each other.
- ✓ AI is providing a competitive edge to the e-commerce industry, and it is becoming more demanding in the e-commerce business. AI is helping shoppers to discover associated products with recommended size, color, or even brand.

E. AI in Gaming

- ✓ AI can be used for gaming purposes.
- ✓ The AI machines can play strategic games like chess, where the machine needs to think of a large number of possible places.

F. . AI in Data Security

- ✓ AI can be used to make your data more safe and secure.
- ✓ Some examples such as AEG bot, AI2 Platform, are used to determine software bugs and cyber-attacks in a better way.

G. AI in Social Media

- ✓ Social Media sites such as Facebook, Twitter, and Snapchat contain billions of user profiles, which need to be stored and managed in a very efficient way.

H. AI in Travel &Transport

- ✓ AI is capable of doing various travel related works such as from making travel arrangements to suggesting the hotels, flights, and best routes to the customers.

I. AI in the Automotive Industry

- ✓ Various Industries are currently working for developing self-driven cars which can make your journey more safe and secure.

J. AI in Robotics:

- ✓ Humanoid Robots are the best examples for AI in robotics, recently the intelligent Humanoid robot named Erica and Sophia has been developed which can talk and behave like humans.

K. AI in Entertainment

- ✓ We are currently using some AI-based applications in our daily life with some entertainment services such as Netflix or Amazon.

Sample AI application with hands activity

A. Commuting

- ✓ Google's AI-Powered Predictions
- ✓ Ridesharing Apps Like Uber and Lyft
- ✓ Commercial Flights Use an AI Autopilot

B. Email

- ✓ Spam Filters
- ✓ Smart Email Categorization

C. Social Networking

- ✓ Facebook - When you upload photos to Facebook, the service automatically highlights faces and suggests friends tag.
- ✓ Pinterest - Pinterest uses computer vision, an application of AI where computers are taught to "see," in order to automatically identify objects in images (or "pins") and then recommend visually similar pins.
- ✓ Instagram -Uses machine learning to identify the contextual meaning of emoji, which have been steadily replacing slang (for instance, a laughing emoji could replace "lol")
- ✓ Snapchat - Snapchat introduced facial filters, called Lenses, in 2015.
- ✓ These filters track facial movements, allowing users to add animated effects or digital masks that adjust when their faces moved.

D. Online Shopping

- ✓ Search - Your Amazon searches (“ironing board”, “pizza stone”, “Android charger”, etc.) quickly return a list of the most relevant products related to your search
- ✓ Recommendations - You see recommendations for products you’re interested in as “customers who viewed this item also viewed.

E. Mobile Use

- ➢ Voice-to-Text - By pressing a button or saying a particular phrase (“Ok Google”, for example), you can start speaking and your phone converts the audio into text.
- ➢ Smart Personal Assistants - Now that voice-to-text technology is accurate enough to rely on for basic conversation, it has become the control interface for a new generation of smart personal assistants.
- ➢ Siri and Google Now (now succeeded by the more sophisticated Google Assistant),
- ➢ Alexa, an AI-powered personal assistant that accepts voice commands to create to-do lists, order items online, set reminders, and answer questions (via internet searches)
- ➢ Echo (and later, Dot) smart speakers that allow you to integrate Alexa into your living room and use voice commands to ask natural language questions, play music, order pizza, hail an Uber, and integrate with smart home devices.
- ➢ Microsoft has followed suit with Cortana, its own AI assistant that comes pre-loaded on Windows computers and Microsoft smartphones.

AI Successes

- **Deep Blue** defeated the leading world chess champion Garry Kasparov in 1997.

IBM's Deep Blue World Champion Garry Kasparov

- **Deep Blue** was a chess-playing computer developed by IBM
- First computer chess-playing system to win a chess match against a ruling world champion under regular time controls.

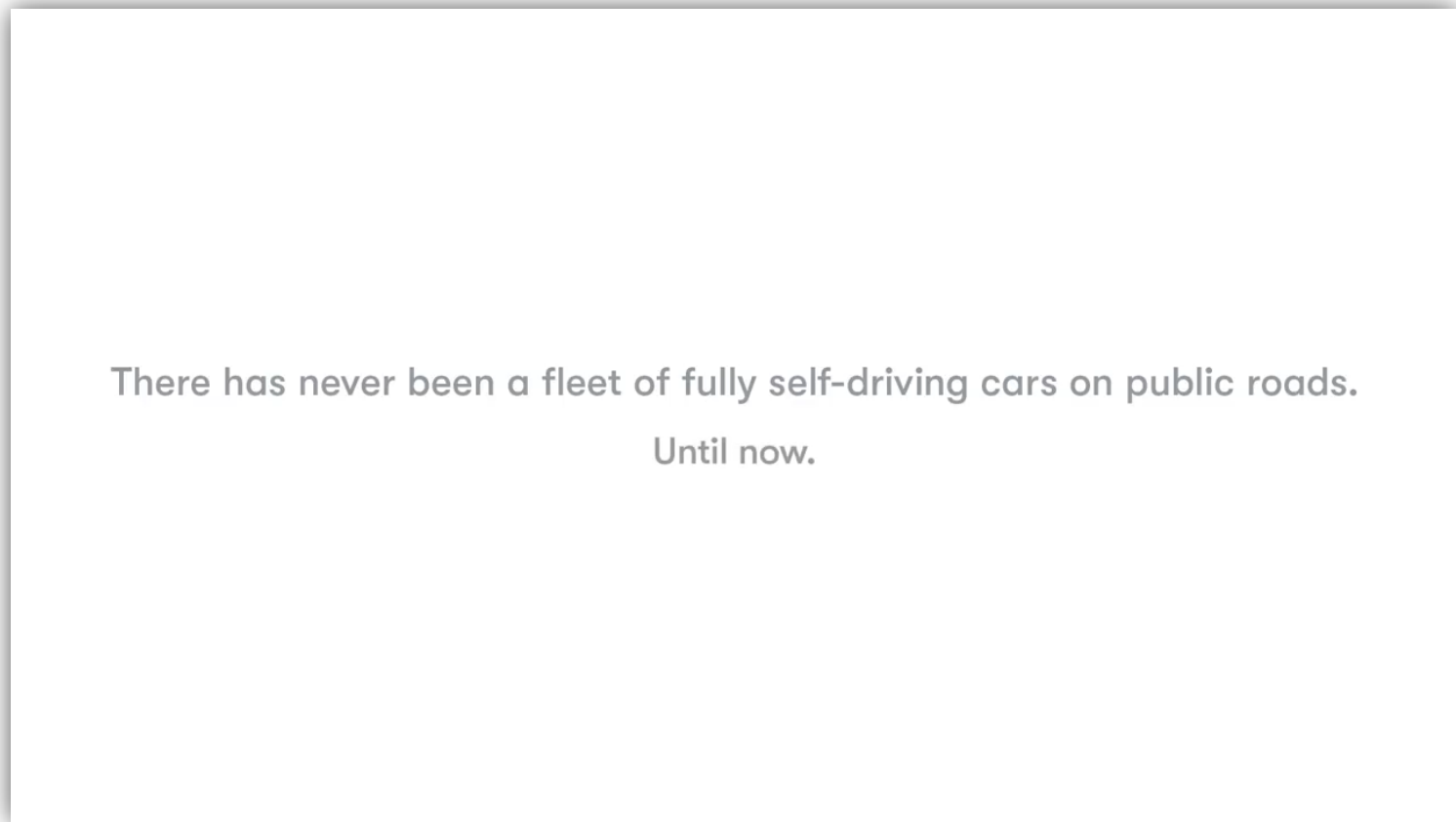
Cont.

- Watson - The computer system was initially developed to answer questions on the quiz show Jeopardy.

- In 2011, the Watson computer system competed on Jeopardy! against legendary champions Brad Rutter and Ken Jennings winning the first place prize of $1 million.

Sophia - From Hanson Robotics - 2016

Google Self Driving Cars



World's first AI presenter unveiled in China - (9-11-2018)
The World’s First AI News Anchor

Cont..

- Games: chess, checkers, poker, etc.
- Physical skills: driving a car, driving a motorcycle, flying a plane or helicopter, playing soccer, vacuuming, etc.
- Art: painting, composing music, performing music, etc.
- Language: machine translation, speech recognition, character recognition, etc.
- Vision: face recognition, face detection, motion tracking, etc.
- Commerce and industry: page rank for searching, fraud detection, stock market investing, etc.

Challenges Ahead

- Note that the examples we discussed so far all involve quite specific tasks.
- The systems lack a level of generality and adaptability. They can't easily (if at all) switch context.
- Key issue:
  - knowledge acquisition/ achievment bottleneck
  - Lack of general common sense knowledge.
  - Big data

Some extra discussion points

1) Discuss one of the following concepts.
   i. Knowledge based system:
      - What is KBS?
      - KBS vs. ES vs. AI
      - Knowledge acquisition, knowledge modeling and knowledge representation (semantic networks, frame, production system, ontology)
   ii. Reasoning:
      - What is reasoning, Case based reasoning, probabilistic reasoning, fuzzy reasoning, rule-based reasoning
   iii. Learning:
      - What is Machine learning? Support Vector Machine, Hidden Markov Model, Bayesian Belief Network

2) Explain the following areas of AI and how they work ?
   i. Natural Language Processing
   ii. Natural Language Generation
   iii. Speech Recognition, Speech synthesis, Speaker Identification
   iv. Optical Character Recognition (OCR),Writer identification, Face recognition
   v. Computer vision and robotics

CHAPTER : TWO
Intelligent Agent

Content

- Intelligent Agent
- Agent Schematic
- Types of Agent
- Rational Agents
- Specifying Task Environments
- Types of Environments

Intelligent Agent

- I want to build a robot that will
  - Clean my house
  - Information filtering agents
  - Cook when I don’t want to
  - Wash my clothes
  - Cut my hair
  - Fix my car (or take it to be fixed)
  - Take a note when I am in a meeting
  - Handle my emails

i.e. do the things that I don’t feel like doing…

- AI is the science of building machines (agents) that act rationally with respect to a goal.

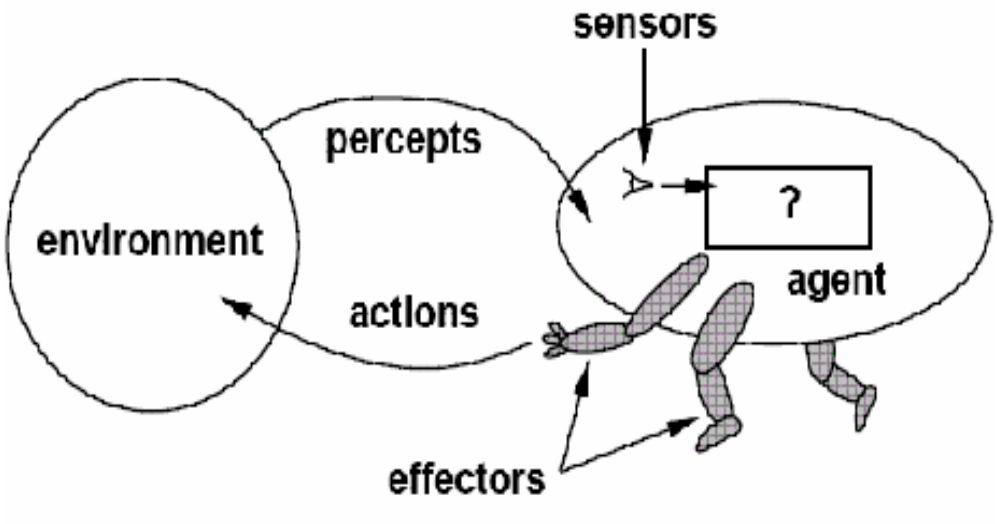
Types of Intelligent Agents

- Software agents:
  - Also called a softbot (software robot)
  - It is an agent that interacts with a software environment by issuing commands and interpreting the environments feedback.
  - E.g. mail handling agent, information filtering agent

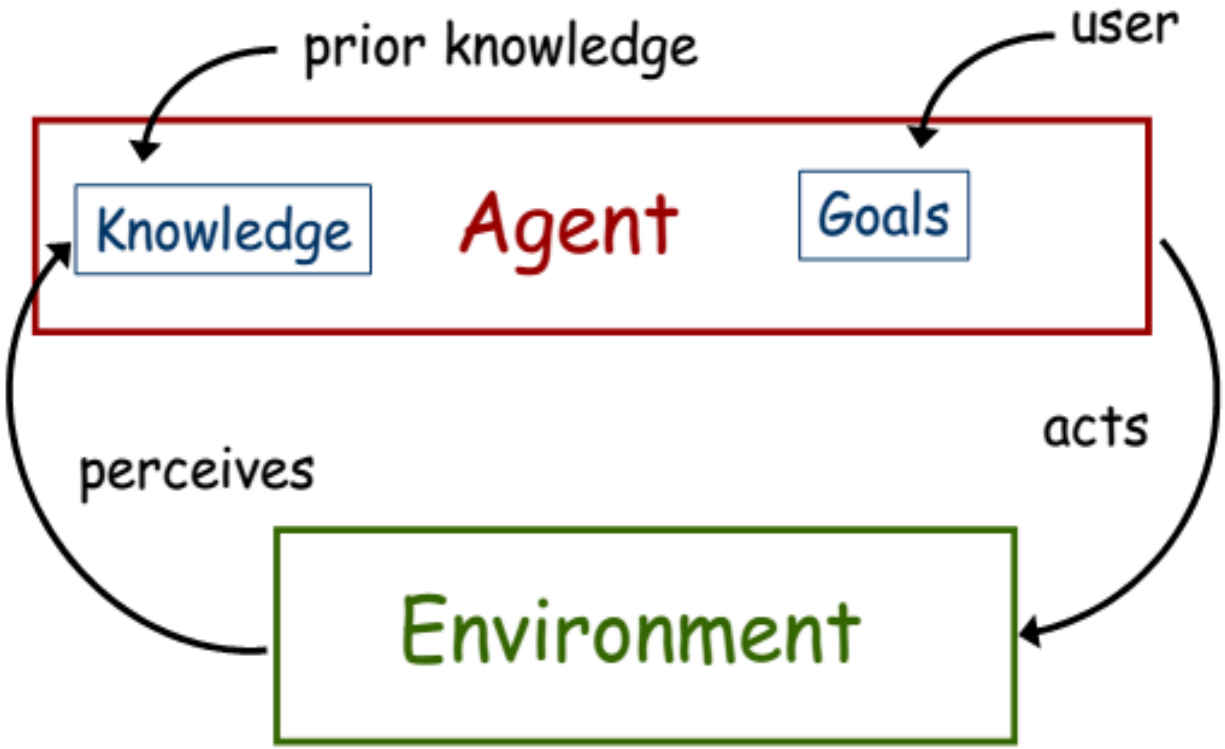
- Physical agents
  - are robots that operates in the physical world and can perceive and manipulate objects in that world

Agent

- Agent is something that perceives/observes its environment through **SENSORS** and acts upon that environment through **EFFECTORS**.
- The agent is assumed to exist in an environment in which it perceives and acts
- An agent is rational/sensible since it does the right thing to achieve the specified goal.



- **Agent = architecture + program**



- Require more flexible interaction with the environment, the ability to modify one's goals, knowledge that be applied flexibly to different situations

Agent

	Human beings	Agents
Sensors	Eyes, Ears, Nose	Cameras, Scanners, Mic, infrared range finders
Effectors	Hands, Legs, Mouth	Various Motors (artificial hand, artificial leg), Speakers, Radio

Examples of agents in different types of applications

Agent type	Percepts	Actions	Goals	Environment
Medical diagnosis system	Symptoms, patient's answers	Questions, tests, treatments	Healthy patients, minimize costs	Patient, hospital
Interactive English tutor	Typed words, questions, suggestions	Write exercises, suggestions, corrections	Maximize student's score on exams	Set of students, materials
Part-picking robot	Pixels of varying intensity	Pick up parts and sort into bins	Place parts in correct bins	Conveyor belts with parts
Satellite image analysis system	Pixels intensity, color	Print a categorization of scene	Correct categorization	Images from orbiting satellite
Refinery controller	Temperature, pressure readings	Open, close valves; adjust temperature	Maximize purity, yield, safety	Refinery

Rationality/Reasonableness vs. Omniscience/Awareness/knowledge

- Rational agent acts so as to achieve one's goals, given one's beliefs (one that does the right thing).
  - What does right thing mean? one that will cause the agent to be most successful and is expected to maximize goal achievement, given the available information
- An Omniscient/all-knowing agent knows the actual outcome of its actions, and can act accordingly, but in reality omniscience is impossible.
- Rational agents take action with expected success, where as omniscient agent take action with 100% sure of its success
- Is human beings Omniscient or Rational agent?

Example

- You are walking along the road to **shewaber**; You see an old friend across the street. There is no traffic.
- So, being rational, you start to cross the street.
- Mean while a big banner falls off from above and before you finish crossing the road, you are flattened.

Were you irrational to cross the street?

- This points out that rationality is concerned with expected success, given what has been perceived.
  - Crossing the street was rational, because most of the time, the crossing would be successful, and there was no way you could have foreseen the falling banner.
  - The EXAMPLE shows that we can not blame an agent for failing to take into account something it could not perceive. Or for failing to take an action that it is incapable of taking.

Rational agent

- In summary what is rational at any given point depends on four things.
  - **Perception/sensitivity**: Everything that the agent has perceived so far concerning the current scenario in the environment
  - **Knowledge**: What an agent already knows about the environment
  - **Action**: The actions that the agent can perform back to the environment
  - **Performance measure**: The performance measure that defines degrees of success of the agent.
- Generally, rationality refers to "doing the right thing".
  - ✓ So that one key property of Intelligent Agents is being a rational agent, because rational agents means doing the right thing.

- Therefore in designing an intelligent agent, one has to remember PEAS (Performance, Environment, Actuators, Sensors) framework.

- **Example 1**: PEAS description for **Automated Taxi Driver.**
  a) Performance Measure
     - ✓ How much it is safe, fast, legal, comfortable trip, maximize profits, etc.

  b) Environment
     - ✓ The roads, pedestrians, customers, other traffic.

  c) Actuators
     - ✓ The steering wheel, accelerator, brake, signal, horn, etc.

  d) Sensors
     - ✓ The cameras, sonar, speedometer, GPS, odometer, engine sensors, keyboard, etc.

- **Example 2**: PEAS description of Part-Sorting Robot
  a) Performance Measure
     - ✓ Percentage of parts in correct bins
  b) Environment
     - ✓ Conveyor belt with parts, bins
  c) Actuators
     - ✓ Robotic arm
  d) Sensors
     - ✓ Camera, joint angle sensors

- **Example 3**: PEAS description of Spam Filter
  a) Performance Measure
     - ✓ Minimizing false positives, false negatives
  b) Environment
     - ✓ A user's email account
  c) Actuators
     - ✓ Mark as spam, delete, etc.
  d) Sensors
     - ✓ Incoming messages, other information about user's account

Performance measure

- How do we decide whether an agent is successful or not?
  - Establish a standard of what it means to be successful in an environment and use it to measure the performance.
  - A rational agent should do whatever action is expected to maximize its performance measure, on the basis of the evidence provided by the percept sequence and whatever built-in knowledge the agent has.

- What is the performance measure for "**crossing the road**"?
- What about "**Chess Playing**"?

Assignment

- Consider the need to design a “taxi driver agent” that serves in Debre markos city;
  - Identify what to perceive, actions to take, the environment it interacts with?
  - Identify **sensors, effectors, goals, environment** and **performance measure** that should be integrated for the agent to be successful in its operation?

Designing an agent/Agent Function

- An **agent function** maps the percept histories to actions.
- An **agent program** is the one that runs on the physical architecture to produce the agent function.
- **Agent = architecture + program**
- Architecture
  - Runs the programs
  - Makes the percept from the sensors available to the programs
  - Feeds the program's action choices to the effectors

- Programs
  - Accepts percept from an environment and generates actions
    - Before designing an agent program, we need to know the possible percept and actions
  - By enabling a learning mechanism, the agent could have a degree of autonomy/independence, such that it can reason and take decision

Types of Environments

1. **Fully Observable (vs. Partially Observable)**
   - The agent's sensors give it access to the complete state of the environment at each point in time.
2. **Deterministic (vs. Stochastic)**
   - The next state of the environment is completely determined by the current state and the agent's action.
3. **Episodic (vs. Sequential)**
   - The agent's experience is divided into atomic "episodes," and the choice of action in each episode depends only on the episode itself.

4. **Static (vs. Dynamic)**
   - The environment is unchanged while an agent is deliberating.
   - **Semi dynamic**:- The environment does not change with the passage of time, but the agent's performance score does.
5. **Discrete (vs. Continuous)**
   - The environment provides a fixed number of distinct percepts, actions, and environment states.
   - Time can also evolve in a discrete or continuous fashion.

6. **Single Agent (vs. Multi-agent)**
   - An agent operating by itself in an environment
7. **Known (vs. Unknown)**
   - The agent knows the rules of the environment

Case Examples for Task Environment Types

- Example 1: Task environment types of three agents: *Chess*, *Taxi with Clock* and *Taxi without Clock*

Fully Observable	Yes	Yes	No
Deterministic	Strategic	Strategic	No
Episodic	No	No	No
Static	Semi	Yes	No
Discrete	Yes	Yes	No
Single Agent	No	No	No

- **Example 2**: Task Environments types of four agents: ***Chess***, ***Poker***, ***Image Analysis*** and ***Butler Robot***

Task Env.	Observable	Agents	Episodic	Static	Discrete
Chess	Fully	Multi	Sequential	static	discrete
Poker	Partially	Multi	Sequential	static	discrete
Image analysis	Fully	Single	Episodic	static	continuous
Butler Robot	Partially	Single	Sequential	dynamic	continuous

Environment Types

Below are lists of properties of a number of familiar environments

Problems	Observable	Deterministic	Episodic	Static	Discrete
Crossword Puzzle	Yes	Yes	No	Yes	Yes
Part-picking/cutting robot	No	No	Yes	No	No
Web shopping program	No	No	No	No	Yes
Tutor	No	No	No	Yes	Yes
Medical Diagnosis	No	No	No	No	No
Taxi driving	No	No	No	No	No

•Hardest case: a environment that is inaccessible, sequential, non-deterministic, dynamic, continuous.

Agents acting in an environment

Hierarchy of Agent Types

- Let's have a closer look on the following five agent categories:-

  I. **Simple Reflex Agents**

  II. **Model-Based Reflex Agents**

  III. **Goal-Based Agents**

  IV. **Utility-Based Agents**

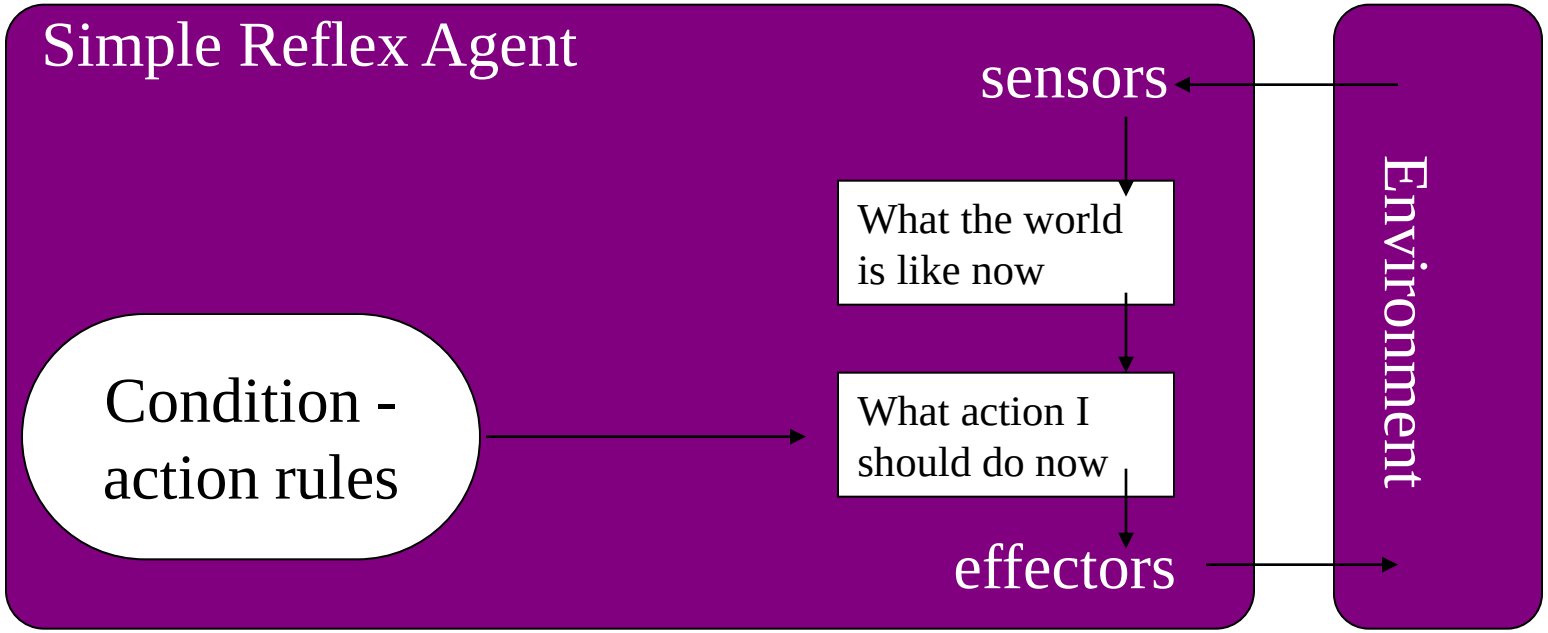
  V. **Learning Agents**

I. Simple Reflex Agents

- Select action on the basis of current percept, ignoring all past percepts
- It uses just ***condition-action rules***
  - ✓ The rules are like the form "if … then …"
  - ✓ Efficient but have narrow range of applicability
  - ✓ Because knowledge sometimes cannot be stated explicitly
  - ✓ Work only
    - ○ If the environment is fully observable

➢ If car-in-front-is breaking then initiate-braking.
➢ Blinking when something approaches the eye.

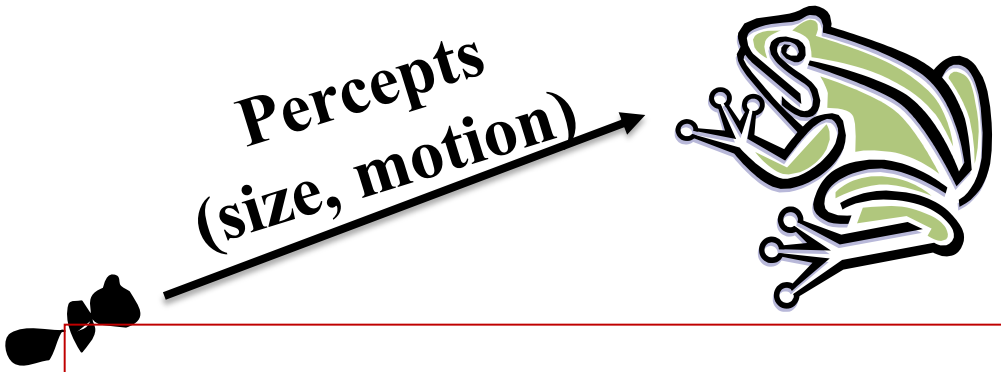
Figure: Structure of a simple reflex agent



function SIMPLE-REFLEX-AGENT(*percept*) **returns** action
  static: *rules*, a set of condition-action rules

  state ← INTERPRET-INPUT (*percept*)
  rule ← RULE-MATCH (*state*,*rules*)
  action ← RULE-ACTION [*rule*]
  return *action*

A Simple Reflex Agent in Nature



<u>RULES</u>

(1) If small moving object, then activate SNAP

(2) If large moving object,
then activate AVOID and inhibit SNAP

ELSE (not moving) then NOOP

Needed for completeness

Action: SNAP or AVOID or NOOP

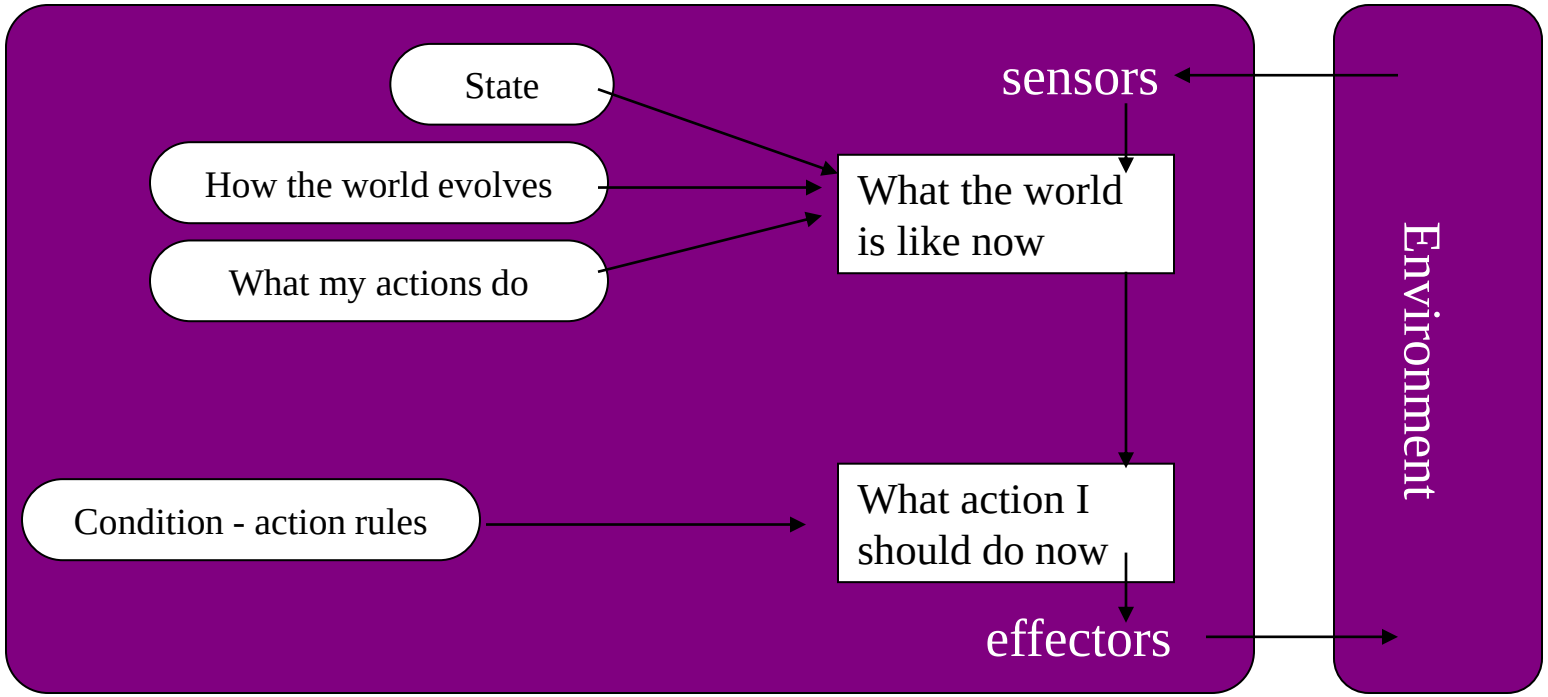
II. Model-Based Reflex Agents

- Maintains internal state that keeps track of aspects of the environment that cannot be currently observed
- For the world that is partially observable
  - **If the car is a recent model -** there is a centrally mounted brake light. With older models, there is no centrally mounted, so what if the agent gets confused?

    Is it a parking light? Is it a brake light? Is it a turn signal light?

    - ✓ The agent has to keep track of an internal state
      - That depends on the percept history
      - Reflecting some of the unobserved aspects
      - Example, driving a car and changing lane
- Requiring two types of knowledge
  - ✓ How the world evolves independently of the agent
  - ✓ How the agent's actions affect the world
  - ✓ The agent is with memory

Figure: Structure of Model-Based reflex agent



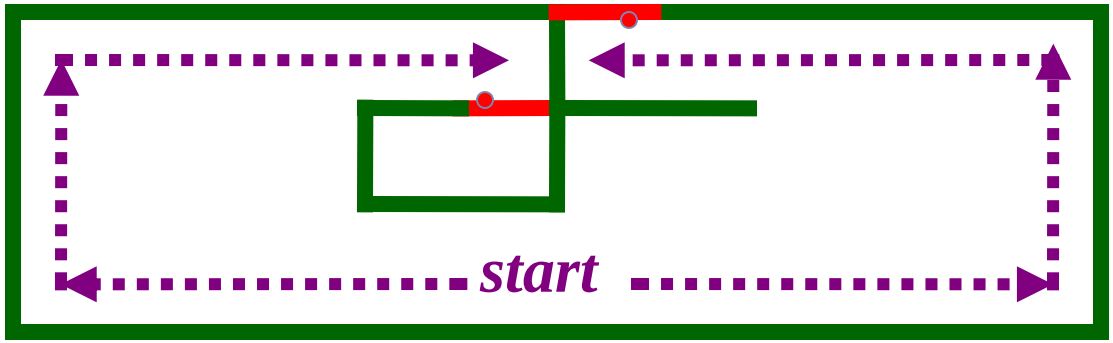
function REFLEX-AGENT-WITH-STATE (*percept*) **returns** action
static: *state*, a description of the current world state
rules, a set of condition-action rules

state ← UPDATE-STATE (*state*, *percept*)
rule ← RULE-MATCH (*state*, *rules*)
action ← RULE-ACTION [*rule*]
state ← UPDATE-STATE (*state*, *action*)
return *action*

Example Table Agent With Internal State

IF	THEN
Saw an object ahead, and turned right, and it's now clear ahead	Go straight
Saw an object Ahead, turned right, and object ahead again	Halt
See no objects ahead	Go straight
See an object ahead	Turn randomly

- Case Example for A Reflex Agent With Internal State
  - ✓ The **Wall-Following**



- The *Actions* are:-
  - ✓ Left
  - ✓ Right
  - ✓ Straight
  - ✓ Open-Door

- The ***Rules*** are:-

1) If open(left) & open(right) and open(straight) then choose randomly between right and left
2) If wall(left) and open(right) and open(straight) then straight
3) If wall(right) and open(left) and open(straight) then straight
4) If wall(right) and open(left) and wall(straight) then left
5) If wall(left) and open(right) and wall(straight) then right
6) If wall(left) and door(right) and wall(straight) then open-door
7) If wall(right) and wall(left) and open(straight) then straight.
8) (Default) Move randomly

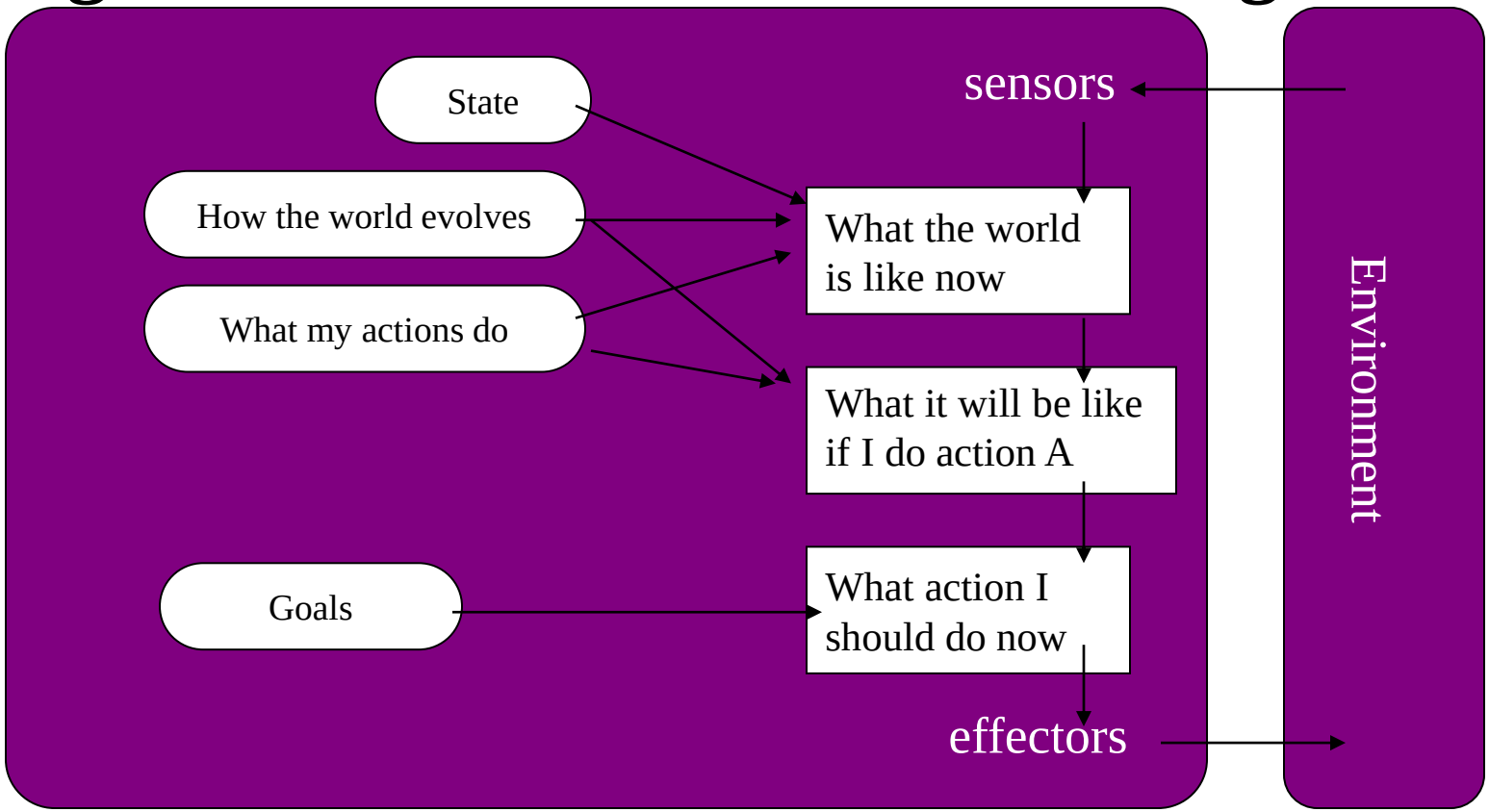
III. Goal-Based Agents

- The agent uses goal information to select between possible actions in the current state
- Current state of the environment is always not enough
- The goal is another issue to achieve
  - ✓ Judgment of rationality / correctness
- Actions chosen ➔ goals, based on
  - ✓ The current state
  - ✓ The current percept

  - E.g. At a road junction, the taxi can turn left, right or go straight.

- Conclusion
  - ✓ Goal-based agents are less efficient
  - ✓ But more flexible
    - Agent ⇐ Different goals ⇐ different tasks
  - ✓ Search and planning
    - Two other sub-fields in AI
    - To find out the action sequences to achieve its goal

Figure: Structure of a Goal-based agent

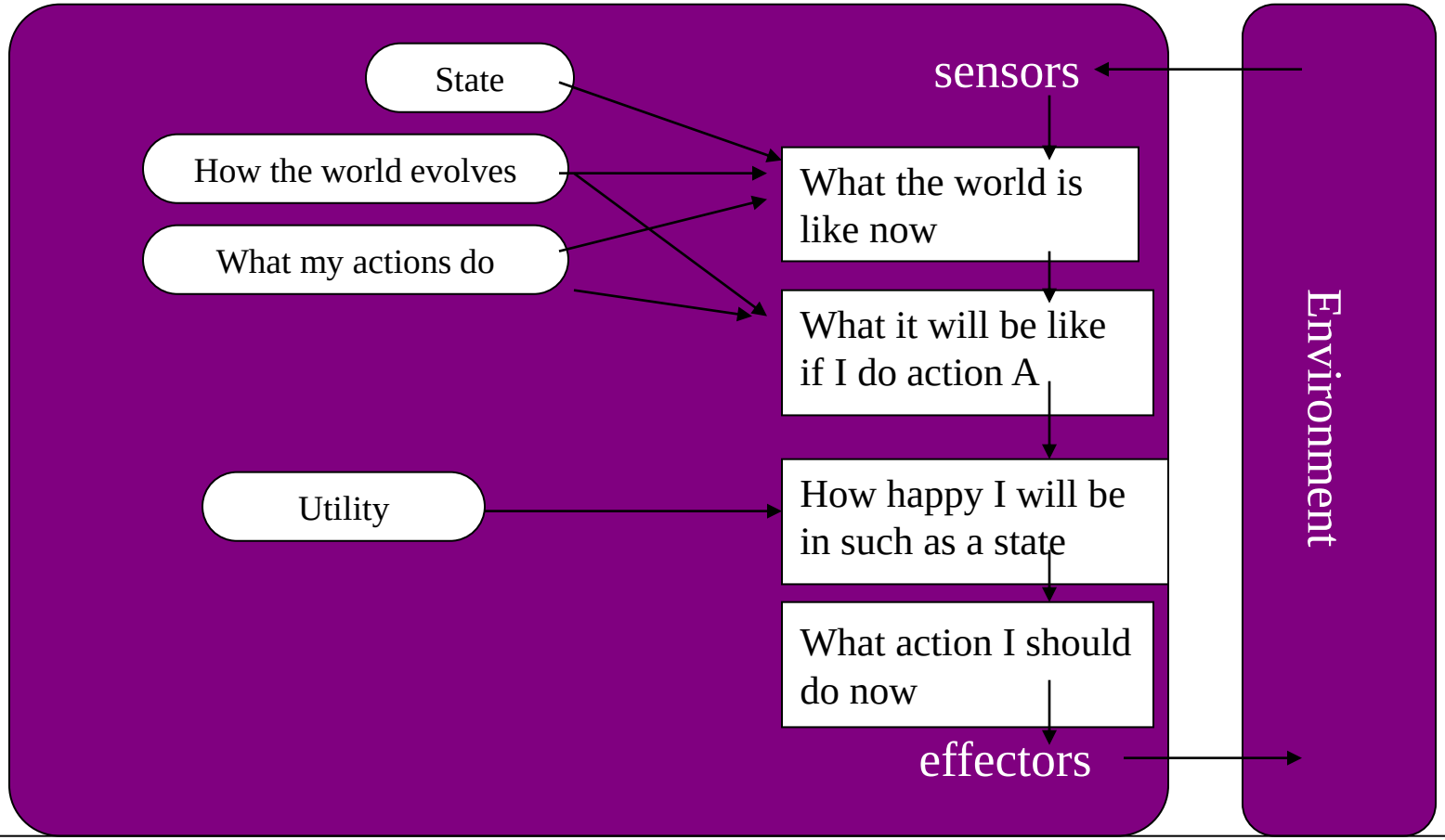


function GOAL_BASED_AGENT (*percept*) **returns** action
  state ← UPDATE-STATE (*state*, *percept*)
  action ← SELECT-ACTION [*state*, *goal*]
  state ← UPDATE-STATE (*state*, *action*)
 return *action*

IV. Utility-Based Agents

- The agent uses a utility function to evaluate the desirability of states that could result from each possible action
- Goals alone are not enough
  - ✓ To generate **high-quality** behavior
  - ✓ Example, meals in Canteen, good or not ?
- Many action sequences ➔ the goals
  - ✓ Some are better and some worse
  - ✓ If goal means success,
  - ✓ Then **utility** means the degree of success (how successful it is).
- E.g. route recommendation system
  - There are many action sequences that will get the taxi to its destination, thereby achieving the goal.
  - Some are quicker, safer, more reliable, or cheaper than others. We need to consider Speed and safety

Figure: Structure of a utility-based agent



function UTILITY_BASED_AGENT (*percept*) **returns** action

state ← UPDATE-STATE (*state*, *percept*)
action ← SELECT-OPTIMAL_ACTION [*state*, *goal*]
state ← UPDATE-STATE (*state*, *action*)
return *action*

- It is said state A has higher utility
  - ✓ If state A is more preferred than others
- Utility is therefore a function
  - ✓ That maps a state onto a real number
  - ✓ The degree of success

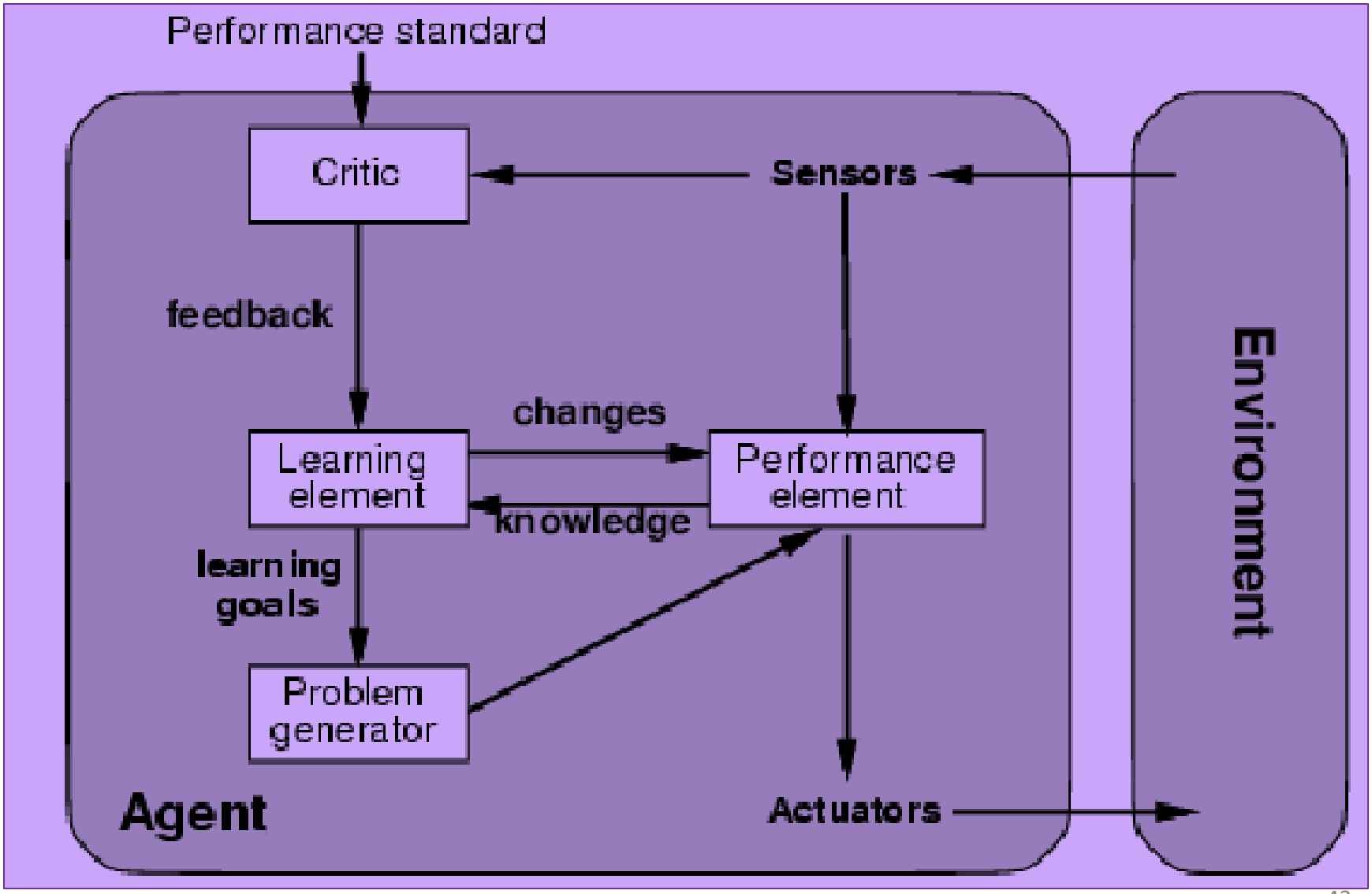
- Utility has several advantages:
  - ✓ When there are conflicting goals,
    - Only some of the goals but not all can be achieved
    - utility describes the appropriate trade-off
  - ✓ When there are several goals
    - None of them are achieved **<u>certainly</u>**
    - Utility provides a way for the decision-making

V. Learning Agents

- To build learning machines and then to teach them.
- After an agent is programmed, can it work immediately?
  - ✓ No, it still need teaching
- In AI,
  - ✓ Once an agent is done
  - ✓ We teach it by giving it a set of examples
  - ✓ Test it by using another set of examples
- We then say the agent learns
  - ✓ A learning agent
  - learning agents are able to perform tasks , analyze performance and look for new ways to improve on those tasks

- Four conceptual components
  - ***a) Learning Element***
    - ✓ Which is responsible for making improvement
  - ***b) Performance Element***
    - ✓ Which is responsible for selecting external actions
  - ***c) Critic***
    - ✓ The learning element uses feedback from the **critic** on how the agent is doing and determines how the performance element should be modified to do better in the future.
    - ✓ Tells the Learning element how well the agent is doing with respect to fixed performance standard.
      - ○ Feedback from user or examples, good or not?
  - ***d) Problem Generator***
    - ✓ Responsible for suggesting actions that will lead to new and informative experiences

Figure: Structure of a Learning agent



Summery hierarchy in types of agents

- **Reflex agents:**

✓ These agents function in a current state, ignoring past history.

✓ Responses are based on the event-condition-action rule (ECA rule) where a user initiates an event and the agent refers to a list of pre-set rules and pre-programmed outcomes.

- **Model-based agents:**

✓ These agents choose an action in the same way as a reflex agent, but they have a more comprehensive view of the environment.

✓ A model of the world is programmed into the internal system that incorporates the agent's history.

- **Goal-based agents:**

✓ These agents expand upon the information model-based agents store by also including goal information, or information about desirable situations.

- **Utility-based agents:**

✓ These agents are similar to goal-based agents but provide an extra utility measurement which rates each possible scenario on its desired result and chooses the action that maximizes the outcome.

✓ Rating criteria examples could be the probability of success or the resources required.

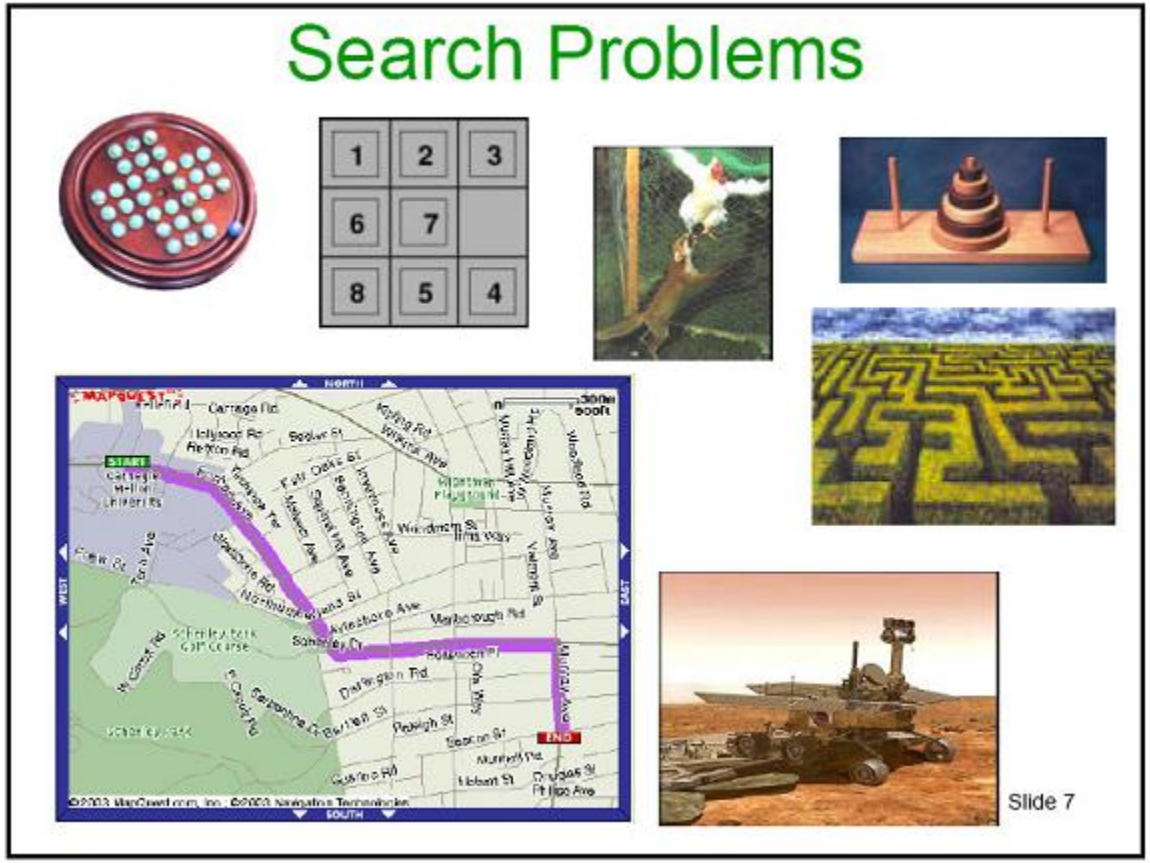
- **Learning agents:**

✓ These agents have the ability to gradually improve and become more knowledgeable about an environment over time through an additional learning element.

✓ The learning element will use feedback to determine how performance elements should be changed to improve gradually.

CHAPTER : THREE
Problem solving

Can solve many problems with lots of computing power that humans are not good



Problem

- It is a gap between what actually is and what is desired.
  - A problem exists when an individual becomes aware of the existence of an obstacle which makes it difficult to achieve a desired goal or objective.

- A number of problems are addressed in AI, both:
  - **Toy problems**: are problems that are useful to test and demonstrate methodologies.
    - Can be used by researchers to compare the performance of different algorithms
    - e.g. 8-puzzle, n-queens, vacuum cleaner world, towers of Hanoi, …

  - **Real-life problems:** are problems that have much greater commercial/economic impact if solved.
    - Such problems are more difficult and complex to solve, and there is no single agreed-upon description
    - E.g. Route finding, Traveling sales person, etc.

Solving a problem

Formalize the problem: Identify the collection of information that the agent will use to decide what to do.

- Define states
  - States describe distinguishable stages during the problem-solving process
  - Example- What are the various states in **route finding** problem?
    - The various places including the location of the agent

- Define the available operators/rules for getting from one state to the next
  - Operators cause an action that brings transitions from one state to another by applying on a current state

- Suggest a suitable representation for the **problem space/state space**
  - Graph, table, list, set, … or a combination of them

State space of the problem

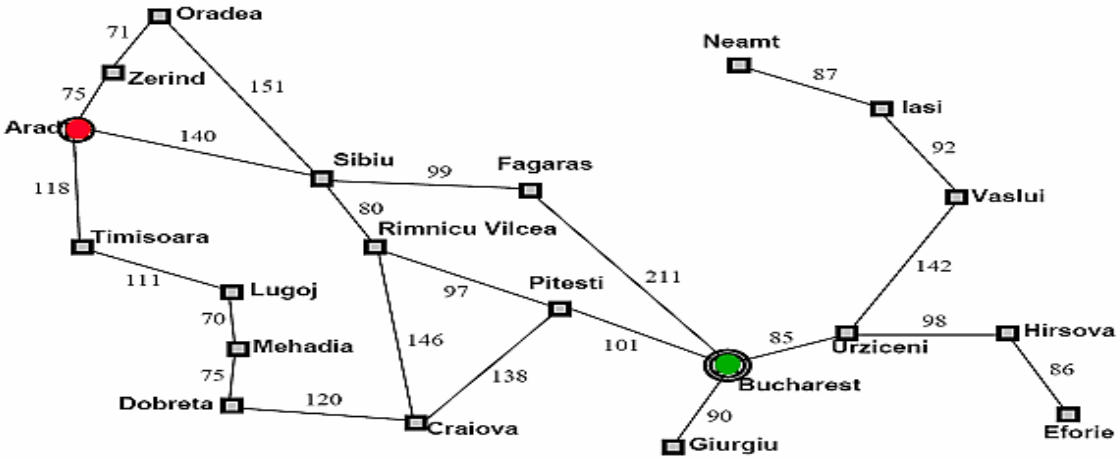
- The **state space** defines the set of all relevant states reachable from the initial state by (any) sequence of actions through iterative application of all permutations and combinations of operators
- **State space** (also called **search space/problem space)** of the problem includes the various states
  - Initial state: defines where the agent starts or begins its task
  - Goal state: describes the situation the agent attempts to achieve
  - Transition states: other states in between initial and goal states

Exercise– Find the state **space** for route finding problem where the agent wants to go from **polle** to **shewaber**.

- Think of the states reachable from the initial state until we reach to the goal state.

Example 1: Romania Travel.

Currently in Arad, need to get to Bucharest by tomorrow to catch a flight. What is the State Space?



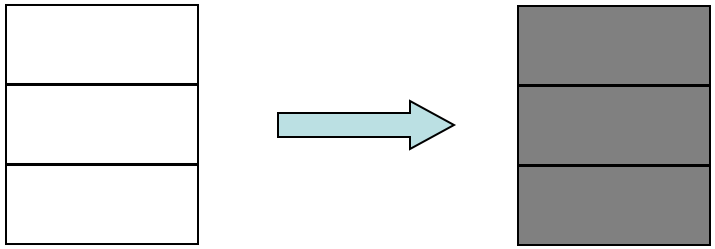
- **State space.**
  - **States**: the various cities you could be located in.
  - **Actions**: drive between neighboring cities.
  - **Initial state**: in Arad
  - **Desired condition (Goal):** be in a state where you are in Bucharest.

- Once the problem has been formulated as a state space search, various algorithms can be utilized to solve the problem

- Solution will be the route, the sequence of cities to travel through to get to Bucharest.

Example 2: Coloring problem

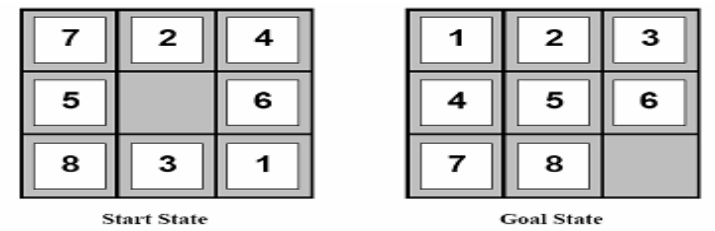
– There are 3 rectangles. Both are initially white. The problem is to change all rectangles with white color to black color. Color change is one rectangle at a time. Define state space for coloring problem?



Example 3: The 8 puzzle problem

- Arrange the tiles so that all the tiles are in the correct positions. You do this by moving tiles or space. You can move a tile/space up, down, left, or right, so long as the following conditions are met:
  A) there's no other tile blocking you in the direction of the movement; and

  B) you're not trying to move outside of the boundaries/edges.



▪**State space.**

• **States**: The different configurations of the tiles.

• **Actions**: Moving the blank up, down, left, right. Can every action be performed in every state?

• **Initial state**: as shown in the 1st picture.

• **Desired condition (Goal):** be in a state where the tiles are all in the positions shown in the 2nd picture

▪Solution will be a sequence of moves of the blank that transform the initial state to a goal state.

Knowledge and types of problems

- There are **four types** of problems:
  - Single state problems (Type 1)
  - Multiple state problems (Type 2)
  - Exploration problems (Type 3)
  - Contingency Problems (Type 4)

- This classification is based on the level of knowledge that an agent can have concerning its action and the state of the world

- Thus, the first step in formulating a problem for an agent is to see what knowledge it has concerning
  - The effects of its action on the environment
  - Accessibility of the state of the world

- This knowledge depends on how it is connected to the environment via its percepts and actions

Single state problem

- **Fully observable:** The world is accessible to the agent
  - It can determine its exact state through its sensors
  - The agent’s sensor knows which state it is in
- **Deterministic:** The agent knows exactly the effect of its actions
  - It can then calculate exactly which state it will be in after any sequence of actions
- Action sequence is completely planned

Multiple state problems

- **Partially observable:** The agent has limited access to the world state
  - It might not have sensors to get full access to the environment states or as an extreme, it can have no sensors at all (due to lack of percepts)
- **Deterministic:** The agent knows exactly what each of its actions do
  - It can then calculate which state it will be in after any sequence of actions
- If the agent has full knowledge of how its actions change the world, but does not know of the state of the world, it can still solve the task

Contingency Problems

- **Partially observable:** The agent has limited access to the world state
- **Non-deterministic:** The agent is ignorant of the effect of its actions
- Sometimes ignorance prevents the agent from finding a guaranteed solution sequence.
- Many problems in the real world are contingency problems (exact prediction is impossible)
    - For this reason many people keep their eyes open while walking around or driving.
- Suppose the agent is in **Murphy's law world**
  - The agent has to sense during the execution phase, since things might have changed while it was carrying out an action. This implies that
    - the agent has to compute a tree of actions, rather than a linear sequence of action.

Exploration problem

- The agent has no knowledge of the environment
  - **World Partially observable** : No knowledge of states (environment)
    - Unknown state space (no map, no sensor)
  - **Non-deterministic**: No knowledge of the effects of its actions
  - Problem faced by (intelligent) agents (like, **new born babies**)

- This is a kind of problem in the real world rather than in a model, which may involve significant danger for an ignorant agent. If the agent survives, it learns about the environment

- The agent must experiment, learn and build the model of the environment through its results, gradually, discovering
  - What sort of states exist and What its action do
  - Then it can use these to solve subsequent (future) problems

Well-defined problems and solutions

To define a problem, we need the following elements: states, operators, goal test function and cost function.

- **The Initial state:** is the state that the agent starts in or begins with.
  - **Example-** the initial state for each of the following:
    - Coloring problem
      - All rectangles white
    - Route finding problem
      - The point where the agent start its journey (SidistKilo?)
    - 8-puzzle problem ??

Operators

- The set of possible actions available to the agent, i.e.
  - –which state(s) will be reached by carrying out the action in a particular state
  - –**A Successor function S(x)**
    - Is a function that returns the set of states that are reachable from a single state by **any** single action/operator
    - Given state x, S(x) returns the set of states reachable from x by any single action

- Example:
  - –Coloring problem: Paint with black color

    paint (color w, color b)

  - –Route finding problem: Drive through cities/places

    drive (place x, place y)

  - – 8-puzzle ??

Goal test function

- The agent execute to determine if it has reached the goal state or not
  - Is a function which determines if the state is a goal state or not

- Example:
  - Route finding problem: Reach Addis Ababa airport on time

    → IsGoal(x, Addis_Ababa)

  - Coloring problem: All rectangles black

    → IsGoal(rectangle[], n)

  - 8-puzzle ??

Path cost function

- A function that assigns a cost to a path (sequence of actions).
  - Is often denoted by **g.** Usually, it is the sum of the costs of the individual actions along the path (from one state to another state)
  - Measure path cost of a sequence of actions in the state space. For example, we may prefer paths with fewer or less costly actions

- Example:
  - Route finding problem:
    - Path cost from initial to goal state
  - Coloring problem:
    - One for each transition from state to state till goal state is reached
  - 8-puzzle?

Steps in problem solving

- **Goal formulation**
  - is a step that specifies exactly what the agent is trying to achieve
  - This step narrows down the **scope** that the agent has to look at

- **Problem formulation**
  - is a step that puts down the **actions** and **states** that the agent has to consider **given a goal** (avoiding any redundant states), like:
    - the initial state
    - the allowable actions etc…

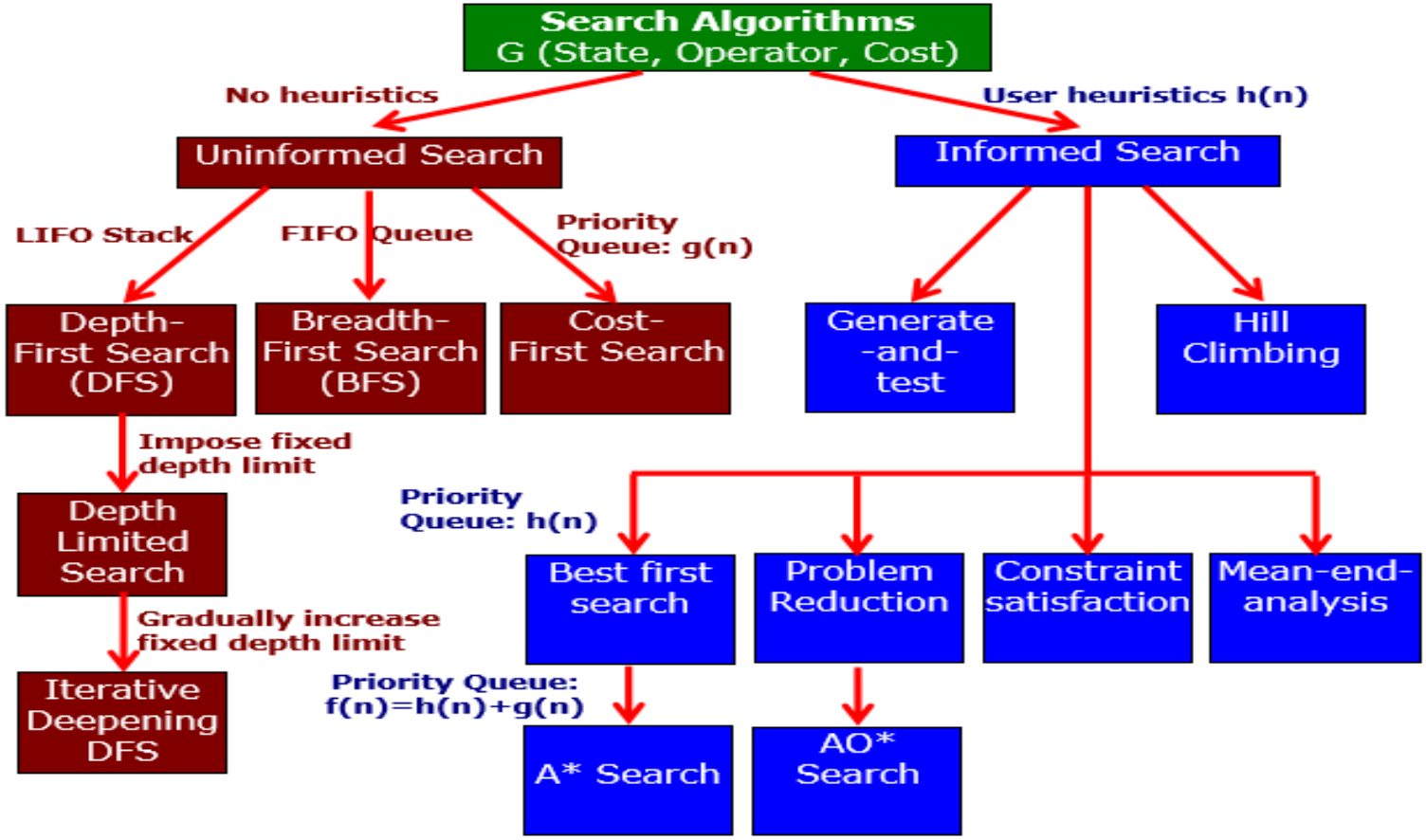
- **Search**
  - is the process of looking for the **various sequence of actions** that lead to a goal state, evaluating them and choosing the **optimal** sequence.

- **Execute**
  - is the final step that the agent executes the chosen sequence of actions to get it to the solution/goal

Searching Algorithms

- **<u>Uniformed</u>**: also called blind, exhaustive, or brute force search. Uses no information about the problem to guide the search, *may not efficient*
  - *These strategies do not take into account any domain specific information about the particular search problem.*
  - Popular uninformed search techniques*: Breadth-First, Uniform-Cost, Depth-First, Depth-Limited, and Iterative Deepening search*
- **<u>Informed</u>**: also called heuristic or intelligent search, uses information about the problem to guide the search, usually guesses the distance to a goal state and *therefore efficient*, but the search may not be always possible

Searching Algorithms



Searching

- Examine different possible sequences of actions & states, and come up with **specific sequence** of operators/actions that will take you from the initial state to the goal state

  –*Given a state space with **initial state** and **goal state,** find* optimal ***sequence of actions** leading through a **sequence of states** to the **final goal state***

- **Searching in State Space**

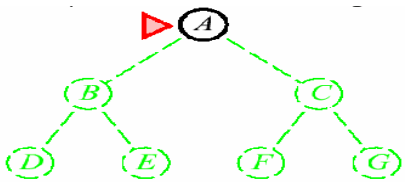
  – *Choose min-cost/max-profit node/state in the state space,*

  – *Test the state to know whether we reached to the goal state or not,*

  – *If not, expand the node further to identify its successor.*

Search Tree

- A search tree is a representation
  - in which *nodes denote paths* and *branches connect paths*.
  - The node with no parent is the **root node**.
  - The nodes with no children are called **leaf**



- *Two functions* needed for conducting search
  - ***Generator (or successors) function****:* Given a state and action, produces its successor states (in a state space)
  - ***Tester (or IsGoal) function***: Tells whether given state S is a goal state

    IsGoal(S) → True/False
  - IsGoal and Successors functions depend on problem domain.

Search algorithm

- ***Input***: a given problem (Initial State + Goal state + transit states and operators)
- ***Output***: returns optimal sequence of actions to reach the goal.
- **Algorithm Evaluation:**
  - *Completeness*: will the search always find a solution if a solution exists ?
  - *Optimality*: will the search always find the *least cost solution*? (when actions have costs)
  - *Time complexity*: how long does it take to find a solution ?
  - *Space complexity*: what is the maximum space used by the algorithm ?
    - *Fast programs can be written But they use up too much memory*
    - *Memory efficient programs can be written But they are slow*

Search strategy

- ***<u>Uninformed (blind) search</u>***
  - They do not need domain knowledge that guide them to the right direction towards the goal
  - Have no information about *the number of steps* or *the path cost* from the current state to the goal
  - It is important for problems for which *there is no additional information* to consider
  - They work fine with *small number of possible states*

Cont.

- ***<u>Informed (heuristic) search</u>***
  - Have problem-specific knowledge (knowledge that is true from experience)
  - Have knowledge about how far are the various state from the goal
  - *Can find solutions more efficiently than uninformed search*

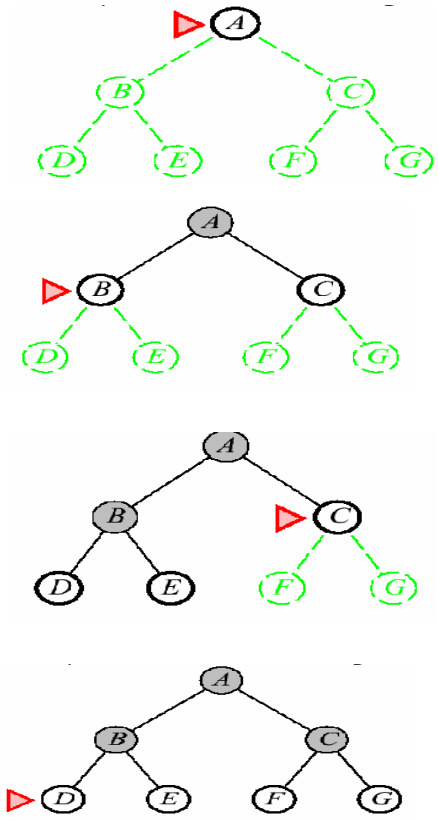
Cont.

- **Uninformed search**
  - Breadth first
  - Depth first
  - Uniform cost, …
  - Depth limited search
  - Iterative deepening
  - etc.

- **Informed search**
  - Greedy search
  - A*-search
  - Iterative improvement,
  - Constraint satisfaction
  - etc.

Uninformed search-*Breadth first search*

- *Expand all nodes on a given level* of the search tree before moving to the next level



- **Implementation**: use **FIFO queue** data structure to store the list:

  – *New successors at the end of queue*

  – *Pop nodes from the front of the queue*

- **Properties**:

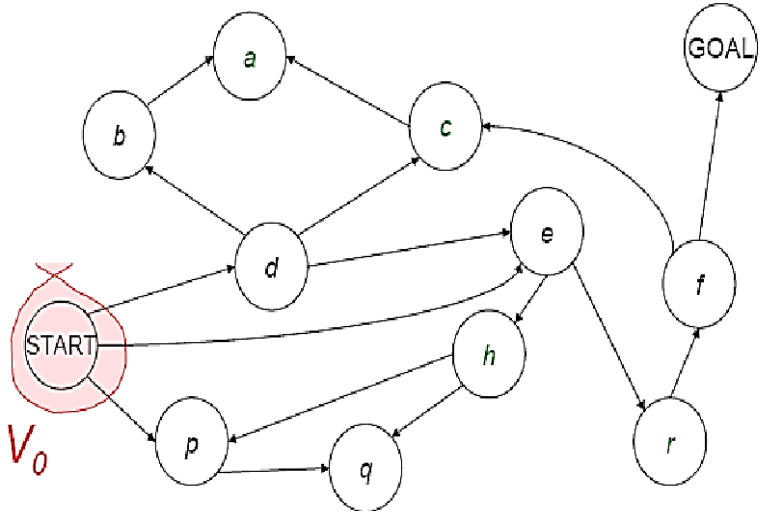
  –Takes space: keeps every node in memory

  –Optimal and complete: guarantees to find solution

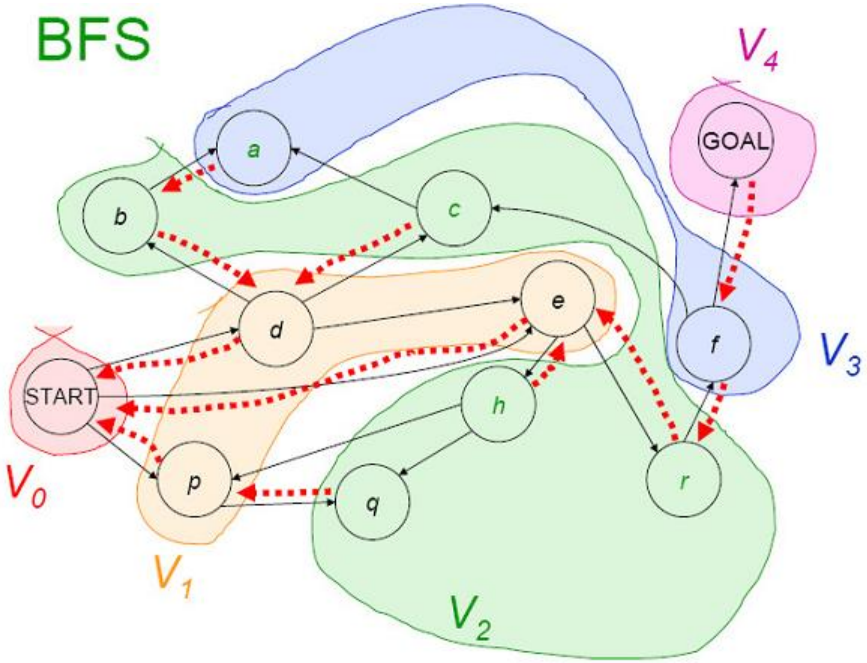
Example

Visiting a vertex – Going to a particular vertex

Exploration of Vertex - Visiting all adjacent vertexes



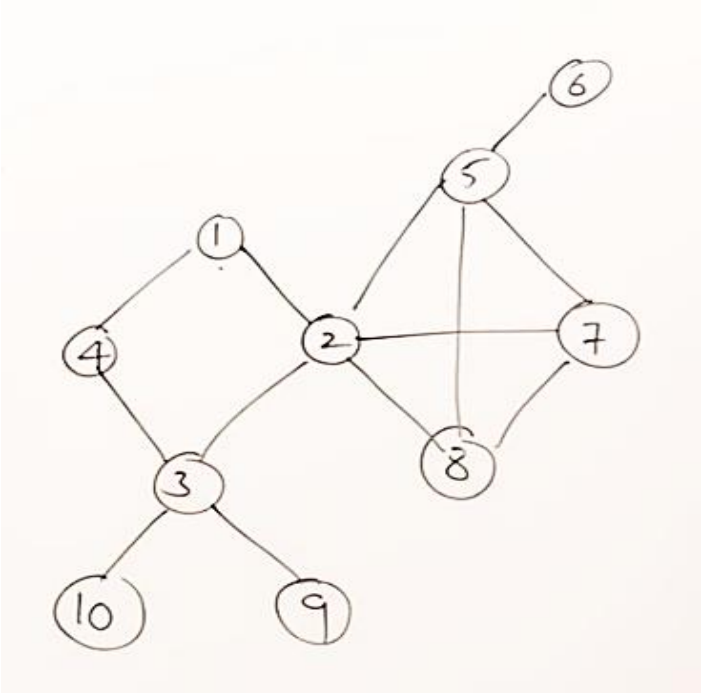
Cont.



Solution is:

Start => **e** => **r** => **f** => **Goal**

Exercise



- **1** is the starting point and **6** is the goal
- *Identify the possible path to reach the goal using BFS*

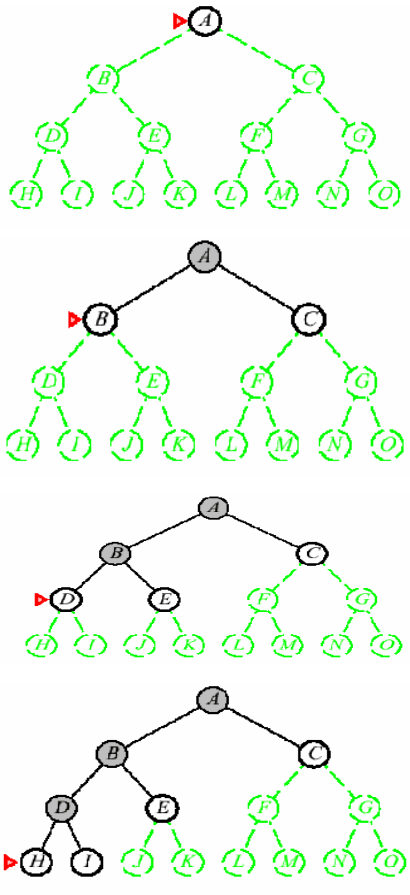
Cont.

- If ***branching factor*** (average number of child nodes for a given node) = **b** and ***depth*** = **d**, then number of nodes at level d **=b^d**
- The total no of nodes created in worst case is $b + b^2 + b^3 + \ldots\ldots + b^d$

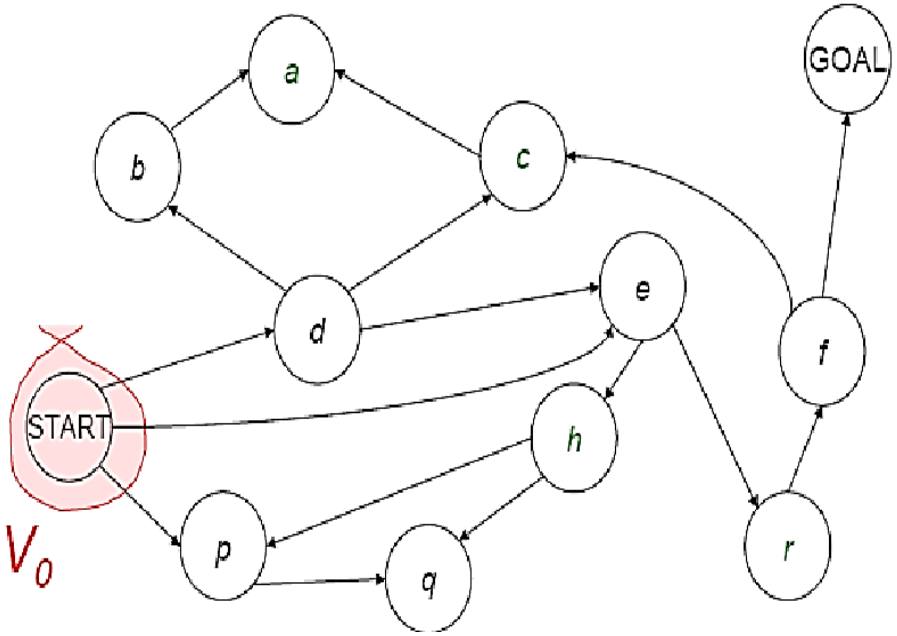
- *Disadvantage:* Since each level of nodes is saved for creating next one, it consumes a lot of memory space

Uninformed search-*Depth-first search*

- *Expand one of the node at the deepest level* of the tree.
  - Only when the search hits a non-goal dead end does the search go back and expand nodes at shallower levels
- **Implementation**: treat the list as **LIFO Stack**
  - *New successors at the top of stack*
  - *Pop nodes from the top of the stack*
- **Properties**
  - Incomplete and not optimal: fails in infinite-depth spaces, spaces with loops.
  - Takes less space (Linear): Only needs to remember up to the depth expanded

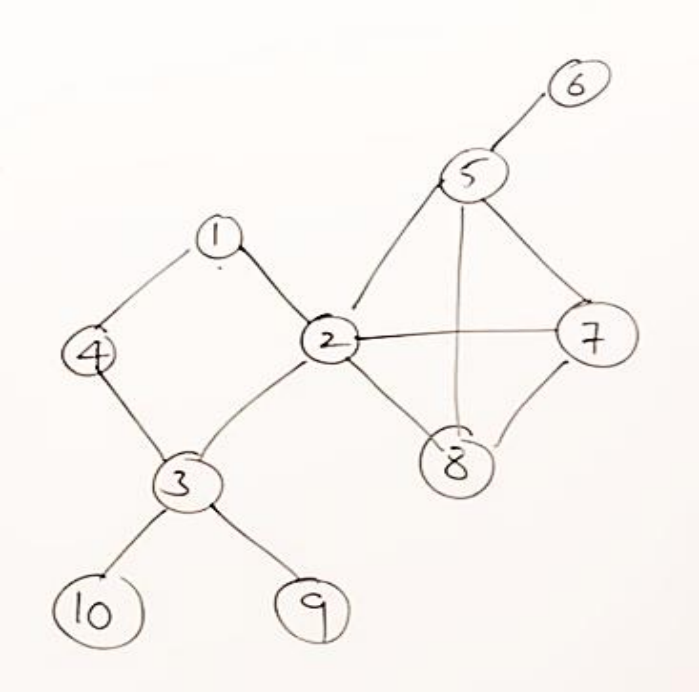


Example



Solution is:
Start => **e** => **r** => **f** => **Goal**

Exercise



- **1** is the starting point and **6** is the goal
- *Identify the possible path to reach the goal using DFS*

Cont.

- *Disadvantage:* This algorithm may not terminate and go on infinitely on one path
  - *The solution to this is to choose a cut-off depth*

Uniformed Searching strategy

BFS

BFS Stands for "**Breadth First Search**".

BFS starts traversal from the root node and then explore the search in the level by level manner i.e. as close as possible from the root node.

Breadth First Search can be done with the help of **queue** i.e. **FIFO** implementation.

This algorithm works in single stage. The visited vertices are removed from the queue and then displayed at once.

BFS is **slower** than DFS.

BFS requires **more** memory compare to DFS.

Applications of BFS

> To find Shortest path
> Single Source & All pairs shortest paths
> In Spanning tree
> In Connectivity

BFS is useful in finding shortest path.BFS can be used to find the shortest distance between some starting node and the remaining nodes of the graph.

DFS

DFS stands for "**Depth First Search**".

DFS starts the traversal from the root node and explore the search as far as possible from the root node i.e. depth wise.

Depth First Search can be done with the help of **Stack** i.e. **LIFO** implementations.

This algorithm works in two stages – in the first stage the visited vertices are pushed onto the stack and later on when there is no vertex further to visit those are popped-off.

DFS is more **faster** than BFS.

DFS require **less** memory compare to BFS.

Applications of DFS

> Useful in Cycle detection
> In Connectivity testing
> Finding a path between V and W in the graph.
> useful in finding spanning trees & forest.

DFS in not so useful in finding shortest path. It is used to perform a traversal of a general graph and the idea of DFS is to make a path as long as possible, and then go back (**backtrack**) to add branches also as long as possible.

Difference between BFS and DFS | BFS vs. DFS

Example :

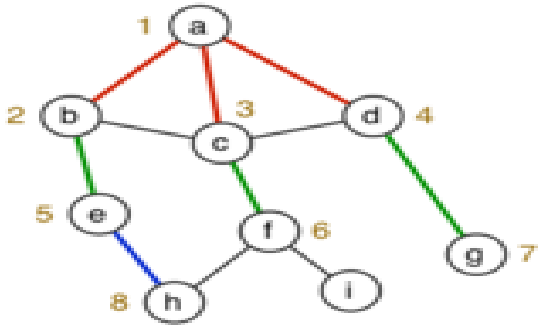
```
      A
     / \
    B   C
   /   / \
  D   E   F
```

A, B, C, D, E, F

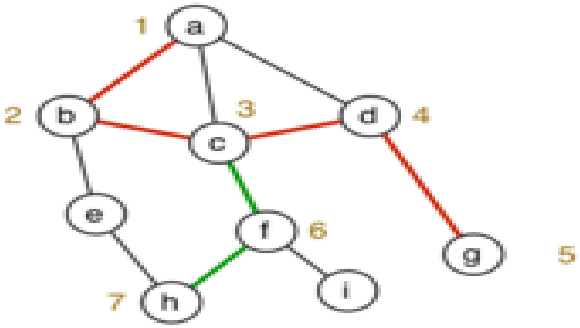
Example :

```
      A
     / \
    B   C
   /   / \
  D   E   F
```

A, B, D, C, E, F

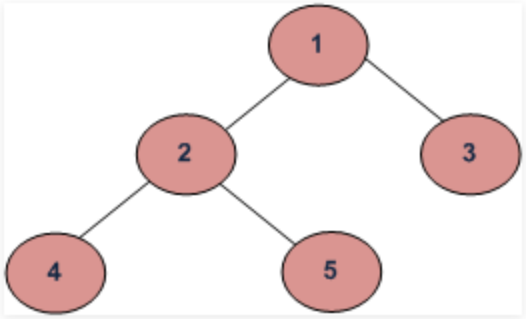


Breadth-first Search



Depth-first Search

- **BFS vs DFS for Binary Tree**
- **What are BFS and DFS for Binary Tree?**
  A Tree is typically traversed in two ways:
- Breadth First Traversal (Or Level Order Traversal)
- Depth First Traversal
  - In order Traversal (Left-Root-Right)
  - Preorder Traversal (Root-Left-Right)
  - Post order Traversal (Left-Right-Root)



- BFS and DFSs of above Tree
- Breadth First Traversal : 1 2 3 4 5
- Depth First Traversals:
- In order Traversal : 4 2 5 1 3
- Preorder Traversal : 1 2 4 5 3
- Post order Traversal : 4 5 2 3 1

Discussion Point

- Compare breadth-first and depth-first search.
  - When would breadth-first be preferable?
  - When would depth-first be preferable?

- Breadth-First:
  - *When completeness is important.*
  - *When optimal solutions are important.*
- Depth-First:
  - *When solutions are dense and low-cost is important, especially space costs*.

Assignment - 1

- Read and discuss each of the following searching algorithms using an example clearly.
  - ***Uniform-Cost Search***
  - ***Iterative Deepening Search (IDS)***
  - ***Bidirectional Search***
  - ***Depth limited search***

Informed search

- In uninformed search, we don't try to evaluate which of the nodes on the frontier/limit are most promising.

- Search efficiency would improve greatly if there is a way to order the choices so that the most promising are explored first.

- This requires domain knowledge of the problem (i.e. heuristic) to undertake focused search.

Cont.

- A **Heuristic** is an operationally-effective piece of information on how to direct search in a problem space.
- Heuristics are only approximately correct. Their purpose is to minimize search on average.
  - It is an estimate of how close we are to a goal state
  - **h(n)** = guesses the cost of getting to the goal from node n
- There are different ways of guessing this cost in different domains. I.e., heuristics are domain specific.

Cont.

- Search algorithms are admissible with respect to a search method if it guarantees finding the **optimal solution first**, even when its value is only an estimate.
  - The algorithm always terminates in an optimal path from the initial state to goal node if one exists.

Informed search- Best-first search

- It is a generic name to the class of informed methods
- The two best first approaches to find the shortest path:
  - **Greedy Search**
  - **A*Search**
- When expanding a node *n* in the search tree, *greedy search uses the estimated cost to get from the current state to the goal state*, define as ***h(n)*.**
- And A*Search uses ***h(n)*** and the sum of the cost to reach that node from the start state, define as ***g(n)*.**

Cont.

- For each node in the search tree, an evaluation function *f(n)* can be defined as the sum of these functions.

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

The heuristic value of node n

The actual cost of node n (from the root to n)

The estimated cost of achieving the goal (from node n to the goal)

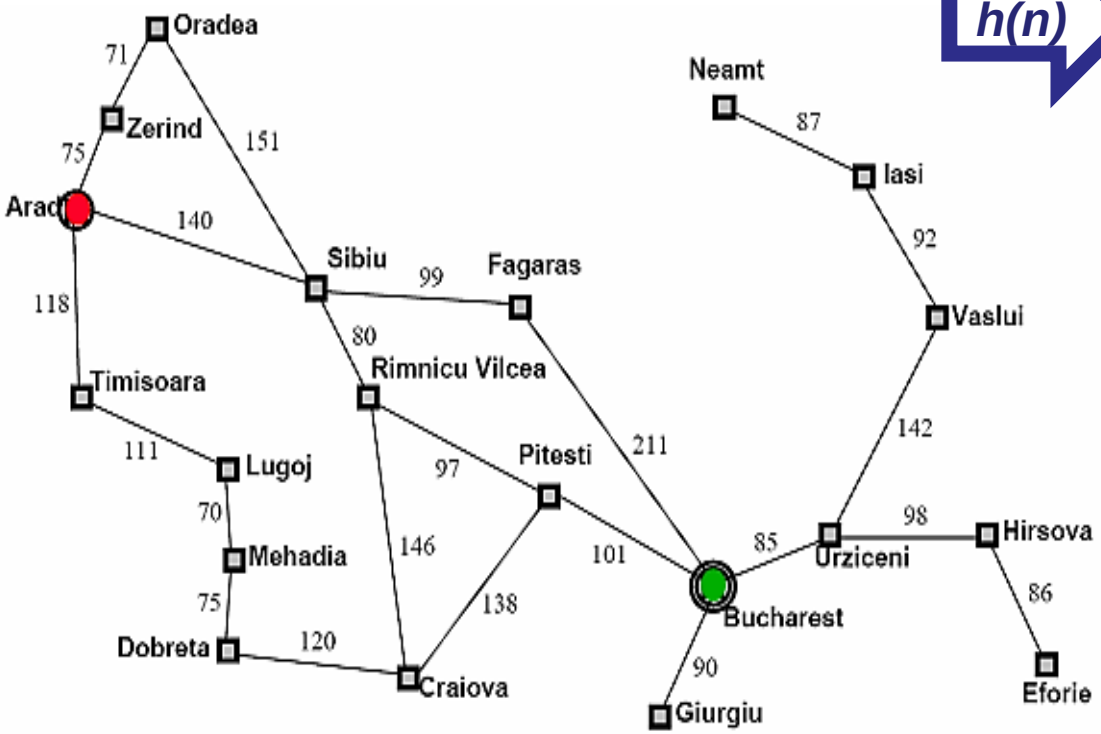
Informed search- *Greedy best-first search*

- Is a best first search that uses a heuristic function h(n) alone to guide the search
  - Selects node to expand that is closest to a goal node (hence it's greedy)
  - The algorithm doesn't take minimum path costs from initial to current nodes into account
  - Always expand node with lowest h-value.

- **Implementation**:
  - expand 1st the node closest to the goal state, i.e. with evaluation function $f(n) = h(n)$
    - $h(n) = 0$ if node n is the goal state
    - Otherwise if $h(n) \geq 0$; continue with an estimated cost of the cheapest path from node n to a goal state

Cont.

- **E.g.**

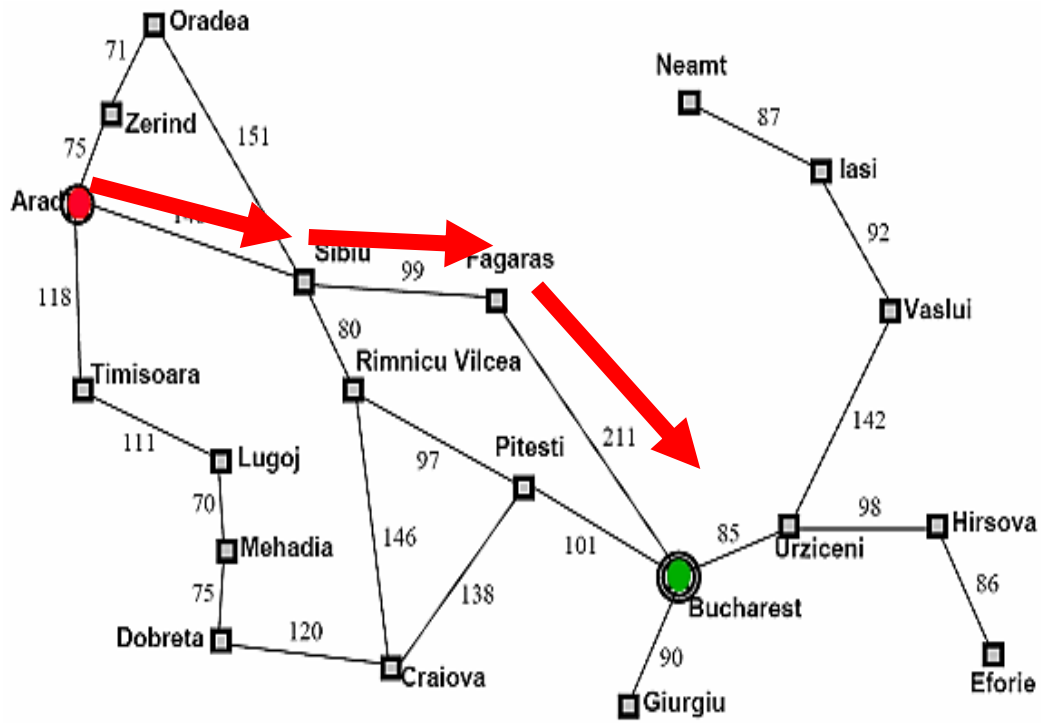


h(n)

Straight–line distance to Bucharest	
Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	178
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	98
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

Cont.

- **Solution**
  - We use h(n) to rank the nodes on the frontier.

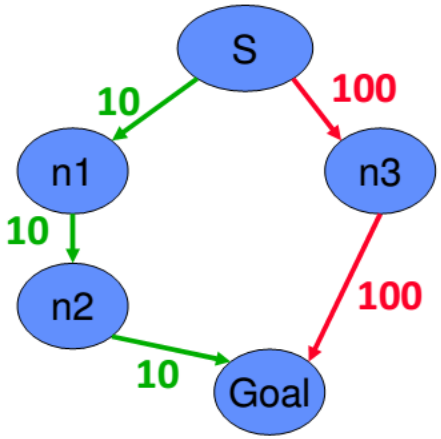


Straight−line distance to Bucharest	
Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	178
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	98
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

Cont.

- However, this method **ignores the cost of getting to n**, so it can be lead astray exploring nodes that cost a lot to get to but seem to be close to the goal:

- **E.g.**



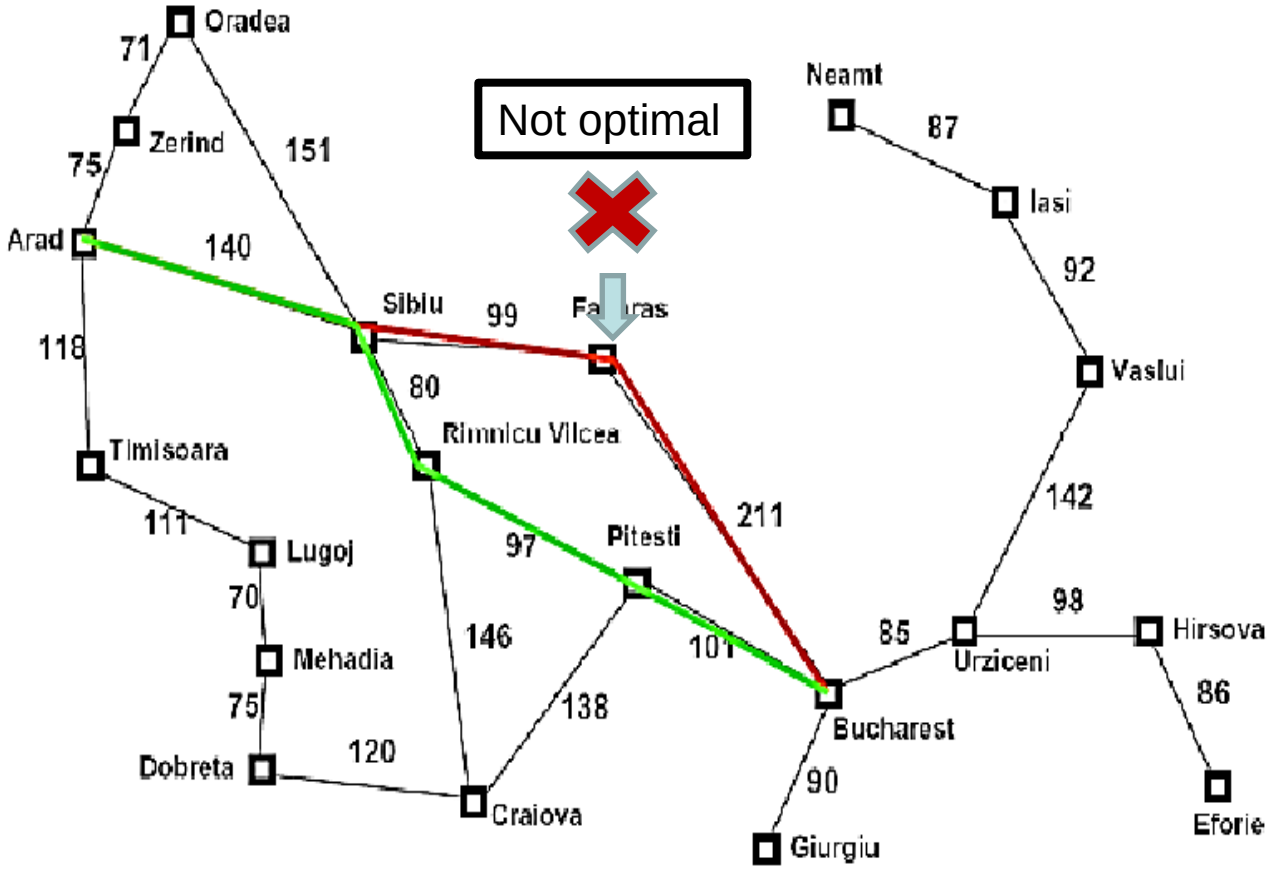
h(n1) = 70
h(n3) = 50

Solution
S => n3 => Goal

Not optimal

Cont.

- **E.g. 2**



Straight–line distance to Bucharest	
Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	178
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	98
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

Cont.

- **Properties**
  - **Incomplete and Not optimal**
    - May not reach goal state because it may start down an infinite path and never return to try other possibilities
  - **Exponential time complexity and memory requirement**
    - It retains all the nodes in memory
    - With good heuristic function (that estimate cost nearer to actual cost) the space and time complexity can be reduced substantially

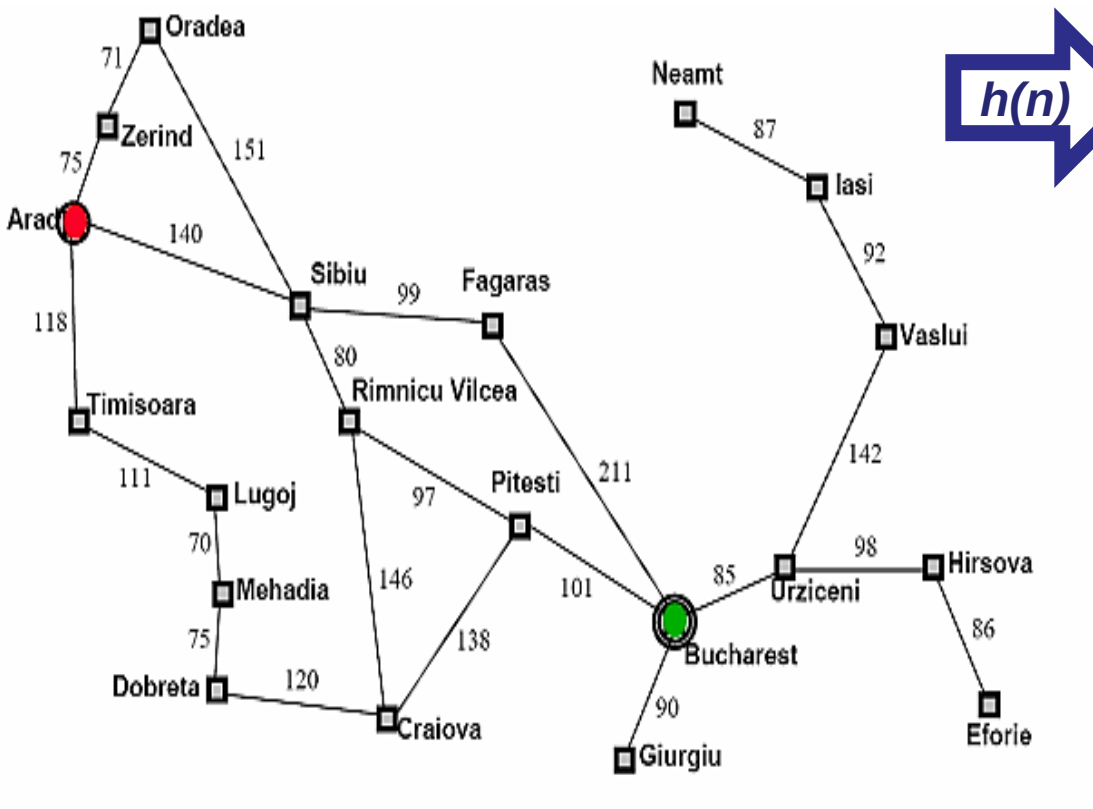
- The **problem** with greedy search is that *it didn't take costs so far into account.*
  - We need a search algorithm (like A* search) that takes into consideration this cost so that it avoids expanding paths that are already expensive

Informed search- A*-search Algorithm

- It considers both *estimated cost of getting from n to the goal node* **h(n)**, and *cost of getting from initial node to node n*, **g(n)**
- Apply three functions over every nodes
  - g(n): Cost of path found so far from initial state to n
  - h(n): Estimated cost of shortest path from n to z
    - ***E.g. in Route finding*** *h(n) is straight line distance*
  - f(n): Estimated total cost of shortest path from a to z via n
  - Evaluation function f(n) = h(n) + g(n)

- **Implementation**: Expand the node for which the evaluation function f(n) is lowest
  - Rank nodes by f(n) that goes from the start node to goal node via given node

Cont.

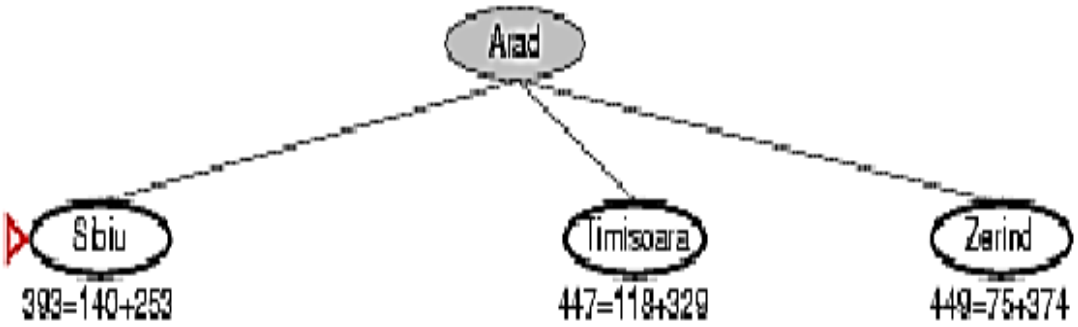


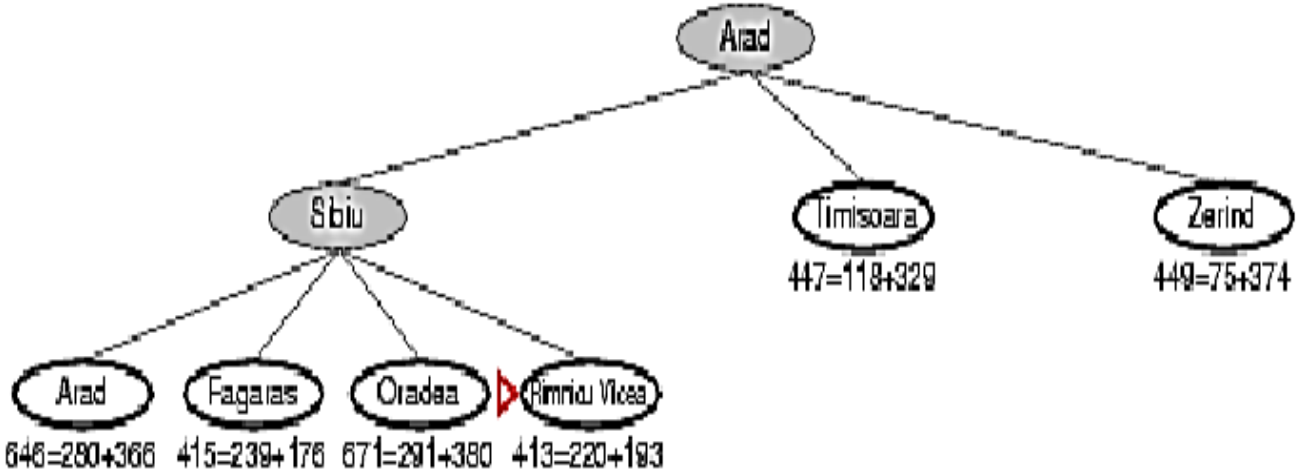
h(n)

Straight–line distance to Bucharest	
Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	178
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	98
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

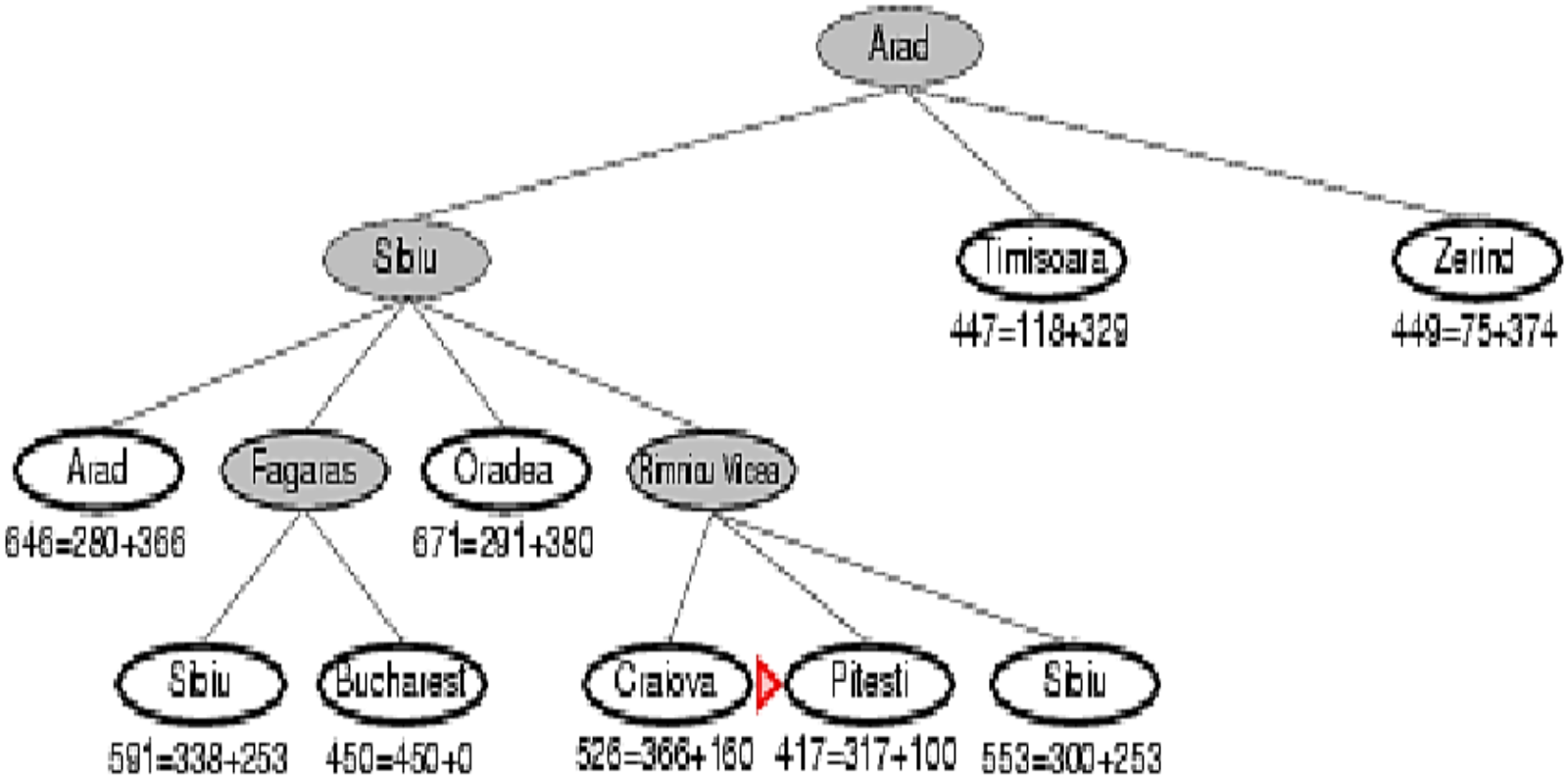
Cont.

- **Solution**

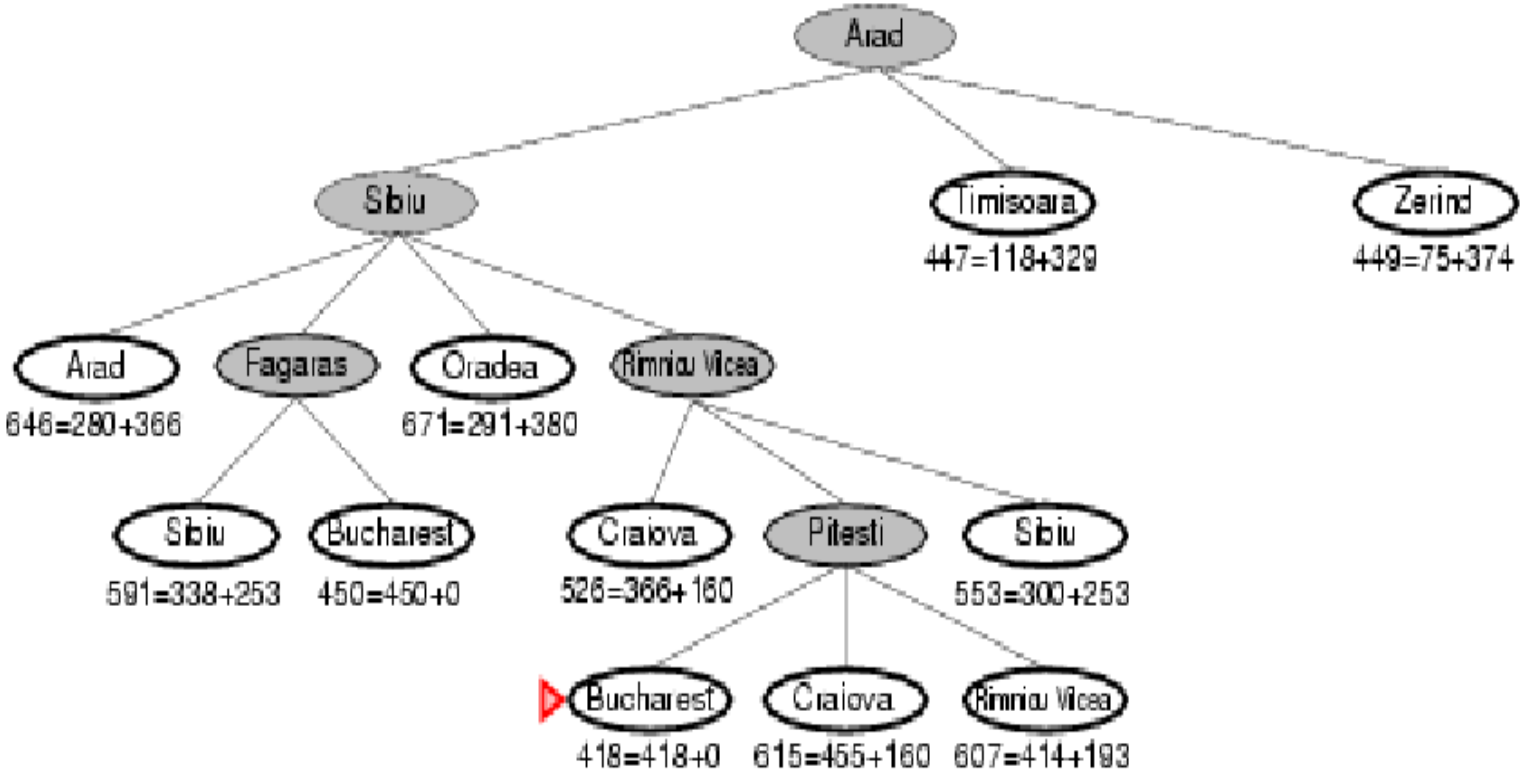




Cont.

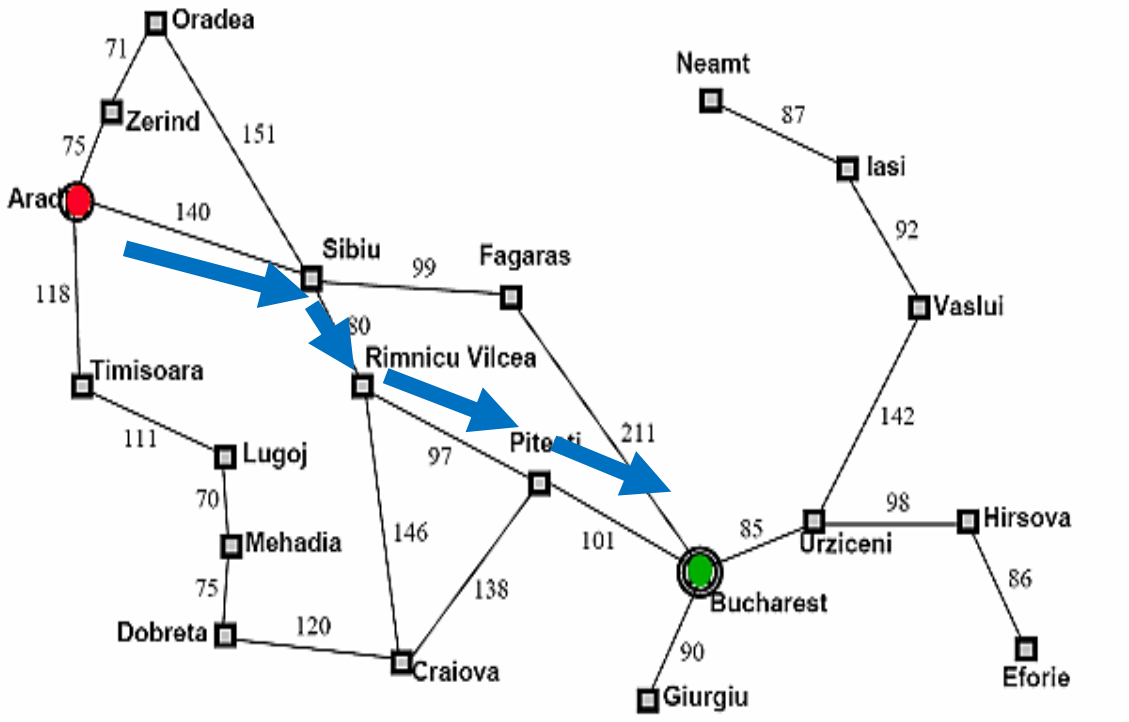

Arad
Sibiu
Timisoara
447=118+329
Zerind
449=75+374
Arad
646=280+366
Fagaras
Oradea
671=291+380
Rimnicu Vilcea
Sibiu
591=339+253
Bucharest
450=450+0
Craiova
526=366+160
Pitesti
417=317+100
Sibiu
553=300+253

Cont.



h(n) = 0 it is the goal state

Cont.


Oradea
71
Zerind
151
75
Arad
140
Sibiu
99
Fagaras
118
80
Rimnicu Vilcea
Timisoara
111
Lugoj
97
Pitesti
211
70
Mehadia
146
101
75
138
Dobreta
120
Craiova
Neamt
87
Iasi
92
Vaslui
142
98
Hirsova
85
Urziceni
86
Bucharest
90
Giurgiu
Eforie

Cont.

- **Properties**
  - **A* is complete**
    - It guarantees to reach to goal state
  - **A* is optimality**
    - If the given heuristic function is admissible, then A* search is optimal.
  - **Takes memory:**
    - It keeps all generated nodes in memory
    - Both time and space complexity is exponential

- Time and space complexity of heuristic algorithms depends on the quality of the heuristic function.

Assignment - 2

- Read and discuss each of the following searching algorithms using an example clearly.
  - ***AO* search***
  - ***Iterative improvement,***
  - ***Constraint satisfaction***
  - ***Problem reduction***
  - ***Hill climbing***
  - ***Generate and test***

Group Assignment (Due: 30 days)

Consider one of the following problem:

- Missionary-and-cannibal problem
- Towers of Hanoi
- Tic-Tac-Toe
- Travelling salesperson problem
- 8-queens problem
- 5 sticks
- Water Jug Problem(3:4; 4:5)
- 8-puzzle problem
- Monkey and Bananas problem
- Crypt arithmetic

1. Identify the set of states and operators; based on which construct the state space of the problem
2. write goal test function
3. Determine path cost
4. Implement the solution

CHAPTER : FOUR
Knowledge representation (KR)

Knowledge

Knowledge and Its Types

- ✓ First let's understand what's "knowledge"? Some scholars define knowledge as "Understanding of a subject area", but in various subject areas it's well-focused as "knowledge domain" like Medical Domain, Engineering Domain, Business Domain, etc.

Knowledge includes facts about the real world entities and the relationship between them

Knowledge-based Systems (KBSs) are useless without the ability to represent knowledge.

we can categorize these knowledge types broadly as follows

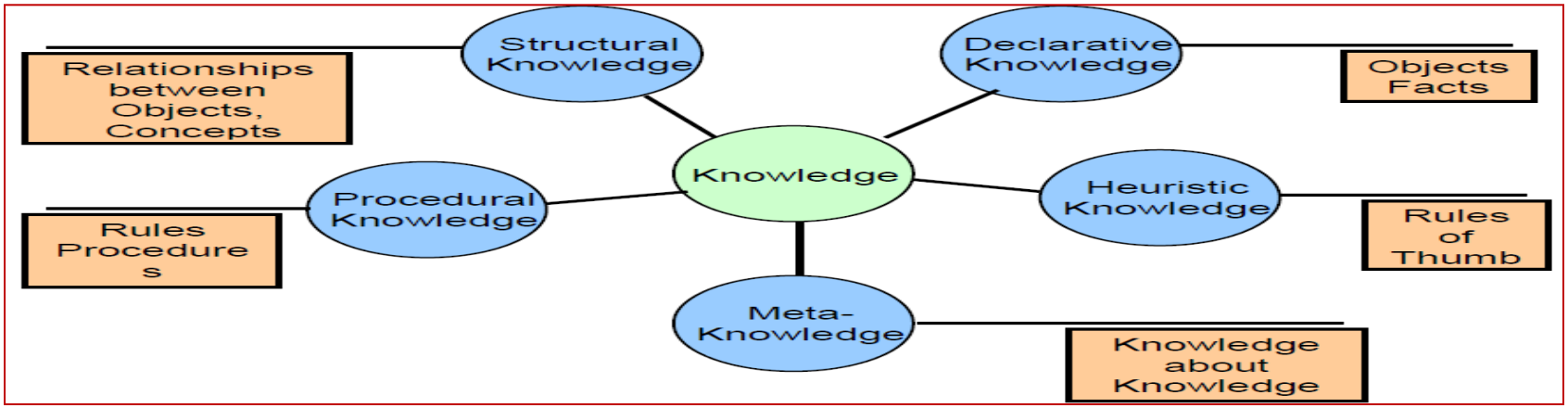
i) **Procedural Knowledge:** It describes how to do things, provides a set of directions of how to perform certain tasks. For example, how to drive a car.

ii) **Declarative Knowledge:** It describes objects, rather than processes or what is known about a situation. For example, it is sunny today and cherries are red.

iii) **Meta Knowledge:** It's knowledge about knowledge. For example, to diagnose someone's medical condition knowledge of Blood Pressure is more important than the Eye Color.

iv) Heuristic Knowledge: It's Rule-of-Thumb, for example, "if I start seeing shops, I am close to the market.". Heuristic knowledge is sometimes called ***shallow knowledge***.

v) Structural Knowledge: Describes structures and their relationships. For example, how the various parts of a car fit together to make the car. So it could be knowledge structures in terms of concepts, sub concepts and objects.



Why Knowledge is important ?

- It enables to:
  - Automate reasoning
  - Discover new facts.
  - Deduce new facts that follow from the KB
  - Answer users queries
  - Make quality decisions - select courses of actions, etc.
- Hence, there is a need to represent knowledge to ease the development of an intelligent system.

– **Towards Representation**

- Thinking about knowledge representation there are multiple approaches and schemes, it can be in ***Pictures and Symbols*** (it's how the earliest humans represented knowledge when sophisticated linguistic systems had not yet evolved), ***Graphs and Networks*** and ***Numbers***.

1. Pictures

- Knowledge can be represented by pictures, for example relationships between individuals in a family. We could also use a series of pictures to store procedural knowledge like how to boil an egg. Thus, we can easily see that pictures are best suited for recognition tasks and for representing structural information.
- Since pictures provide a high-level view of a concept to be obtained they are useful for human understanding, but translating the useful information of pictorial representations in to computers is not as straight forward or *not very easily*, because computers cannot interpret pictures directly without complex reasoning. So. See Figure it's a pictorial representation of a knowledge of some family relationship.

Figure : Pictorial Representation of Family Relationships

- ✓ This kind of representation makes sense readily to humans, but if we give this picture to a computer, it would not have an easy time figuring out the relationships between the individuals or even figuring out how many individuals are there in the picture.
- ✓ For instance, computers might need complex computer vision algorithms to understand pictures.

2. Graphs and Networks

Graphs and networks allow relationships between objects/entities to be incorporated, for example in the below Figure a graph is used to show family relationships. This representation highlights family relationship and its more direct.

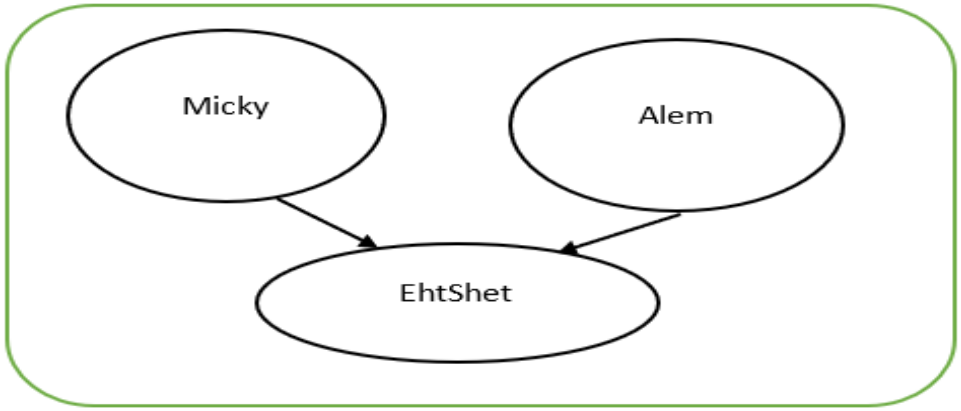


Figure : Graph Representation of Family Relationships

In words we can describe the above family relationship as follows: -

- Micky is EhtShet's Father
- Alem is EhtShet's Mother
- EhtShet is Micky and Alem's Daughter

We can also represent procedural knowledge using graphs, as shown in Figure below it is a knowledge about “how to start a car”.

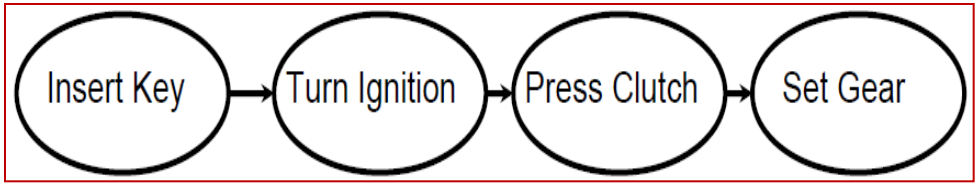


Figure :Graph Representation for Procedural Knowledge

3. Numbers

- ✓ Numbers are an integral part of knowledge representation used by humans and they can be easily translated into computers. Because, eventually every representation we use gets translated into numbers in the computer’s internal representation.
- ✓ Remember that each knowledge representation scheme has its own strengths and weaknesses.

Formal Knowledge Representation Techniques

- ✓ In the above examples we explored intuitive ways of knowledge representations, but how the formal Knowledge Representation (KR) techniques in AI is discussed next.
- ✓ Each technique is suited to represent a certain type of knowledge, so choosing proper representation is an important aspect because it's related with reasoning.

1. Facts

- ✓ Facts are a basic building blocks of knowledge, or they are atomic units of knowledge. Fact represent declarative knowledge, for example we can declare knowledge about objects.
- ✓ A ***proposition*** is a statement of some given fact, so that each proposition has an associated truth value (that is either ***True*** or ***False***).
- ✓ Thus, in AI we represent facts with such propositions and associated truth values, for example:

- Proposition A: "***It is raining.***"
- Proposition B: "***I have an umbrella.***"
- Proposition C: "***I will go to school.***"

1. 1. Types of Facts

A) <u>Single-Valued</u> or ***<u>Multiple–Valued</u>***

•Facts may be single-valued or multi-valued, where each fact (attribute) can take one or more values at the same time.

•For example, an individual can only have one eye color, but may have many cars, so that attribute of cars may contain more than one value.

B) <u>Uncertain Facts</u>

•Sometimes we need to represent facts with uncertain information, for example "***it will probably be sunny today***".

•We may choose to store numerical certainty values to tell that how much uncertainty there is in the fact.

C) <u>Fuzzy Facts</u>

•Fuzzy facts are ambiguous in nature, for example "***the book is heavy or light***", because it is unclear what heavy means or it is a subjective description.

•While defining fuzzy facts, we use certainty factor values to specify value of "truth".

D) <u>Object-Attribute-Value (OAV) Triplets</u>

- Object-Attribute-Value Triplets or OAV triplets are a type of fact composed of three parts (object, attribute and value).
- Such facts are used to assert a particular property of some object, for example in the fact "***Ali's eye color is brown.***" Ali is the Object, Eye Color is an Attribute and its Value is Brown.
- Again in the fact "Ahmed's son is Ali." Object is Ahmed, Attribute is Son and Value is Ali (Figure describes these OAV Triplets diagrammatically).

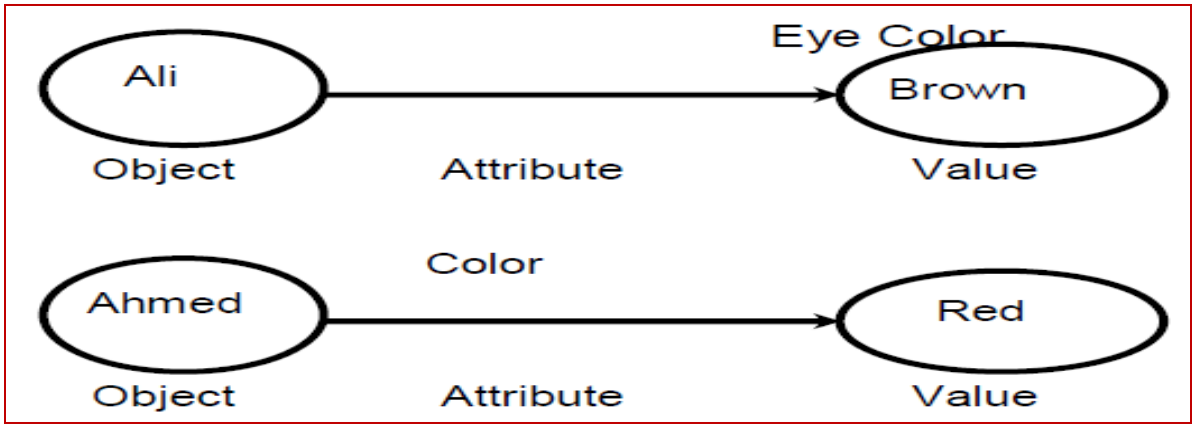


Figure : Example of OAV Triplets

2. Rules

- Rules are another form of knowledge representation and we can define rules as follows.
- "***A knowledge structure that relates some known information to other information that can be concluded or inferred to be true.***"

1. Components of a Rule

- A Rule consists of two components: -

a) Antecedent or Premise (which is the IF part)

b) Consequent or Conclusion (which is the THEN part)

- Example: In the rule "***If it is raining, then I will not go to school.***"

  The Premise is: "***It is raining***"

  The Conclusion is: "***I will not go to school***"

2. Compound Rules

- We can also use connectives like ***AND*** (Conjunctions) and **OR** (Disjunctions) to join multiple premises or antecedents, look examples below.
- "IF it is raining **AND** I have an umbrella, THEN I will go to school."
- "IF it is raining **OR** it is snowing, THEN I will not go to school."

Semantic Networks

- ✓ Semantic networks are graphs with Nodes representing objects and Arcs representing relationships between objects.
- ✓ Various types of relationships maybe defined using semantic networks, but the two most common types of relationships are the IS-A (Inheritance Relation) and the HAS (Ownership Relation).
- ✓ Let's consider an example that demonstrates how knowledge is represented in a semantic network, as shown in Figure

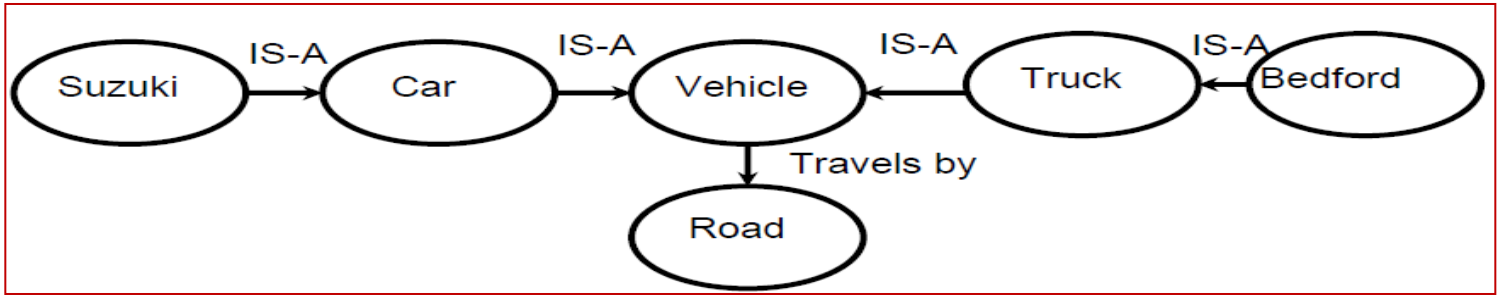


Figure : Vehicle Semantic Network

Network Operation

As shown below to infer new information from semantic networks, we can ask questions from nodes.

- Ask node Vehicle: "How do you travel?"
  - ➢This node looks at arc and replies: Road

- Ask node Suzuki: “How do you travel?”
  - This node does not have a link to travel therefore it asks other nodes linked by the IS-A link.
  - Asks node Car (because of IS-A relationship)
  - Asks node Vehicle (IS-A relationship)
  - Node Vehicle Replies: Road

Frames

- we can define frames as “data structures for representing stereotypical knowledge of some concept or object”.
- So that a frame is like a schema as we would call it in a database design, they were developed from semantic networks and later evolved into our modern-day Classes and Objects.
- For example, Figure : is a frame representation of student.

Frame Name: Student

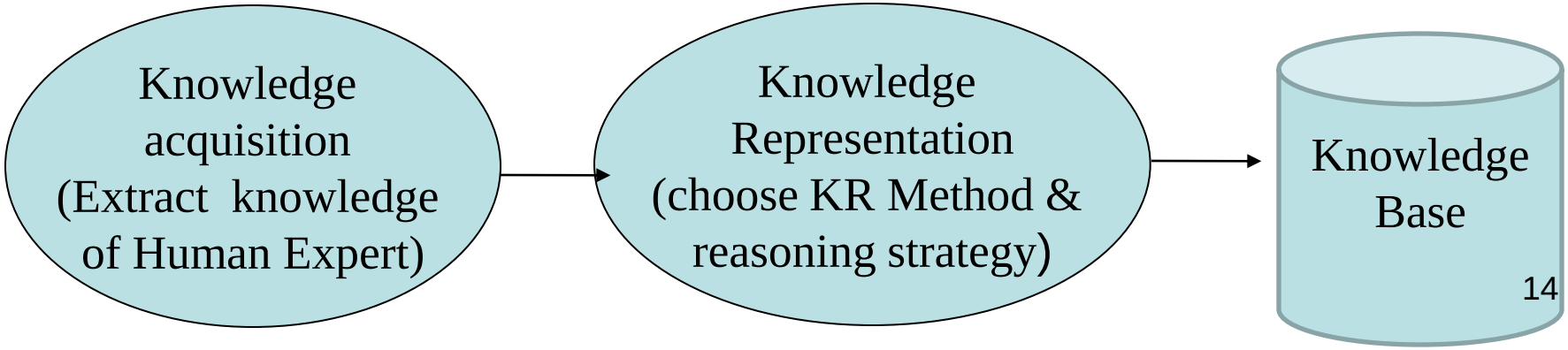
Properties:
Age: 19
GPA: 4.0
Ranking: 1

Figure : Example of Frame Representation

The various components within the frame are called slots, or example Name is a slot.

Knowledge engineering (KE)

- KE is the process of building a knowledge base through extracting the knowledge from the human expert.
- Knowledge engineering is the process of
  - –Extracting the knowledge from the human expert.
  - –Choose knowledge representation formalism
  - –Choose reasoning and problem solving strategy.
- A knowledge engineer is someone who investigates a particular domain, determines what concepts are important in that domain, and creates a formal representation of the objects and relations in the domain.
  - –A KE has to decide what objects and relations are worth representing, and which relations hold among which objects



The Three main tasks of KE

- **Knowledge acquisition:** The knowledge engineer interview the real human experts to be educated about the domain and to elicit the required knowledge, in a process called knowledge acquisition
- **Knowledge Representation** techniques such as logic are a powerful tool for KR and reasoning. However, such techniques consists of only the syntax, semantics and proof theory.
  - KR techniques do not offer any guidance as to what facts should be expressed, nor what vocabulary should be used to express them
- **Knowledge base** is used to store a set of facts and rules about the domain expressed in a suitable representation language
  - Each individual representation are called sentences
  - Sentences are expressed in a (formal) **knowledge representation (KR) language**

LOGIC AND REASONING

Logic

- ✓ One of the prime activities of human intelligence is reasoning.
- ✓ The activity of reasoning involves construction, organization and manipulation of statements to arrive at new conclusions.
- ✓ Thus, logic can be defined as a scientific study of the process of reasoning and the system of rules and procedures that help in the reasoning process. There are two logic types: -
    - ➢ Propositional Logic (PL)
    - ➢ Predicate Logic or First-Order Predicate Logic (FOPL)

Propositional Logic (PL)

- ✓ Propositional Logic (PL) is the simplest form of logic.
- ✓ It is made of statements and we call each statement as proposition.
- ✓ In propositional logic a proposition takes only two values, that is either TRUE or FALSE.

For example, consider the following statements: -

Statement 1:Gondar was former capital city of Ethiopia.

Statement 2: Sun rises in the west.

Statement 3: What is your name?

- First two statements are propositions with Statement 1 having a value of TRUE and Statement 2 having a value of FALSE.
- Third statement is not a proposition, because it can't take a value TRUE or FALSE.

As shown below, usually we can assign a symbolic variable to represent a proposition.

P: It is raining

Q: I carry an umbrella

There are two kinds of propositions, these are **Atomic** or **Simple** Propositions and **Compound** or **Molecular** Propositions

- ✓ Compound Propositions are formed by combining one or more atomic propositions using a set of logical connectives. For example, "Addisu is member of ITD staff." ITD is an atomic proposition. But "Addisu is member of ITD staff and he is also member of CG chair." is a compound proposition, because it is formed from two atomic propositions.
- ✓ When looking at the real-world problems compound propositions are more useful than atomic propositions.
- ✓ The logical connectives that are generally used to form compound propositions are given below in Table.

Name	Connective	Symbols
Conjunction	AND	$\wedge$
Disjunction	OR	$\vee$
Negation	NOT	$\neg$
Conditional	If … then (IMPLIES)	$\rightarrow$
Bi-conditional	If and only if (IFF)	$\leftrightarrow$

Table : Commonly Propositional Logic Connectives

- ✓ In addition to the connectives listed above in Table , ***Parentheses*** (left and right), ***Braces*** (left and right), the ***Period*** and delimiters are also used for punctuation.
- ✓ For example, the compound sentence "**It is raining and the wind is blowing.**" could be represented as **R∧B** where **R** stand for propositions "**It is raining.**" and **B** stand for "**The wind is blowing.**".

- ✓ If we write **R∨B** we mean "**It is raining or the wind is blowing or both**.".

Let's sum up the above discussion with the following points.

- Some examples of Atomic or Simple Propositions are: -
  - ✓ It is raining.
  - ✓ My home is painted silver.
  - ✓ Our PM has a PhD in Computer Science.
  - ✓ People live on the moon.
- Examples of Compound Propositions are: -
  - ✓ It is raining and the wind is blowing.
  - ✓ Addis Ababa is capital of Ethiopia and electricity is available 24 hours in Ethiopia.
  - ✓ If you study hard you will be rewarded.
- Most often we use capital letters to stand for propositions (sometimes followed by digits), and we use the special symbols T and F for the values true and false, respectively.

Syntax of Propositional Logic

The syntax of PL is defined recursively as follows: -

- **T** and **F** are **formulas**.
- If **P** and **Q** are **formulas**, the following are formulas: -
  - $\neg$**P**
  - **P** $\wedge$ **Q**
  - **P** $\vee$ **Q**
  - **P** $\rightarrow$ **Q**
  - **P** $\leftrightarrow$ **Q**

All formulas are generated from a finite number of the above operations.
An example of a compound formula is: -

((P $\wedge$ Q) $\rightarrow$ (R $\vee$ $\neg$S))

When there is no chance of ambiguity, we will omit parentheses, for example if we omit the parentheses from the above statement, we can get the following: -

(P $\wedge$ Q) $\rightarrow$ (R $\vee$ $\neg$S)

Omitting the parentheses, given the connectives their precedence from highest to lowest is:-

$\neg$, $\wedge$, $\vee$, $\rightarrow$, and $\leftrightarrow$

Propositional logic (PL) sentences

- A sentence is made by linking prepositional symbols together using logical connectives.
  - There are atomic and complex sentences.
  - Atomic sentences consist of propositional symbol (e.g. P, Q, TRUE, FALSE)
  - Complex sentences are combined by using **connectives** or **parenthesis:**
  - while S and T are atomic sentences, $S \vee T$, $(S \vee T)$, $(S \wedge T)$, $(S \rightarrow T)$, and $(S \leftrightarrow T)$ are complex sentences.

Examples: Given the following sentences about the "weather problem" convert them into **PL sentences:**

- "It is humid.": Q
- "If it is humid, then it is hot" : $Q \rightarrow P$
- "If it is hot and humid, then it is raining": $(P \wedge Q) \rightarrow R$

Exercise

Examples: Convert the following **English sentences to Propositional logic**

Let A = Lectures are active and R = Text is readable, P = Kebede will pass the exam, then represent the following:

- the lectures are ***not*** active: $\neg A$
- the lectures are active **and** the text is readable: $A \wedge R$
- either the lectures are active **or** the text is readable: $A \vee R$
- **if** the lectures are active, **then** the text is not readable: $A \rightarrow \neg R$
- the lectures are active **if and only if** the text is readable: $A \leftrightarrow R$
- if the lectures are active, then if the text is not readable, Kebede will not pass the exam: $A \rightarrow (\neg R \rightarrow \neg P)$

Semantics of Propositional Logic

- ✓ The semantics or meaning of a sentence is just the values true or false, that is an assignment of a truth value to the sentence.
- ✓ An interpretation for a sentence or group of sentences is an assignment of a truth value to each propositional symbol.

For example, consider the statement $(P \wedge \neg Q)$.

- ✓ One interpretation assigns true (T) to P and false (F) to Q. A different interpretation assigns T to P and T to Q.
- ✓ Clearly, there are four distinct interpretations for this sentence.

Table below summarizes the semantic rules where T and T' denote any true statements, F and F' denote any false statements and A (either true or false) is any statement.

Rule Number	True Statements	False Statements
1	T	F
2	$\neg F$	$\neg T$
3	$T \wedge T'$	$F \wedge A$
4	$T \vee A$	$A \wedge F$
5	$A \vee T$	$F \vee F'$
6	$A \rightarrow T$	$T \rightarrow F$
7	$F \rightarrow A$	$T \leftrightarrow F$
8	$T \leftrightarrow T'$	$F \leftrightarrow T$
9	$F \leftrightarrow F'$	

Table : Semantics Rules for Statements

•We can now find the meaning of any statement for a given interpretation, for example let an interpretation assigns **T** to **P**, **F** to **Q** and **F** to **R** in the following statement: -

• $$((P \wedge \neg Q) \rightarrow R) \vee Q$$

Application of Rule 2 then gives $\neg Q$ as **T**, Rule 3 gives $(P \wedge \neg Q)$ as **T**, Rule 6 gives $(P \wedge \neg Q) \rightarrow R$ as **F**, and Rule 5 gives the statement value as **F** (false).

•A clear meaning of the logical propositions can be arrived at by constructing appropriate truth tables for the compound propositions.

A	**B**	**¬A**	**¬B**	**A ∨ B**	**A ∧ B**	**A → B**	**A ↔ B**
T	T	F	F	T	T	T	T
F	T	T	F	T	F	T	F
T	F	F	T	T	F	F	F
F	F	T	T	F	F	T	T

Table : Truth Tables for Logical Connectives

Properties of Statements

- Some properties of statements are stated below: -

I. Satisfiable: A statement is satisfiable if there is some interpretation for which it is true.

- Example is **P**.

II. Contradiction: A sentence is contradictory (unsatisfiable) if there is no interpretation for which it is true.

- For example, **P ∧ ¬P** is contradiction since every interpretation result in a value of false for **(P ∧ ¬P)**.

III. Tautology (Valid): A sentence is valid if it is true for every interpretation.

- For example, **P ∨ ¬P** is valid since every interpretation result in a value of true for **(P ∨ ¬P)**.
- Remember that a valid sentence is also called tautology.

IV. Equivalence: Two sentences are equivalent if they have the same truth value under every interpretation.

- For example, **P** and **¬(¬P)** are equivalent since they have the same truth values under every interpretation.

Terminology

- **Valid sentence:** A sentence is **valid sentence** or **tautology** if and only if it is True under all possible interpretations in all possible worlds.

  Example: "It's raining or it's not raining." ($R \vee \neg R$).

- **Satisfiable**: A sentence is satisfiable if and only if there is some interpretations in some world for which the sentence is True.

  Example: "It is raining or it is humid". R v Q, R

- **Unsatisfiable:** A sentence is unsatisfiable (inconsistent sentence or self-contradiction) if and only if it is not satisfiable, i.e. a sentence that is False under all interpretations. The world is never like what it describes.

  Example: "It's raining and it's not raining." $R \wedge \neg R$

Inference Rules

- Inference rules of PL provide means to perform logical ***proofs*** or ***deductions***, here below are few inference rules.

1) Modus Ponens

- From **P** and **P → Q** we can infer **Q**, this rule can be written as follows: -
- **P**
- **P → Q**
- **∴ Q**

- For example
  - ✓ Given two statements
- "***Adamu is a father.***"
- "***Adamu is a father → Adamu has a child.***"
  - ✓ We can conclude that "***Adamu has a child.***".

1) Chain Rule

- From **P → Q** and **Q → R** we can infer **P → R**, this rule can be written as: -
- **P → Q**
- **Q → R**
- **∴ P → R**

1) Transposition

- From **P → Q** we can infer **¬Q → ¬P**, this rule can be written as follows: -
- **P → Q**
- **∴ ¬Q → ¬P**

First-Order Predicate Logic (FOPL)

- First-Order Logic (FOL) is expressive enough to concisely represent any kind of situation that are expressed in natural language.
- FOL is type of PC (Predicate calculus)
- Propositional logic works fine in situations where the result is either TRUE or FALSE, but not both.
- Expressiveness is one of the requirements for any serious representation scheme, furthermore PL does not permit us to make generalized statements about classes of similar objects.
- For example, from the following two statements, it should be possible to conclude that ***"kebede must take Calculus course."***.
- ***"All students in Computer Science must take Calculus."***
- ***"kebede is a Computer Science student."***

As stated, it is not possible to conclude in PL that ***kebede*** must take ***Calculus*** since the second statement does not occur as part of the first one.

- FOPL was developed by logicians to extend the expressiveness of PL.
- It is a generalization of PL that permits reasoning about world objects as relational entities as well as classes of objects.

1. ***Syntax of FOPL***

- The symbols and rules of combination permitted in FOPL are defined as follows: -

I. Connectives

- There are five connectives symbols, these are: -

- ¬ **Not (Negation)**
- ∧ **And (Conjunction)**
- ∨ **Or (Disjunction)**
- → **Implication**
- ↔ **Equivalence (If-and-only-if)**

II. Quantifiers

- Predicate calculus allows us to use quantifiers for statements in order to refer ***Some*** or ***All*** objects from some set of objects. In basic predicate calculus we use two types of logical quantifiers, these are the Universal Quantifies and the Existential Quantifier.

A) <u>The Universal Quantifier</u> (∀)

- The symbol for the universal quantifier is ∀, it is read as "***for every***" or "***for all***" and used in formulae to assign the same truth value to all variables in the domain.
  - For example, in the domain of numbers, we can say **(∀X) (X + X = 2X)**.
  - It means that "***For every x (where x is a number), x + x = 2x is true.***".

B) <u>Existential</u> <u>Quantifier</u> (∃)

- The symbol for the existential quantifier is ∃, it is read as "***there exists***" or "***for some***" or "***for at least one***" or "***there is one***". It is used in formulae to say that something is true for at least one value in the domain.
  - For example, for domain of persons, ***(∃X) (Person(X) ∧ Father(X, Alemu))***.
  - This can be read as "***There exists some person, X who is Alemu's father.***".

III. Constants

- Constants are fixed-value terms that belongs to a given domain of discourse. Constants are used to name specific objects or properties, for example Mule, Tomi, Blue, Ball, etc.

IV. Predicates

•A fact or proposition is divided into two parts: -

i)Predicate: It is the assertion of the proposition.

ii)Argument: It is the object of the proposition.

•For example, the proposition "***Addisu likes Tibs.***" can be represented in predicate logic as ***Likes(Addisu, Tibs)***, where ***Likes*** is the predicate and ***Addisu*** and ***Tibs*** are the arguments.

V. Variables

•Variables are used to a represent general class of objects/properties, for example in the predicate **Likes(X, Y)**, **X** and **Y** are variables that assume the values **X=Addisu** and **Y=Tibs**.

VI. Functions

•Function symbols denote relations defined on a domain, for example symbols **f**, **g** and **words** represent functions.

VII. Formulae

•Formulas combine predicates and quantifiers to represent information, see Figure below that illustrate these symbols using an example.

man (addisu)

father (addisu, mesfin)

brother (addisu, asfaw)

owns (home, car)

tall (asfaw)

hates (addisu, lies)

family ()

Predicates

$\forall$ Y ($\neg$sister(y, addisu))

$\forall$ X, Y, Z (man(X) $\wedge$ man(Y) $\wedge$ father (Z, Y) $\wedge$ father (Z, X) $\rightarrow$ brother(X, Y)

Formulae

X, Y and Z

Variables

addisu, mesfin, lies, car

Constant

Figure : Example of Formulae

- The predicate section outlines the known facts about the situation in the form of predicates, that is predicate name and its arguments.
- So, **man(addisu)** means that **addisu** is a **man**, **hates(addisu, lies)** means that **addisu** hates **lies**.
- The formulae section outlines formulae that use universal quantifiers and variables to define certain rules.

- **∀Y (¬sister(Y, addisu))** says that there exist
- s no **Y** such that **Y** is the **sister** of **addisu**, that addisu has no sister.
- Similarly,**∀X, Y, Z (man(X) ∧ man(Y) ∧ man(Z) ∧ father(Z, Y) ∧ father(Z, X) → brother(X, Y))** means that if there are three men, **X**, **Y** and **Z**, and **Z** is the father of both **X** and **Y**, then **X** and **Y** are bothers. This expresses the rule for the two individuals being brothers.

- Suppose we want to represent the following statements in FOPL form: -
- E1: All employees earning Birr 14000 or more per year pay taxes.
- E2: Some employees are sick today.
- E3: No employee earns more than the president

- To represent such expressions in FOPL, we must define abbreviations for the predicates and functions. We might for example, define the following:
- E(X) for X is an employee.
- P(X) for X is president.
- I(X) for the income of X (lower case denotes a function).
- GE(U, V) for U is greater than V.
- S(X) for X is sick today.
- T(X) for X pays taxes.

- Using the above abbreviations, we can represent E1, E2 and E3 as
- E1: $\forall X((E(X) \wedge GE(I(X), 14000)) \rightarrow T(X)$
- E2: $\exists X\ (E(X) \rightarrow S(X))$
- E3: $\forall X \exists Y\ ((E(X) \wedge P(Y)) \rightarrow \neg GE(I(X), I(Y)))$

Reasoning

- So far in this chapter we've looked at knowledge representation and logic, next is about mechanisms to reason on the knowledge once we have represented it using some logical scheme.

- ***What's Reasoning?***

- Reasoning is the process of deriving logical conclusions from given facts.
- According to Durkin definition reasoning is "***the process of working with knowledge, facts and problem-solving strategies to draw conclusions***".

- In this section we'll look at six types of reasoning: -

1. Deductive Reasoning

- As the name implies, Deductive Reasoning is based on deducing new information from logically related known information. A deductive argument offers assertions that lead automatically to a conclusion, for example: -

- **If there is dry wood, oxygen and a spark, there will be a fire**.
  - ✓Given: **There is dry wood, oxygen and a spark/flash**.
  - ✓We can deduce: **There will be a fire**.
- Given: **All men are mortal. Socrates is a man.**
  - ✓We can deduce: **Socrates is mortal.**

2. Inductive Reasoning

- Inductive reasoning is based on forming, or inducing a "generalization" from a limited set of observations, for example: -

- Observation: **All the crows that I have seen in my life are black**.
- Conclusion: **All crows are black**

- ***<u>Comparison of Deductive and Inductive Reasoning</u>***
- We can compare deductive and inductive reasoning using an example. We conclude what will happen when we let a ball go using both each type of reasoning in turn.

- The inductive reasoning is as follows: -
- **By experience, every time I have let a ball go, it falls downwards. Therefore, I conclude that the next time I let a ball go, it will also come down.**

- The deductive reasoning is as follows: -
- **I know Newton's Laws. So, I conclude that if I let a ball go, it will certainly fall downwards.**

- Thus, the essential difference is that inductive reasoning is ***based on experience*** while deductive reasoning is ***based on rules***, hence the latter will always be correct.

3. Abductive Reasoning

- Deduction is exact in the sense that deductions follow in a logically provable way from the axioms. Abduction is a form of deduction that allows for plausible inference, that is the conclusion might be wrong, for example: -
  - Implication: **She carries an umbrella if it is raining.**
  - Axiom: **She is carrying an umbrella.**
  - Conclusion: **It is raining.**
- This conclusion might be false, because there could be other reasons that she is carrying an umbrella, for example she might be carrying it to protect herself from the sun.

4. Analogical Reasoning

- Analogical reasoning works by drawing analogies between two situations, looking for similarities and differences. For example, when you say driving a truck is just like driving a car, by analogy you know that there are some similarities in the driving mechanism, but you also know that there are certain other distinct characteristics of each.

5. Common-Sense Reasoning

- Common-sense reasoning is an informal form of reasoning that uses rules gained through experience or what we call rules-of-thumb. It operates on heuristic knowledge and heuristic rules.

6. Non-Monotonic Reasoning

- Monotonic reasoning is used when the facts of the case are likely to change after some time, for example: -
  - Rule: **IF the wind blows, THEN the curtains sway.**
  - When the wind stops blowing, the curtains should sway no longer.
- In non-monotonic reasoning, we have a “truth maintenance system”.
- It keeps track of what caused a fact to become true.
- If the cause is removed, that fact is removed also.

Inference

- Inference is the process of deriving new information from known information.
- In the domain of AI, the component of the system that performs inference is called an inference engine.
- We will look at inference within the framework of “logic”, which we introduced earlier.
- Logic, which we introduced earlier, can be viewed as a formal language. As a language, it has the following components: Syntax, Semantics and Proof Systems.

Syntax

- Syntax is a description of valid statements, the expressions that are legal in that language.
- We have already looked at the syntax of two types of logic systems called propositional logic and predicate logic.
- The syntax of proposition gives us ways to use propositions, their associated truth value and logical connectives to reason.

Semantics

- Semantics pertain to what expressions mean, e.g. the expression ‘the cat drove the car’ is syntactically correct, but semantically non-sensible.

Proof Systems

- A logic framework comes with a proof system, which is a way of manipulating given statements to arrive at new statements.
- The idea is to derive ‘new’ information from the given information.
- A proof is a sequence of statements aiming at inferring some information.
- While doing a proof, we usually proceed with the following steps:

☞ Begin with initial statements, called premises of the proof (or knowledge base).

☞ Use rules, i.e. apply rules to the known information.

☞ Add new statements, based on the rules that match.

☞ Repeat the above steps until you arrive at the statement you wished to prove.

Rules of Inference

- Rules of inference are logical rules that you can use to prove certain things. As you look at the rules of inference, try to figure out and convince yourself that the rules are logically sound, by looking at the associated truth tables.

- The rules we will use for propositional logic are ***Modus Ponens (MP)***, ***Modus Tollens (MT)***, ***And-Introduction ($\wedge_i$)*** and ***And-Elimination ($\wedge_e$)***.

I. <u>Modus Ponens</u> (MP)

- Modus Ponens means "affirming method". In our discussion of logic, anything that is written down in a proof is a statement that is true.
- Modus Ponens says that if you know that **A** implies **B** is true, and you know **A** to be true, you can automatically say that **B** is true.
    - $\mathbf{A \rightarrow B}$
    - $\underline{\mathbf{A}}$
    - $\therefore \mathbf{B}$

II. <u>Modus Tollens</u> (MT)

- Modus Tollens says that from "**A** implies **B**" and "not **B**" you can conclude

- ". In other words, if A implies **B** is true and **B** is known to be not true, then **A** could not have been true. Had **A** been true, **B** would automatically have been true due to the implication.
    - $\mathbf{A \rightarrow B}$
    - $\underline{\mathbf{\neg B}}$
    - $\therefore \mathbf{\neg A}$

III. <u>And-Introduction</u> ($\wedge_i$)

- And-introduction says that from "**A**" and from "**B**" you can conclude "**A** and **B**". That seems pretty obvious, but is a useful tool to know upfront.
    - **A**
    - $\underline{\mathbf{B}}$
    - $\therefore \mathbf{A \wedge B}$

IV. <u>And-Elimination</u> ($\wedge_e$)

- And-elimination says that from "**A** and **B**" you can conclude "**A**". It also says that from "**A** and **B**" you can conclude "B".
    - $\underline{\mathbf{A \wedge B}}$
    - $\therefore \mathbf{A}$

Example: Proofing Using Inference Rules

Proof E from the following given three premises, by applying the above inference rules.

Premise 1: $(A \vee B) \rightarrow (C \quad \neg D)$

Premise 2: $\neg E \rightarrow D$

Premise 3: $A \quad B$

Proof Steps

$A \vee B) \rightarrow (C \wedge \neg D)$	Given Premise
$\neg E \rightarrow D$	Given Premise
$A \vee B$	Given Premise
$(C \wedge \neg D)$	Apply MP to 1 and 3
$\neg D$	Apply $\wedge_e$ to 4
$\neg (\neg E)$	Apply MP to 2 and 5
E	Proved

FOL: Basic elements

– Two standard quantifiers:

- Universal - $\forall$ - (for all, for every) **and** Existential - $\exists$ - (there exists, some)

FOL represents objects and relations between objects, variables, and quantifiers in addition to propositions

- **Universal Quantifiers**: makes statements about every object

  $\forall$<***variables***> <***sentence***>

  –Everyone at AAU is smart:

  $\forall_x$ At(x,AAU) $\Rightarrow$ Smart(x)

  –All cats are mammals:

  $\forall_x$ cat(x) $\Rightarrow$ mammal(x)

- $\forall_x$ sentence *P* is true iff *P* is true with *x* being each possible object in the given universe

  –The above statement is equivalent to the **conjunction**

  At(Jonas, AAU) $\Rightarrow$ Smart(Jonas) $\wedge$

  At(Rawad, AAU) $\Rightarrow$ Smart(Rawad) $\wedge$

  ...

- A common mistake to avoid

  –Typically, $\Rightarrow$ is the main connective with $\forall$

  –**Common mistake**: the use of $\wedge$ as the main connective with $\forall$:

  $\forall_x$ At(x,AAU) $\wedge$ Smart(x) is true if "Everyone is at AAU & everyone is smart"

Existential quantification

- Makes statements about some objects in the universe

  $\exists$<***variables***> <***sentence***>

  –Someone at AAU is smart:

  $\exists_x$ At(x,AAU) $\wedge$ Smart(x)

  –Spot has a sister who is a cat:

  $\exists_x$ sister(spot,x) $\wedge$ cat(x)

- $\exists_x$ sentence *P* is true iff *P* is true with *x* being some possible objects

  –The above statement is equivalent to the **disjunction**

  At(Jonas, AAU) $\wedge$ Smart(Jonas) $\vee$

  At(Alemu, AAU) $\wedge$ Smart(Alemu) $\vee$

  …..

- Common mistake to avoid

  –Typically, $\wedge$ is the main connective with $\exists$

  –Common mistake: using $\Rightarrow$ as the main connective with $\exists$:

  $\exists_x$ At(x,AAU) $\Rightarrow$ Smart(x) is true if there is anyone who is not at AAU

Nested quantifiers

- $\forall_{x,y}$ parent(x,y) $\Rightarrow$ child(y,x)
  - for all x and y, if x is the parent of y then y is the child of x.
- $\exists_x \forall_y$ Loves(x,y)
  - There is a person who loves everyone in the given world
- $\forall_y \exists_x$ Loves(x,y)
  - Everyone in the given universe is loved by at least one person

Properties of quantifiers

- $\forall_x \forall_y$ is the same as $\forall_y \forall_x$
- $\exists_x \exists_y$ is the same as $\exists_y \exists_x$
- $\exists_x \forall_y$ is not the same as $\forall_y \exists_x$

- **Quantifier duality**: each can be expressed using the other, using negation ($\neg$)

  $\forall_x$ Likes(x,icecream) $\neg\exists_x \neg$Likes(x,icecream)

  - Everyone likes ice cream means that there is nobody who dislikes ice cream

  $\exists_x$ Likes(x,cake)

  - There is someone who likes cake means that there is no one who dislikes cake $\neg\forall_x \neg$Likes(x,cake)

Example

- Can have several quantifiers, e.g.,

  $\forall_x \ \exists_y$ loves(x, y)

  $\forall_x$ handsome(x) $\Rightarrow$ $\exists_y$ loves(y, x)

Represent the following in FOL:

- Everything in the garden is lovely → $\forall$x in(x, garden) $\Rightarrow$ lovely(x)
- Everyone likes ice cream → $\forall$x likes(x, icecream)
- Peter has some friends → $\exists$y friends(y, Peter)
- John plays the piano or the violin

  plays(john, piano) v plays(john, violin)
- Some people like snakes -- $\exists$x(person(x) $\wedge$ likes(x, snakes))
- Winston did not write Hamlet – $\neg$write(winston, hamlet)
- Nobody wrote Hamlet – $\neg\exists$x write(x, hamlet)

Unification/union

- Unification is an algorithm for determining the substitutions needed to make two expressions match

  – Unification is a "pattern matching" procedure that takes two atomic sentences as input, and returns the **most general unifier,** i.e., a shortest length substitution list that makes the two literals match.

  – **E.g**: To make, say p(X, X) and p(Y, Z) match, subst(X/ Y) or subst(X/ Z)

- Note: It is possible to substitute two variables with the same value, but not the same variables with different values.

- **Example**

Sentence 1	**Sentence 2**	**Unifier**
group(x, cat(x), dog(Bill))	group(Bill, cat(Bill), y)	{x/Bill, y/dog(Bill)}
group(x, cat(x), dog(Bill))	group(Bill, cat(y), z)	{x/Bill, y/Bill, z/dog(Bill)}
group(x, cat(x), dog(Jane))	group(Bill, cat(y), dog(y))	Failure

Cont…

- A variable can never be replaced by a term containing that variable. For example, x/f(x) is illegal.
- Unification and inference rules allows us to make inferences on a set of logical assertions. To do this, the logical database must be expressed in an appropriate form
- A substitution α unfies atomic sentences p and q if pα = qα

p	q	α
Knows(John,x)	Knows(John,Jane)	{x/Jane}
Knows(John,x)	Knows(y, Abe)	{y/John, x/Abe}
Knows(John,x)	Knows(y,Mother(y))	{y/John, x/Mother(John)}
Knows(John,x)	Knows(x,Abe)	Fail

Unify rule: Premises with known facts apply unifier to conclusion

Example. if we know q and Knows(John,x) → Likes(John,x)
then we conclude Likes(John,Jane)
Likes(John,Abe)
Likes(John,Mother(John))

Inference Mechanisms

- Inference is a means of interpretation of knowledge in the KB to reason and give advise to users query.
- There are two inference strategies to control and organize the steps taken to solve problems:
  - **Forward chaining**: also called data-driven chaining
    - It starts with facts and rules in the KB and try to draw conclusions from the data
  - **Backward chaining**: also called goal-driven chaining
    - It starts with possible solutions/goals and tries to gather information that verifies the solution

Properties of Good KB

- A KB should be:
  - Clear
  - Correct
  - Expressive
  - Concise
  - Context-insensitive and
  - Effective.
- The relations that matter should be defined, and the irrelevant details should be suppressed

CHAPTER : FIVE
Learning Agents

Introduction

- Artificial Intelligence (AI) is concerned with programming computers to perform tasks that are presently done better by humans.
- AI is about human behavior, the discovery of techniques that will allow computers to learn from humans.
- One of the most often heard criticisms of AI is that machines cannot be called Intelligent until they are able to learn to do new things and adapt to new situations, rather than simply doing as they are told to do.
- There can be little question that the ability to adapt to new surroundings and to solve new problems is an important characteristic of intelligent entities.

What is Learning?

- In the psychology literature, learning is considered one of the keys to human intelligence.
- *what learning is?* Learning is:
  - Memorizing something
  - Knowing facts through observation and exploration
  - Improving motor and/or cognitive skills through practice
- The idea behind learning is that percepts should not only be used for acting now, but also for improving the agent's ability to act in the future.
  - Learning is essential for unknown environments, i.e. when designer lacks omniscience. It enables to organize new knowledge into general, effective representations
  - Learning modifies the agent's decision making mechanisms to improve performance

Cont……..Learning

- Learning can be described as normally a relatively permanent change that occurs in behavior as a result of experience.
- Learning occurs in various regimes, for example: -
- It is possible to learn to open a lock as a result of trial and error.
- It is possible to learn how to use a word processor as a result of following particular instructions.
- There's no decisive definition of learning, but here are some that do justice: -
- "***Learning denotes changes in a system that ... enables a system to do the same task more efficiently the next time.***" Herbert Simon
- "***Learning is constructing or modifying representations of what is being experienced.***" Ryszard Michalski
- "***Learning is making useful changes in our minds.***" Marvin Minsky
- The idea behind learning is that percepts should be used not only for acting, but also for improving the agent's ability to act in the future.
- Learning takes place as the agent observes its interactions with the world and its own decision-making processes.

Machine Learning

- The goal of machine learning is to build computer systems that can learn from their experience and adapt to their environments.
- Obviously, learning is an important aspect or component of intelligence.

<u>Why Machine Learning</u>?

- One response to the idea of AI is to say that computers cannot think because they only do what their programmers tell them to do.
- However, it is not always easy to tell what a particular program will do, but given the same inputs and conditions it will always produce the same outputs.
- If the program gets something right once, it will always get it right and if it makes a mistake once it will always make the same mistake every time it runs.
- Also, humans are able to notice similarities between things and therefore can generate new ideas about the world we live in.
- So, machine learning is a prerequisite for any mature program of artificial intelligence.

Three Phases of Machine Learning

- Machine Learning follows three phases, these are: -

<u>Phase I: Training</u>

- A training set of examples of correct behavior is analyzed and some representation of the newly learnt knowledge is stored.
- This is often some form of rules.
- ***<u>Phase II: Validation</u>***
- The rules are checked and if necessary, additional training is given. Sometimes additional test data are used.
- But instead of using a human to validate the rules, some other automatic knowledge-based component may be used.
- The role of tester is often called the critic.
- ***<u>Phase III: Application</u>***
- The rules are used in responding to some new situations.

Learning Techniques

Rote Learning

- In this kind of learning there is no prior knowledge.
- When a computer stores a piece of data, it is performing an elementary form of learning.
- Examples of correct behavior are stored and when a new situation arises it is matched with the learnt examples.
- The values are stored so that they are not re-computed later. One of the earliest game-playing programs is Checkers Program (Samuel, 1963). This program learned to play checkers well enough to beat its creator/designer.

Deductive Learning

- Deductive learning works on existing facts and knowledge and deduces new knowledge from the old.
- This is best illustrated in the following example: -
- **Example**: Assume, **A = B** and **B = C**, then we can deduce with much confidence that **C = A**.

- Deductive learning does not generate “new” knowledge at all, it simply memorizes the logical consequences of what is known already.
- This implies that virtually all mathematical research would not be classified as learning “new” things.
- However, regardless of whether this is termed as new knowledge or not, it certainly makes the reasoning system more efficient.

<u>Inductive Learning</u> (Learning from Observations)

- Inductive learning takes examples and generalizes rather than starting with existing knowledge.
- For example, having seen many cats, all of which have tails, one might conclude that all cats have tails.
- There is scope of error in inductive reasoning, but still it is a useful technique that has been used as the basis of several successful systems.

Learning: Symbol-Based

- Our world is a world of symbols.
- We use symbolic interpretations to understand the world around us.
- For instance, if we saw a ship and were to tell a friend about its size, we will not say that we saw a 254 meters long ship, instead we'd say that we saw a "huge" ship about the size of "Our Building".
- And our friend would understand the relationship between the size of the ship and its hugeness with the analogies of the symbolic information associated with the two words used: "huge" and "Our Building".
- Similarly, the techniques we are to learn now use symbols to represent knowledge and information. Let us consider a small example to help us see where we're headed.
- **Example**: What if we were to learn the concept of a ***GOOD STUDENT***.
- We would need to define, first of all some attributes of a student, on the basis of which we could tell apart the good student from the average.

- Then we would require some examples of good students and average students.
- To keep the problem simple, we can label all the students who are “not good” (Average, Below Average, Satisfactory, Bad) as ***NOT GOOD STUDENT***.
- Let’s say we choose two attributes to define a student, ***Grade*** and ***Class Participation***. Both attributes can have either of the two values, ***High*** or ***Low***.
- Now from the above Student concept, the learner program might look like: -
- Student (GOOD STUDENT): **Grade (High) $\wedge$ Class Participation (High)**
- Student (GOOD STUDENT): **Grade (High) $\wedge$ Class Participation (Low)**
- Student (NOT GOOD STUDENT): **Grade (Low) $\wedge$ Class Participation (High)**
- Student (NOT GOOD STUDENT): **Grade (Low) $\wedge$ Class Participation (Low)**

- As you can see the system is composed of symbolic information, based on which the learner can even generalize that a student is a GOOD STUDENT if his/her grade is **High**, even if the class participation is **Low**.
- Student (GOOD STUDENT): **Grade (High) $\wedge$ Class Participation (?)**
- This is the final rule that the learner has learnt from the enumerated examples.

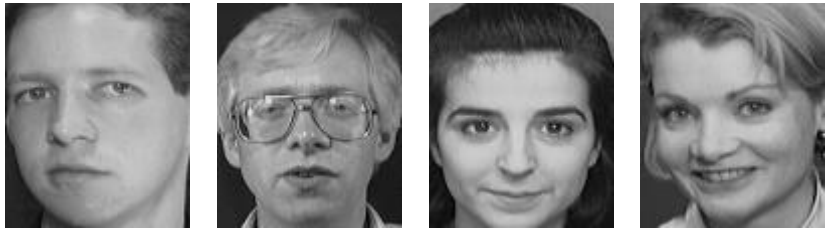
- Here the "**?**" means that the attribute class participation can have any value, as long as the grade is High.

Face Recognition

Learning: it is training (adaptation) from data set

•Training examples of a person –

Test images



Feature extraction

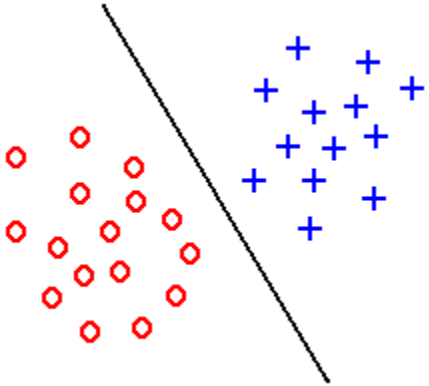
x = Hair length y =0

x = Hair length y = 30 cm

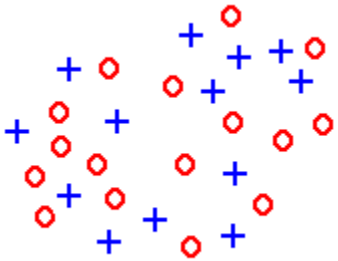
Task: to extract features which are good for classification.

Good features:

- Objects from the same class have similar feature values.
- Objects from different classes have different values.



"Good" features

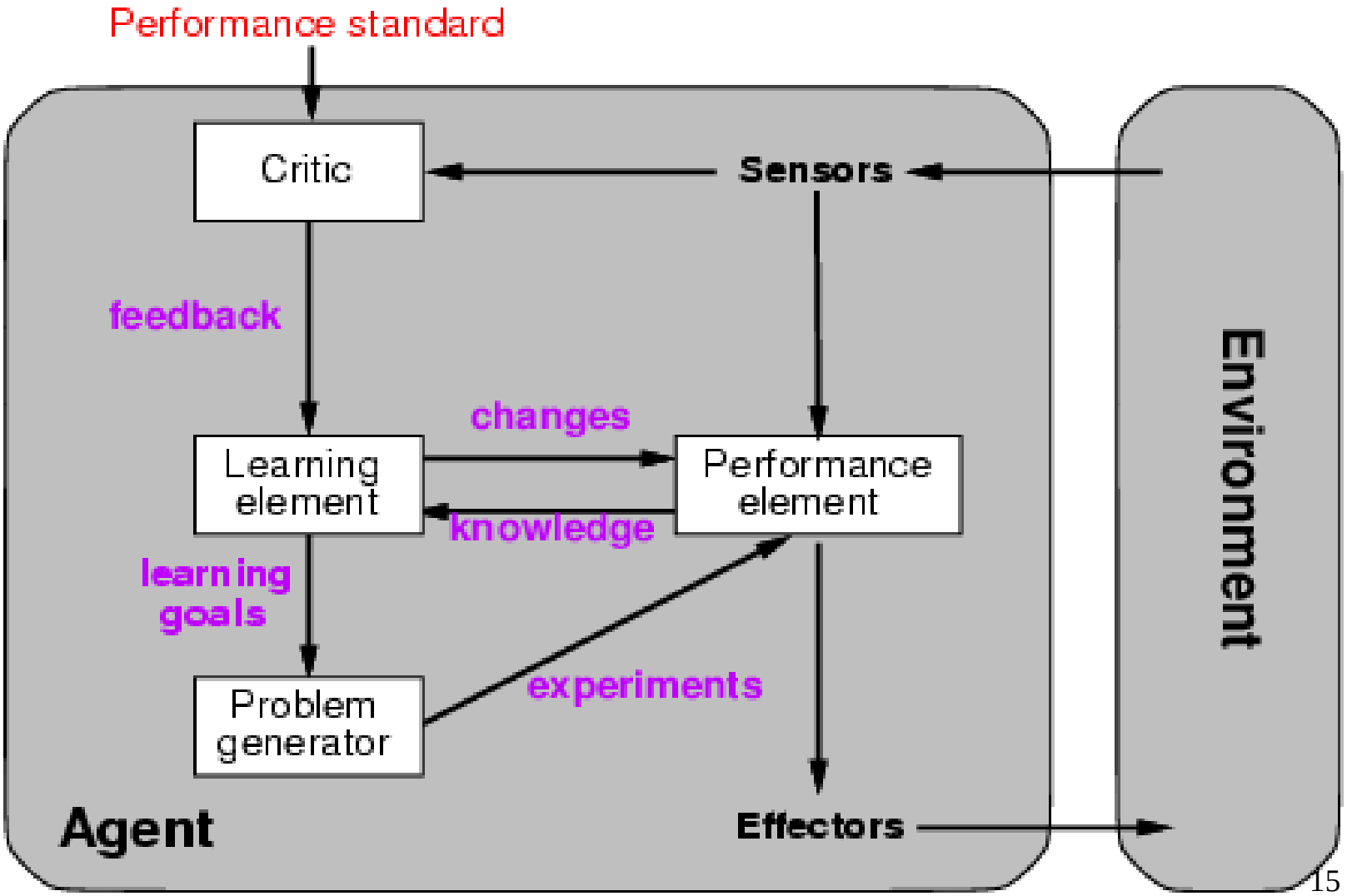


"Bad" features

The Basic Learning Model

- A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P,
  - if its performance at tasks T, as measured by P, improves with experience E.
- Learning agents consist of four main components:
  - **learning element --** the part of the agent responsible for improving its performance
  - **performance element --** the part that chooses the actions to take
  - **critic** – provides feedback for the learning element how the agent is doing with respect to a performance standard
  - **problem generator --** suggests actions that could lead to new, informative experiences (suboptimal from the point of view of the performance element, but designed to improve that element)

Learning Agents



Types of learning

- **Supervised learning:** occurs where a set of input/output pairs are explicitly presented to the agent by a teacher
  - The teacher provides a category label for each pattern in a training set, then the learning algorithm finds a rule that does a good job of predicting the output associated with a new input.
- **Unsupervised learning:** Learning when there is no information about what the correct outputs are.
  - In unsupervised learning or clustering there is no explicit teacher, the system forms clusters or natural groupings of the input patterns.

Supervised and Unsupervised

Supervised Classification = Classification

➔ We know the class labels and the number of classes

1 2 3 4 … n

Unsupervised Classification = Clustering

➔ We do not know the class labels and may not know the number of classes

1 2 3 4 … n

1 2 3 4 … ?

The Learning Problem

- Given <x,f(x)> pairs, infer f

x	f(x)
1	1
2	4
3	9
4	16
5	?

Given a finite sample, it is often impossible to guess the true function f.

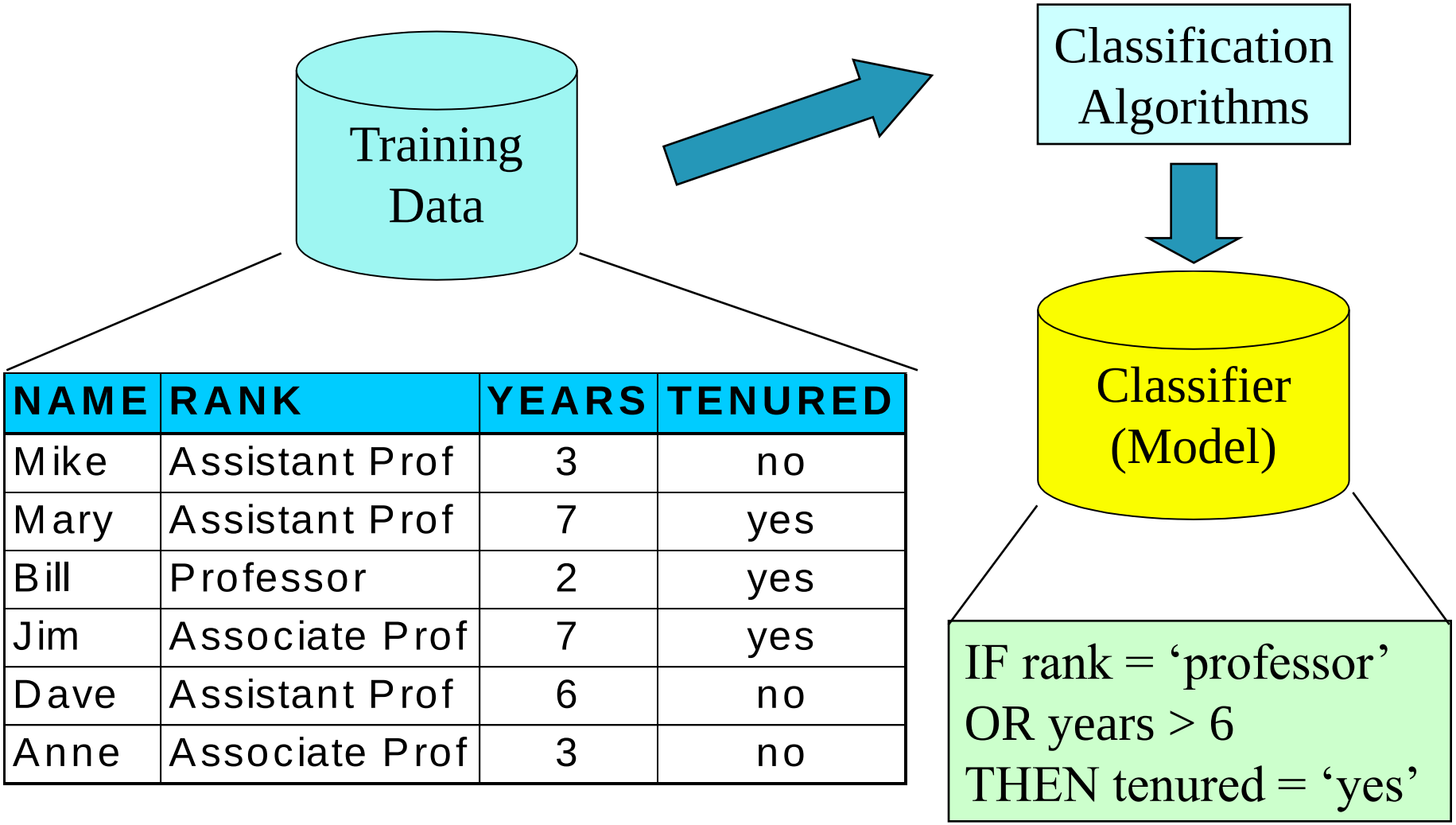
Approach: Find some pattern (called a ***hypothesis***) in the training examples, and assume that the pattern will hold for future examples too.

Data sets preparation for learning

- **Training set**
  - Used in supervised learning, a training set is a set of problem instances (described as a set of properties and their values), together with a classification of the instance.
- **Test set**
  - A set of instances and their classifications used to test the accuracy of a learned hypothesis.

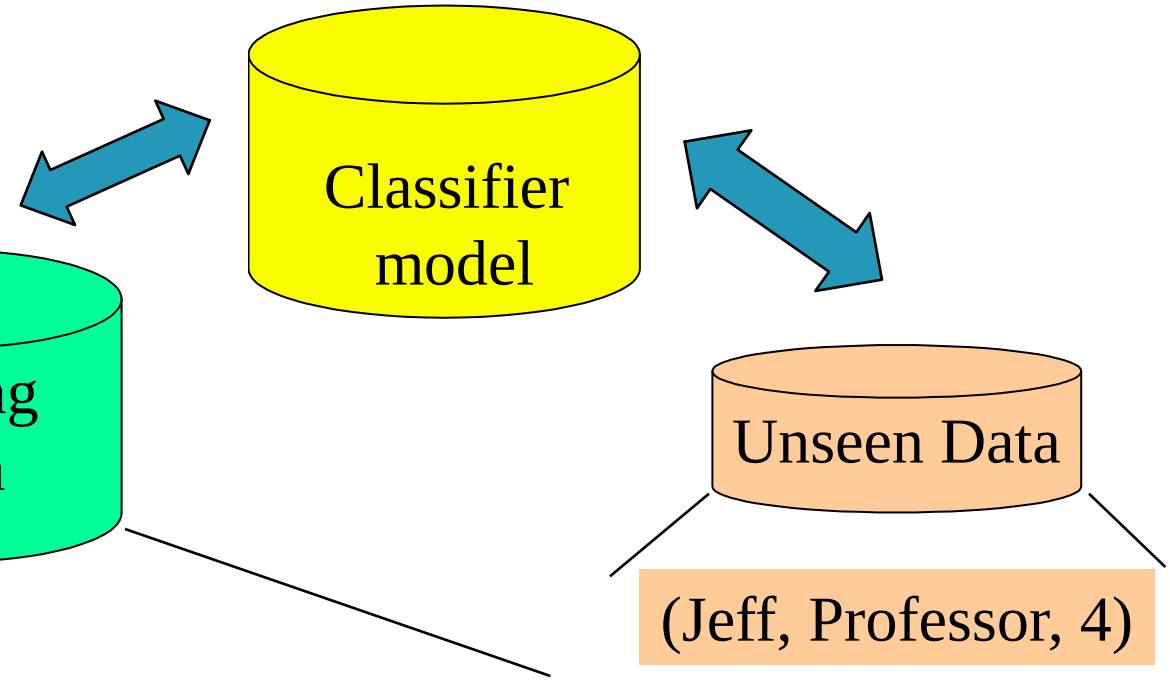
- Learning—A Two-Step Process
  - **Model construction:** The training set is used to create the model. The model is represented as classification rules, decision trees, or mathematical formulae
  - **Model usage:** the test set is used to see how well it works for classifying future or unknown objects

Step 1: Model Construction



NAME	RANK	YEARS	TENURED
Mike	Assistant Prof	3	no
Mary	Assistant Prof	7	yes
Bill	Professor	2	yes
Jim	Associate Prof	7	yes
Dave	Assistant Prof	6	no
Anne	Associate Prof	3	no

Step 2: Using the Model in Prediction



NAME	RANK	YEARS	TENURED
Tom	Assistant Prof	2	no
Merlisa	Associate Prof	7	no
George	Professor	5	yes
Joseph	Assistant Prof	7	yes

Learning methods

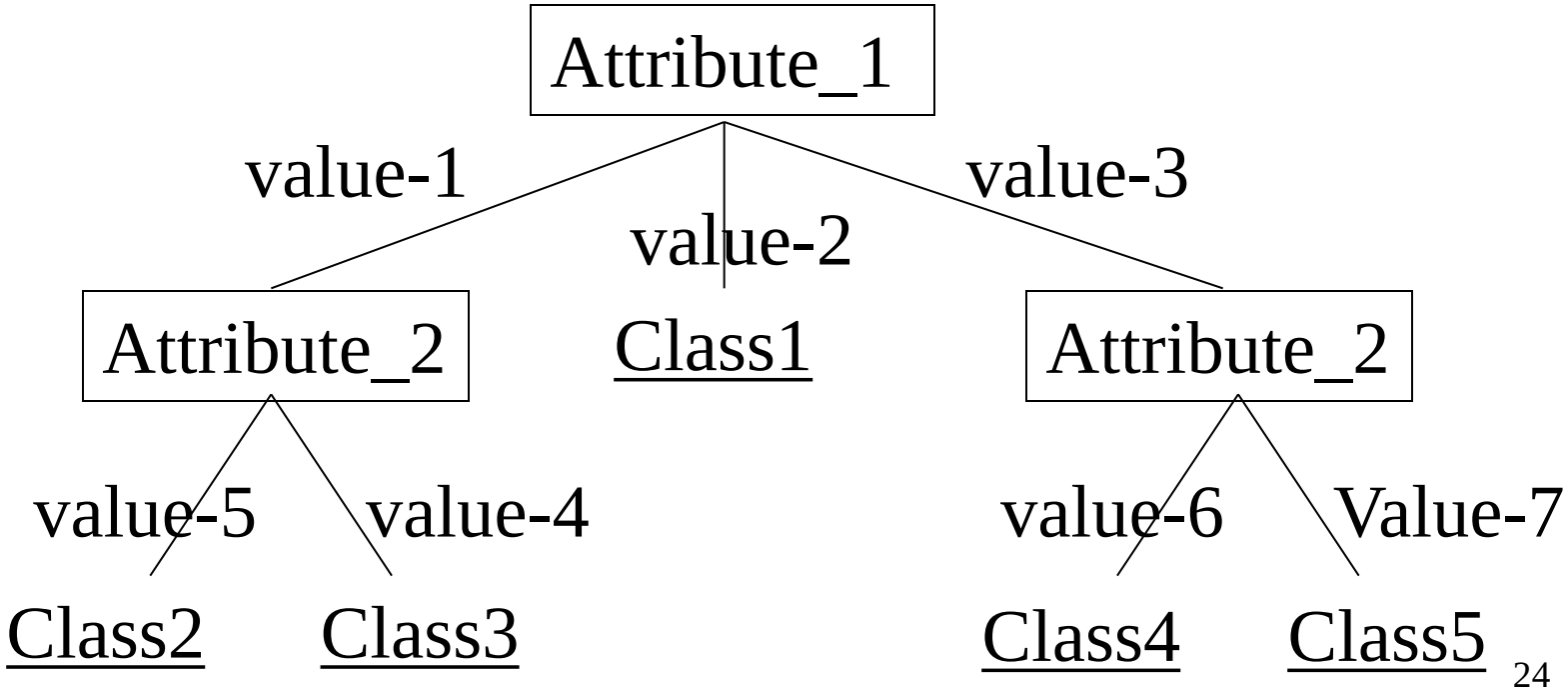
- There are various learning methods. Popular learning techniques include the following.
  - Decision tree : divide decision space into piecewise constant regions.
  - Neural networks: partition by non-linear boundaries
  - Bayesian network: a probabilistic model
  - Regression: (linear or any other polynomial)
  - Support vector machine
  - Expectation maximization algorithm

Decision tree

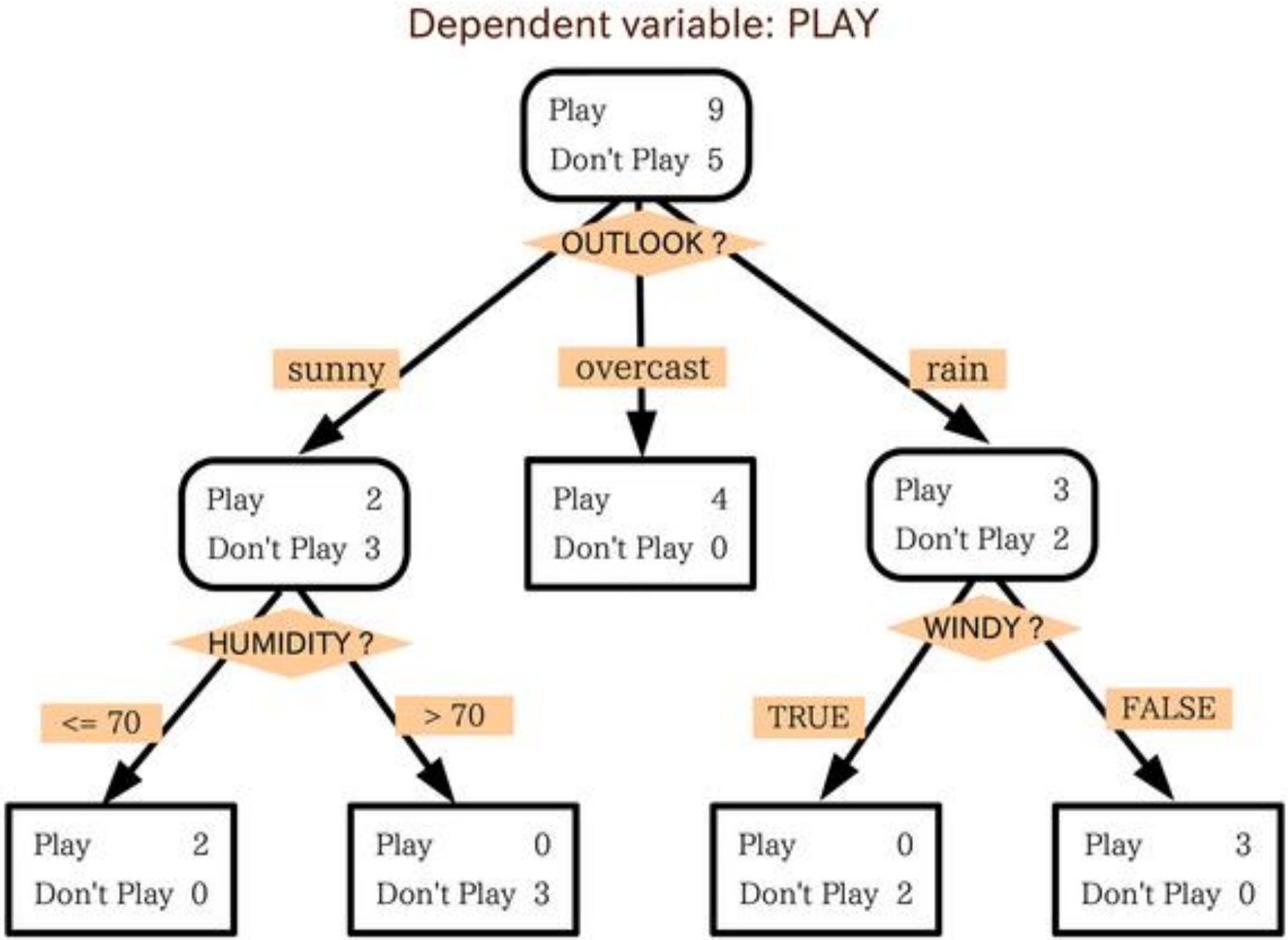
- *Decision tree* performs classification by constructing a tree based on training instances with leaves having class labels.
  - The tree is traversed for each test instance to find a leaf, and the class of the leaf is the predicted class. This is a directed knowledge discovery in the sense that there is a specific field whose value we want to predict.
- Widely used learning method. It has been applied to:
  - classify medical patients based on the disease,
  - equipment malfunction by cause,
  - loan applicant by likelihood of payment.
- Easy to interpret: can be re-represented as if-then-else rules
- Does not require any prior knowledge of data distribution, works well on noisy data.

Decision Trees

- Tree where internal nodes are simple decision rules on one or more attributes and leaf nodes are predicted class labels; i.e. a Boolean classifier for the input instance.
  - ✓ Given an instance of an object or situation, which is specified by a set of properties, the tree returns a "yes" or "no" decision about that instance.



Decision trees



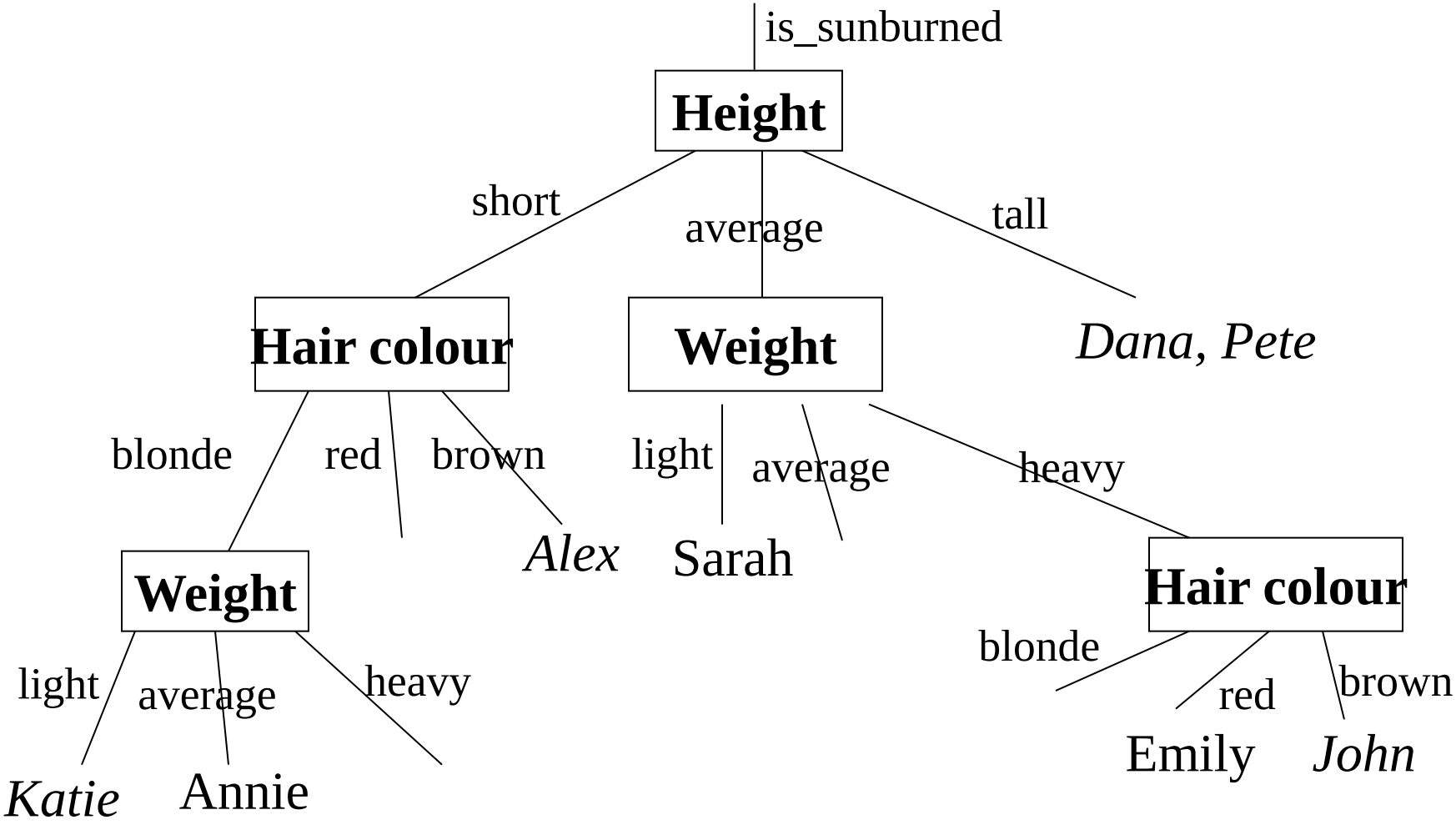
Example 1: The problem of "Sunburn"

- You want to predict whether another person is likely to get sunburned if he is back to the beach. How can you do this?
- Data Collected: predict based on the observed properties of the people

Name	Hair	Height	Weight	Lotion	class
Sarah	Blonde	Average	Light	No	Sunburned
Dana	Blonde	Tall	Average	Yes	*None*
Alex	Brown	Short	Average	Yes	*None*
Annie	Blonde	Short	Average	No	Sunburned
Emily	Red	Average	Heavy	No	Sunburned
Pete	Brown	Tall	Heavy	No	*None*
John	Brown	Average	Heavy	No	*None*
Kate	Blonde	Short	Light	Yes	*None*

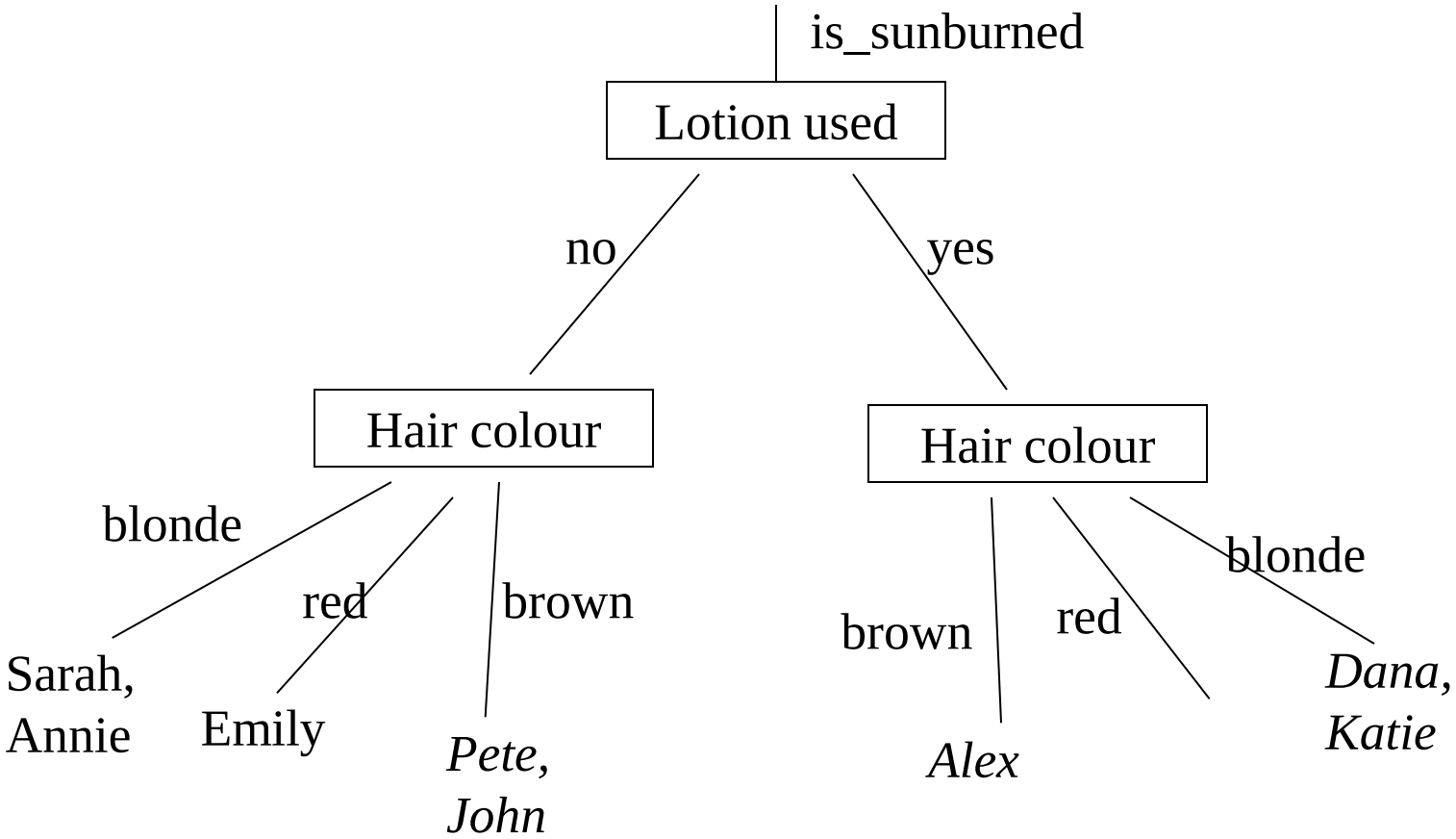
Decision Tree 1

- This is one way of looking at the data, starting with ‘Height’ property. The decision tree correctly CLASSIFIES all the data!



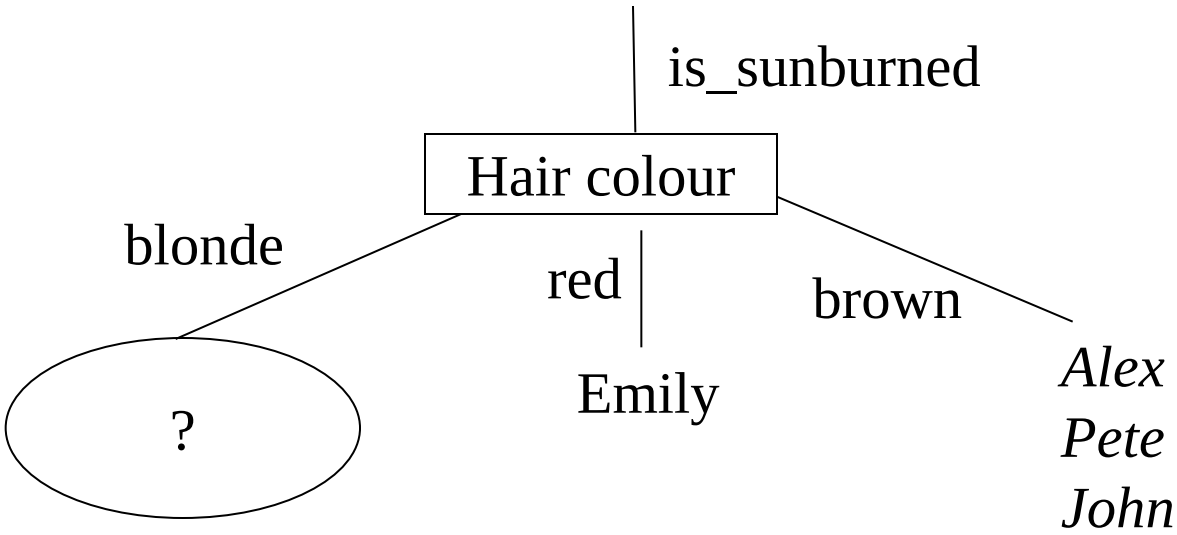
Decision Tree 2

- Is there any attributes you don't like to use for predicting whether people will suffer from sunburn? Of course it isn't reasonable to use height.



- This decision tree is a lot better than the first one and it doesn't involve any of the irrelevant attributes. Unfortunately though it uses the hair colour attribute twice which doesn't seem so efficient.

The best decision tree?



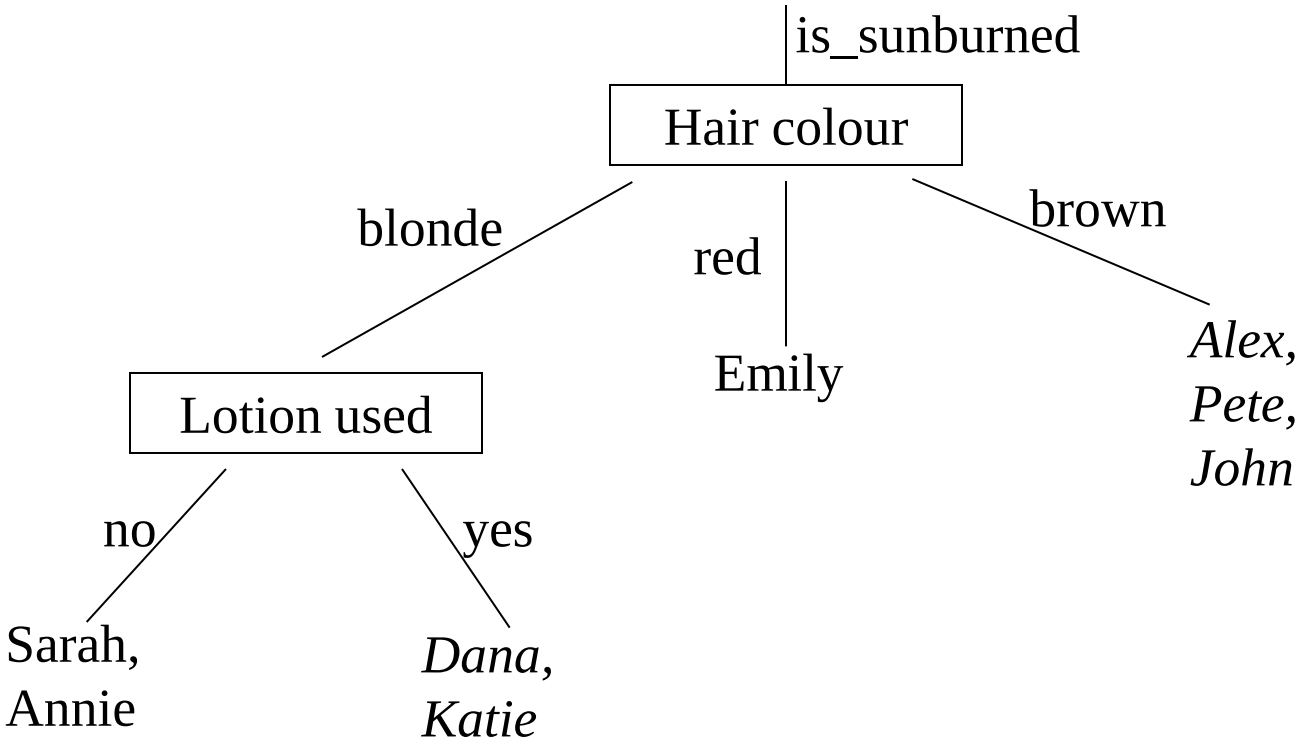
Sunburned = Sarah, Annie,

None = *Dana*, *Katie*

- Once we have finished with hair colour we then need to calculate the remaining branches of the decision tree.
- Which attributes is better to classify the remaining ?

The best Decision Tree

- This is the simplest and optimal one possible and it makes a lot of sense.
- It classifies 4 of the people on just the hair colour alone.



Sunburn sufferers are ...

- You can view Decision Tree as an IF-THEN_ELSE statement which tells us whether someone will suffer from sunburn.

```
If (Hair-Colour="red") then
                    return (sunburned = yes)
else if (hair-colour="blonde" and lotion-used="No") then
                    return (sunburned = yes)
else
                    return (false)
```

Example: Wait for a Table at a Restaurant

Problem: decide whether to wait for a table at a restaurant, based on the following attributes:

1. Alternate: is there an alternative restaurant nearby?
2. Bar: is there a comfortable bar area to wait in?
3. Fri/Sat: is today Friday or Saturday?
4. Hungry: are we hungry?
5. Patrons: number of people in the restaurant (None, Some, Full)
6. Price: price range ($, $$, $$$)
7. Raining: is it raining outside?
8. Reservation: have we made a reservation?
9. Type: kind of restaurant (French, Italian, Thai, Burger)
10. WaitEstimate: estimated waiting time (0-10, 10-30, 30-60, >60)

Attribute-based representations

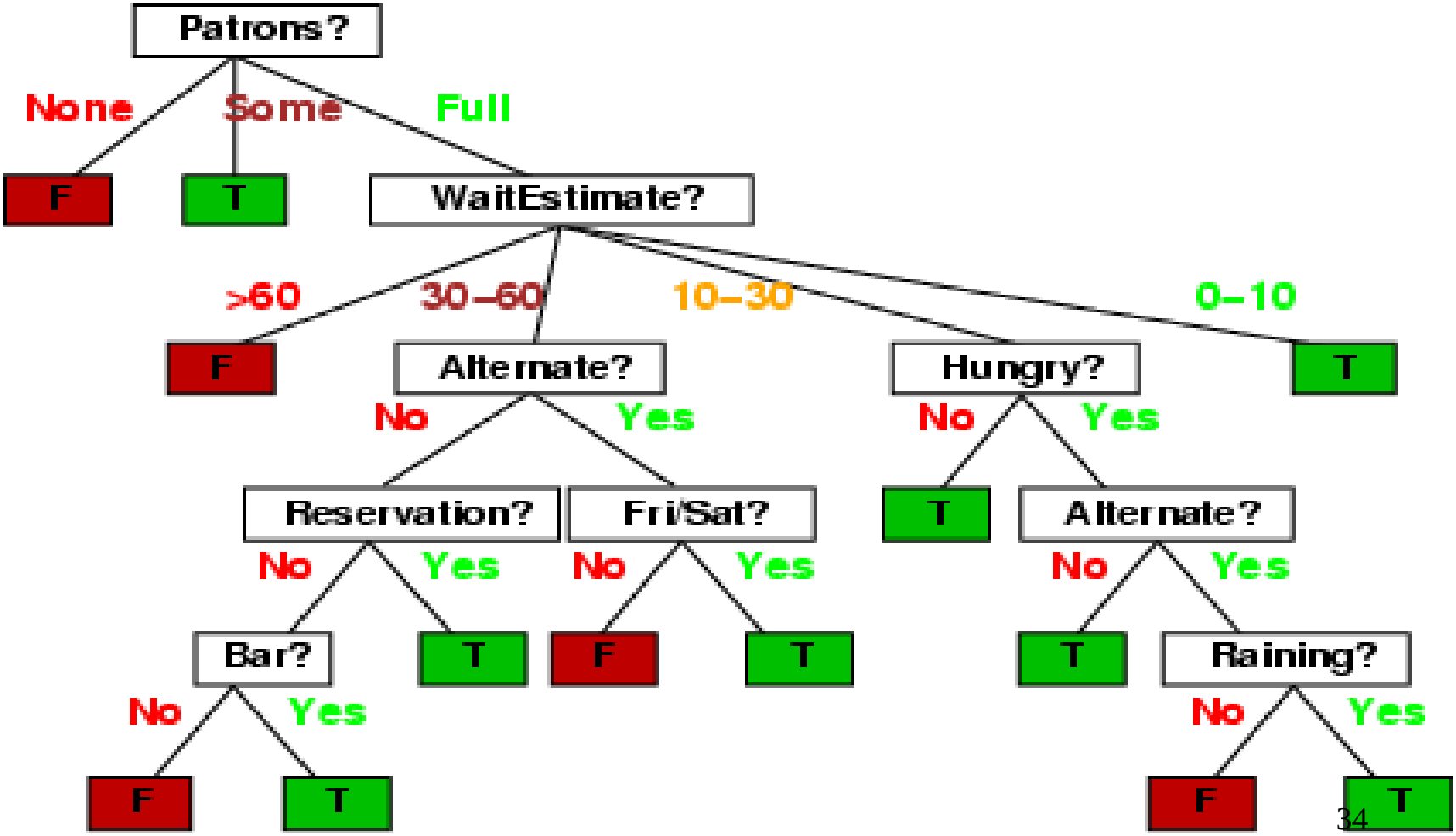
- Examples described by **attribute values** (Boolean, discrete, continuous)
- E.g., situations where I will/won't wait for a table:

Example	Attributes										Target
	Alt	*Bar*	*Fri*	*Hun*	*Pat*	*Price*	*Rain*	*Res*	*Type*	*Est*	*Wait*
X_1	T	F	F	T	Some	$$$	F	T	French	0–10	T
X_2	T	F	F	T	Full	$	F	F	Thai	30–60	F
X_3	F	T	F	F	Some	$	F	F	Burger	0–10	T
X_4	T	F	T	T	Full	$	F	F	Thai	10–30	T
X_5	T	F	T	F	Full	$$$	F	T	French	>60	F
X_6	F	T	F	T	Some	$$	T	T	Italian	0–10	T
X_7	F	T	F	F	None	$	T	F	Burger	0–10	F
X_8	F	F	F	T	Some	$$	T	T	Thai	0–10	T
X_9	F	T	T	F	Full	$	T	F	Burger	>60	F
X_{10}	T	T	T	T	Full	$$$	F	T	Italian	10–30	F
X_{11}	F	F	F	F	None	$	F	F	Thai	0–10	F
X_{12}	T	T	T	T	Full	$	F	F	Burger	30–60	T

- **Classification** of examples is **positiv**e (T) or **negative** (F)

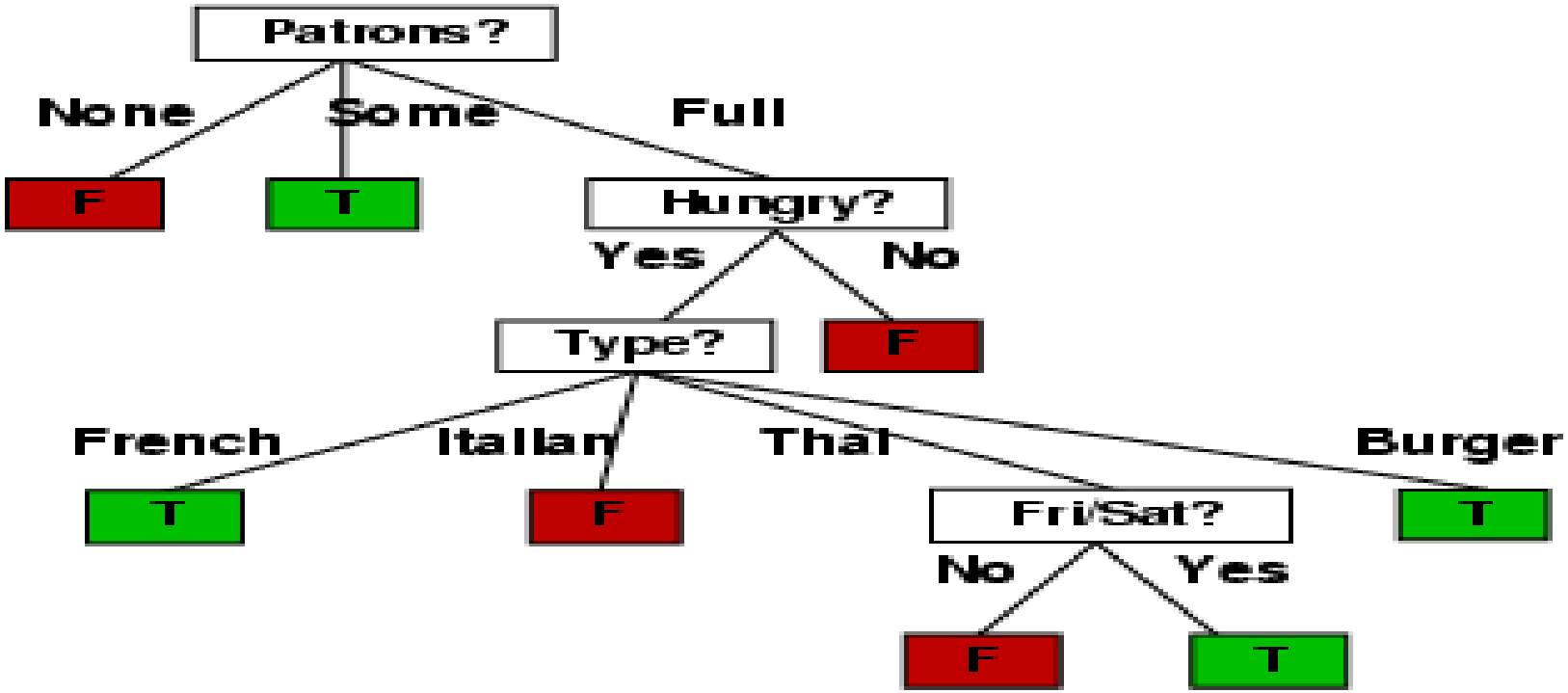
Learning Decision trees 1

- One possible representation for hypotheses
- E.g., here is the "true" tree for deciding whether to wait:



Learning decision trees 2

- Decision tree learned from the 12 examples:

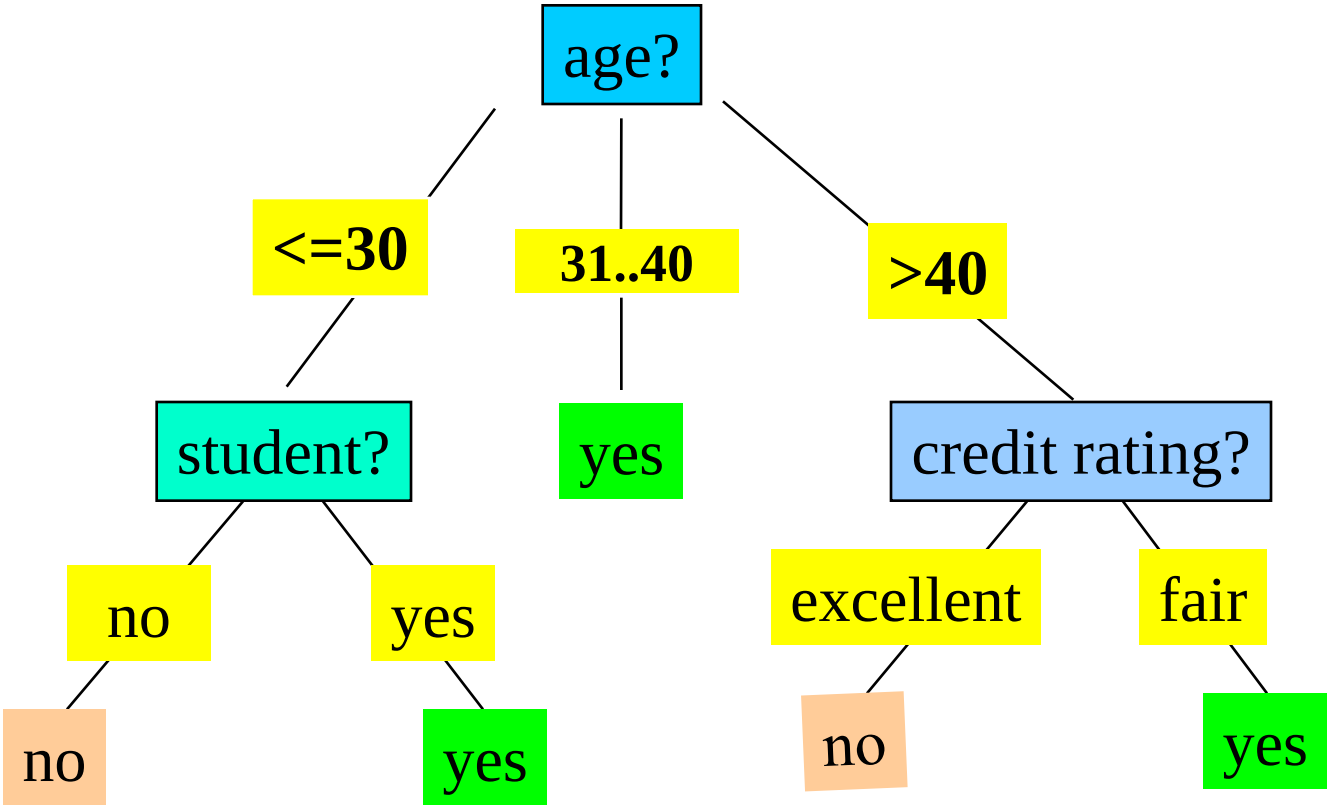


- Substantially simpler than “true” tree---a more complex hypothesis isn’t justified by small amount of data

Exercise: Decision Tree for “buy computer or not”. Use the training Dataset given below to construct decision tree

age	income	student	credit_rating	buys_computer
<=30	high	no	fair	no
<=30	high	no	excellent	no
31…40	high	no	fair	yes
>40	medium	no	fair	yes
>40	low	yes	fair	yes
>40	low	yes	excellent	no
31…40	low	yes	excellent	yes
<=30	medium	no	fair	no
<=30	low	yes	fair	yes
>40	medium	yes	fair	yes
<=30	medium	yes	excellent	yes
31…40	medium	no	excellent	yes
31…40	high	yes	fair	yes
>40	medium	no	excellent	no

Output: A Decision Tree for "*buys_computer*"



Pros and Cons of decision trees

· **Pros**

+ Reasonable training time
+ Fast application
+ Easy to interpret
+ Easy to implement
+ Can handle large number of features

· **Cons**

- Cannot handle complicated relationship between features
- simple decision boundaries
- problems with lots of missing data

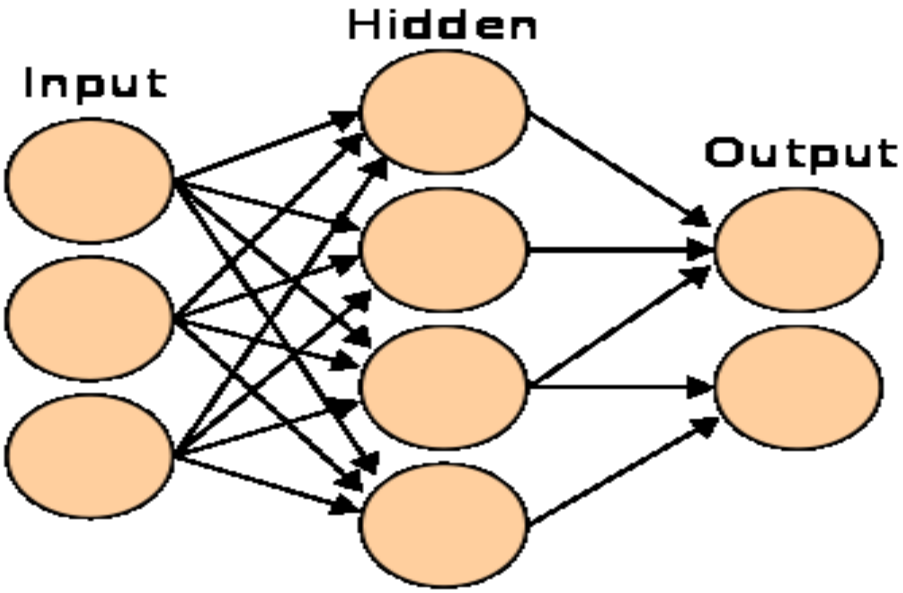
Why decision tree induction in data mining?

- relatively faster learning speed (than other classification methods)
- convertible to simple and easy to understand classification rules
- comparable classification accuracy with other methods

Neural Network

- It is represented as a layered set of interconnected processors. These processor nodes has a relationship with the neurons of the brain. Each node has a weighted connection to several other nodes in adjacent layers. Individual nodes take the input received from connected nodes and use the weights together to compute output values.

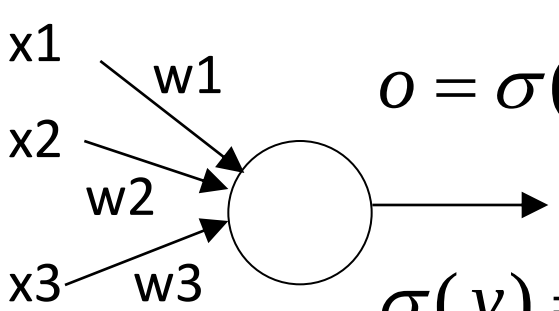
- The inputs are fed simultaneously into the input layer.
- The weighted outputs of these units are fed into hidden layer.
- The weighted outputs of the last hidden layer are inputs to units making up the output layer.



Architecture of Neural network

- Neural networks are used to look for patterns in data, learn these patterns, and then classify new patterns & make forecasts
- A network with the input and output layer only is called single-layered network. Whereas, a multilayer neural network is a generalized one with one or more hidden layer.
  - A network containing two hidden layers is called a three-layer neural network, and so on.

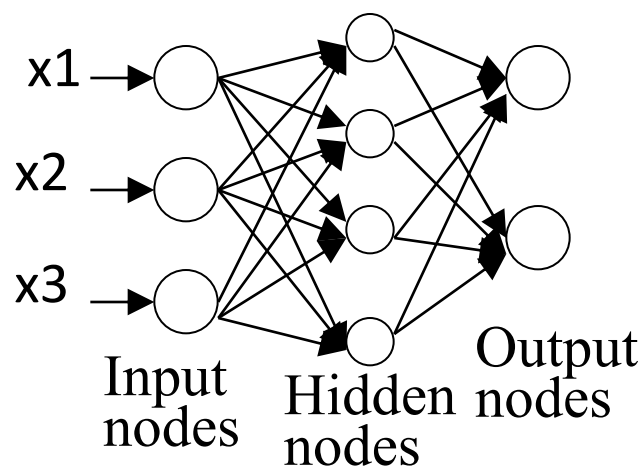
Single layered NN



$$o = \sigma\left(\sum_{i=1}^{n} w_i x_i\right)$$

$$\sigma(y) = \frac{1}{1+e^{-y}}$$

Multilayer NN



Hidden layer: Neuron with Activation

- The neuron is the basic information processing unit of a NN. It consists of:

1 A set of links, describing the neuron inputs, with weights $W_1, W_2, \ldots, W_m$

2. An adder function (linear combiner) for computing the weighted sum of the inputs (real numbers):

$$y = \sum_{j=1}^{m} w_j x_j$$

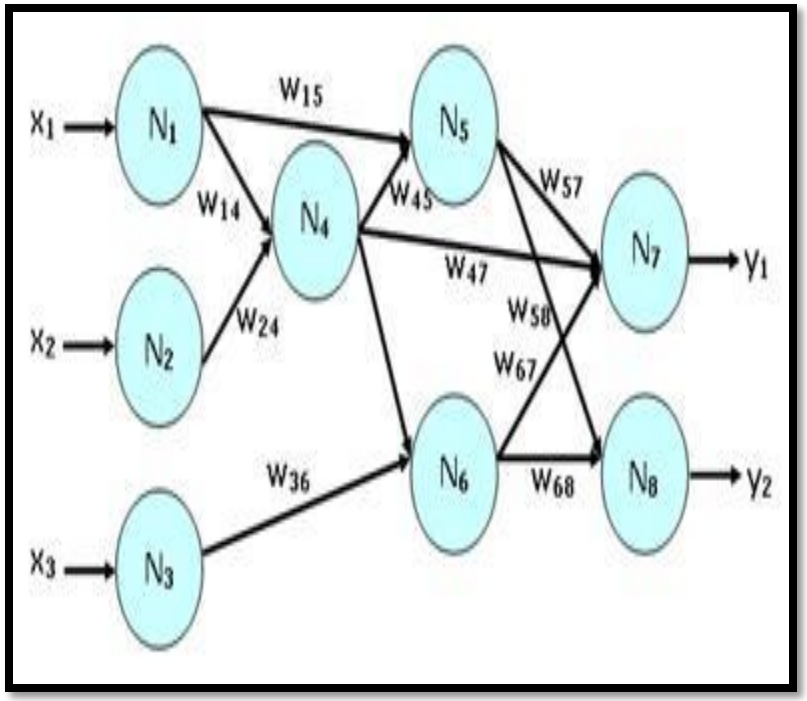
3. Activation function : for limiting the output behavior of the neuron.

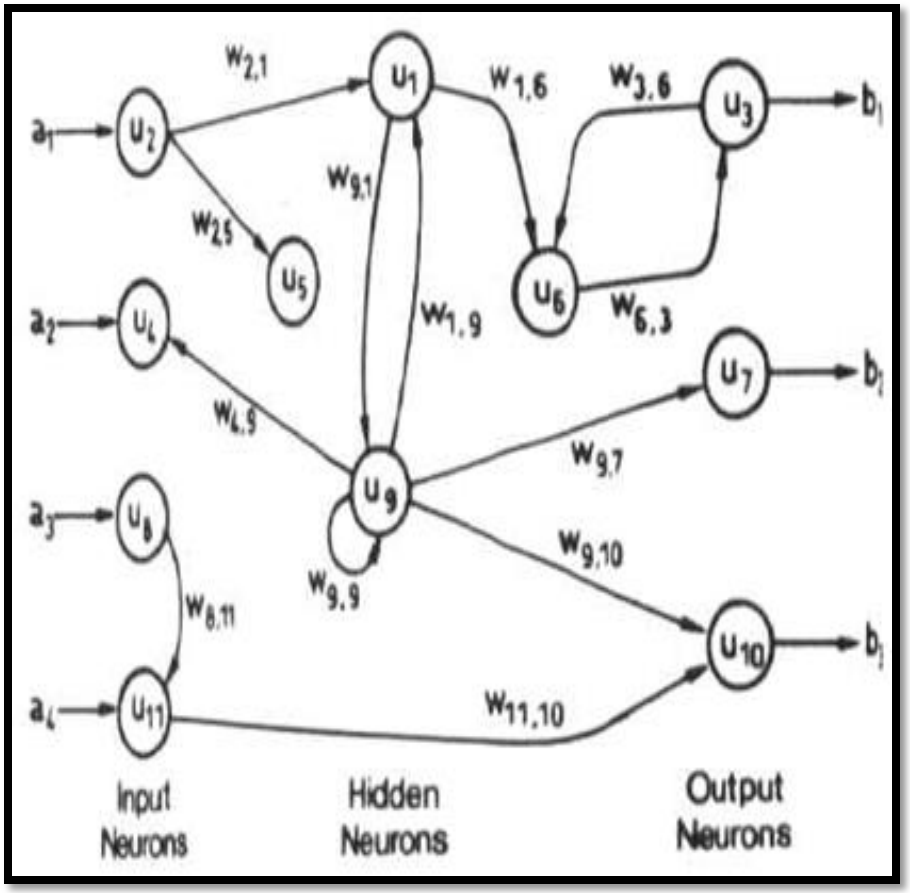
$$\sigma(y) = \frac{1}{1 + e^{-y}}$$

Topologies of neural network

- In a feed forward neural network connections between the units do not form a directed cycle.
  - In this network, the information moves in only one direction, forward, from the input nodes, through the hidden nodes (if any) and to the output nodes. There are no cycles or loops in the network.
- *In recurrent networks* data circulates back and forth until the activation of the units is stabilized
  - Recurrent networks have a feedback loop where data can be fed back into the input at some point before it is fed forward again for further processing and final output.
- In a feed forward NN the data processing can extend over multiple (layers of) units, but no feedback connections are present, that is, connections extending from outputs of units to inputs of units in the same layer or previous layers.

Topologies of neural network

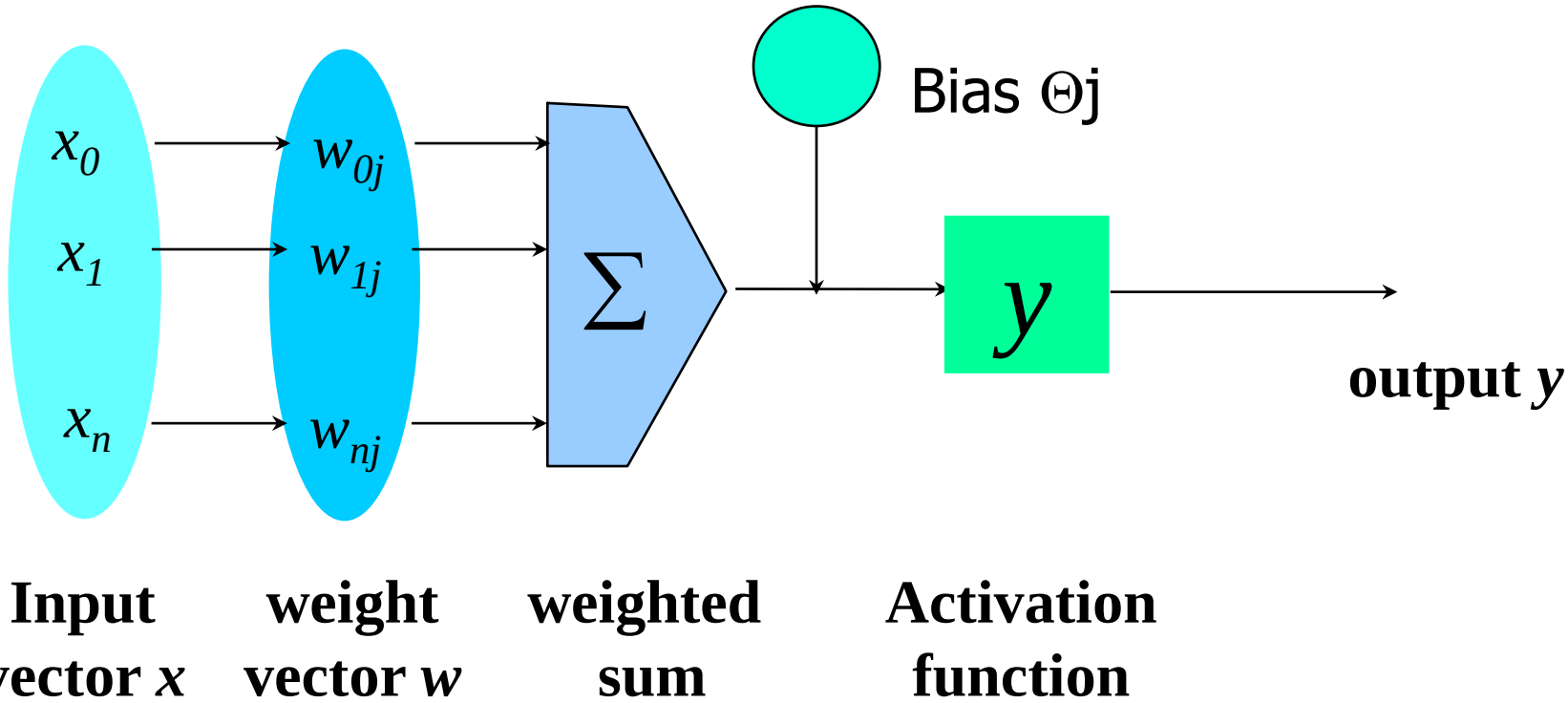




Training the neural network

- *Back Propagation* is the most commonly used method for training multilayer feed forward NN.
  - *Back propagation* learns by iteratively processing a set of training data (samples).
  - For each sample, weights are modified to minimize the error between the desired output and the actual output.
- After propagating an input through the network, the error is calculated and the error is propagated back through the network while the weights are adjusted in order to make the error smaller.

Propagation through Hidden Layer



- The inputs to unit j are outputs from the previous layer. These are multiplied by their corresponding weights in order to form a weighted sum, which is added to the bias associated with unit j.
- A nonlinear activation function f is applied to the net input.

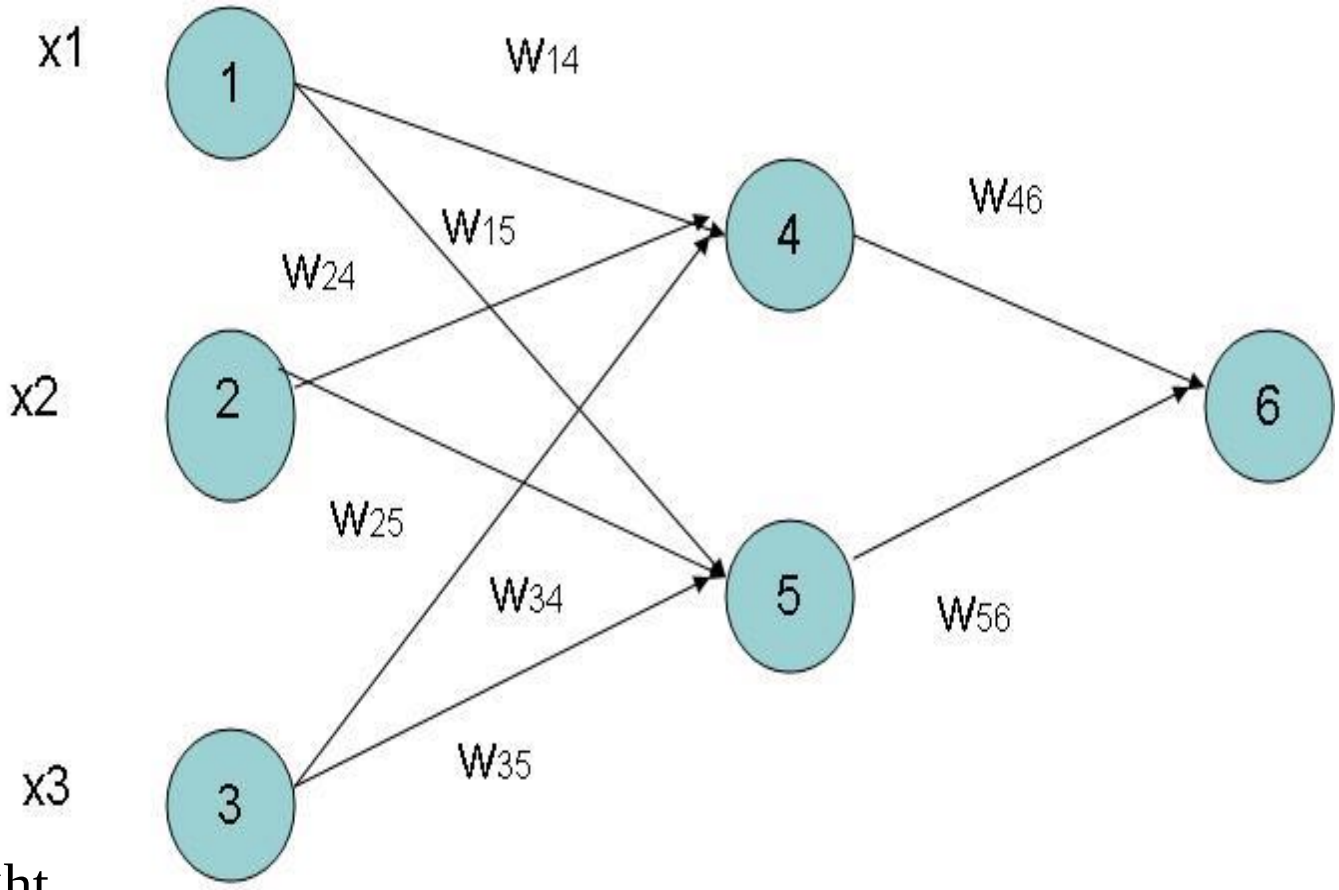
Steps in Back propagation Algorithm

- STEP ONE: initialize the weights and biases.
  - The weights in the network are initialized to random numbers from the interval [-1,1].
  - Each unit has a BIAS associated with it
  - The biases are similarly initialized to random numbers from the interval [-1,1].
- STEP TWO: feed the training sample.
- STEP THREE: Propagate the inputs forward; we compute the net input and output of each unit in the hidden and output layers.
- STEP FOUR: back propagate the error.
- STEP FIVE: update weights and biases to reflect the propagated errors.
- STEP SIX: terminating conditions.

Example of Back propagation

Input = 3, Hidden Neuron = 2, Output =1

Initialize weights : Random Numbers from -1.0 to 1.0



Initial Input and weight

x1	x2	x3	w14	w15	w24	w25	w34	w35	w46	w56
1	0	1	0.2	-0.3	0.4	0.1	-0.5	0.2	-0.3	-0.1

Pros and Cons of Neural Network

- Useful for learning complex data like handwriting, speech and image recognition

Pros

+ Can learn more complicated class boundaries
+ Fast application
+ Can handle large number of features

Cons

- Slow training time
- Hard to interpret
- Hard to implement: trial and error for choosing number of nodes

- Neural Network needs long time for training.
- Neural Network has a high tolerance to noisy and incomplete data

- Conclusion: Use neural nets only if decision-trees fail.