Mt Kenya University

P.O. Box 342-01000 Thika
Email: info@mku.ac.ke
Web: www.mku.ac.ke

DEPARTMENT OF INFORMATION TECHNOLOGY

BACHELOR OF BUSINESS INFORMATION TECHNOLOGY (BBIT)

COURSE CODE: BIT 3202

COURSE TITLE: ARTIFICIAL INTELLIGENCE

INSTRUCTIONAL MANUAL FOR BBIT – DISTANCE LEARNING

(Version 1)

PREPARED BY MERCY MAINA

BIT 3202 ARTIFICIAL INTELLIGENCE
Contact Hours 42
Pre Requisite **BIT1203: Basic Discrete Mathematics,**
BIT1101: Basic Physics and Optics

Purpose: To explore the idea that computers can be programmed to emulate "human-like" intelligence in Business Information Technology.

Objectives: By the end of the course unit a learner shall be able to:

- Give an overview of0 the main issues of Artificial Intelligence (AI) in Business Information Technology (BIT)
- Define the main sub-disciplines of AI in BIT.
- Explain AI techniques
- Apply knowledge representations techniques.
- Develop simple AI programmes.
- Apply methods used in AI programmes to make the programme behave intelligently in particular heuristic planning.

Course Content

- Introduction to artificial Intelligence; Definition of AI, Problems, Problem space and search
- Overview of AI application in areas of business
- Problem Solving, Searching and Controls Strategies
  - Problem Representation; Choosing a problem, Space
  - Search, search problem and closed world assumptions.
  - Search depth first and breath first techniques with back tracking.
  - Structures and strategies for state space search, heuristic search.
  - Introduction to semantic nets, rules, scripts, values and production systems.
- Knowledge Representation Network representation, Structured representations, Type hierarchies, Inheritance
- AI languages and prolog; Requirements for AI languages, Introduction to prolog, Understanding natural languages and speech recognition, AI programming language e.g. LISP/Prolog
  - Advanced representation in prolog.
- Machine learning; Definition and concepts, Learning models, Pattern recognition, Intelligence agents/learning agents
- AI systems in business specifically, ruled – based expert systems.
  - Learning and adapture systems
  - Vision image processing, scene analysis.
- Introduction to knowledge based and experimental systems.
- Introduction to AI game playing, planning and robotics and their application to business.

Teaching Methodology: Lectures, Tutorials, Computer laboratory exercises.

Instructional materials/Equipment; Audio visual aids in lecture rooms., Computer laboratory. Java language and development environment

Assessments: A learner is assessed through ; Continuous Assessment Tests (CATs) (30%), End of semester examination (70%)

Required text books

1. Winson P.H., Artificial Intelligence 3rd Ed, Addison Wesley
2. Prattt I., Artificial Intelligence, Macmillan press ltd
3. Bratko I., Programming in prolog 2nd Ed, Addison Wesley
4. Rich E & Knight K., Artificial Intelligence, McGraw Hill

Text books for further reading

1. Robert S.F, Artificial Intelligence, an Engineering Approach, McGraw Hill
2. Jones J. L. && Flynn A. M., Mobile Robots : Inspiration to implementation
3. Russell S.J & Norvig, Artificial Intelligence; A modern approach, Prentice Hall
4. O'keefe. The craft of prolog, MIT press Other support materials Various applicable manuals and journals, Variety electronic information resources as prescribed by the lecturer.

TABLE OF CONTENT

CHAPTER ONE

Over View of Artificial Intelligence

Chapter Objectives

By the end of this chapter the learner should be able to;

- Define Artificial Intelligence (AI).
- Explain the Goals of AI,
- Describe the AI Approaches
- Explain AI Techniques
- Describe the Branches of AI
- Explain the Applications of AI.

1.1. What is Artificial Intelligence (AI) and Intelligence

According to McCarthy (1956), Artificial Intelligence is the science and engineering of making intelligent machines, especially intelligent computer programmes.

Artificial intelligence (AI) is a broad field of computing that tries to understand the human intelligence and using that understanding build agents or entities that can act intelligently.

Artificial Intelligence (AI) could be defined as the ability of computer software and hardware to do those things that we, as humans, recognize as intelligent behavior. Traditionally those things include such activities as:

- Searching: finding "good" material after having been provided only limited direction, especially from a large quantity of available data.
- Surmounting constraints: finding ways that something will fit into a confined space, taking apart or building a complex object, or moving through a difficult maze.
- Recognizing patterns: finding items with similar characteristics, or identifying an entity when not all its characteristics are stated or available.
- Making logical inferences: drawing conclusions based upon understood reasoning methods such as deduction and induction.

A sophisticated technology is then a cumulative building of learned and well-refined skills and processes. In the AI area, these processes have manifested themselves in a number of well-recognized and maturing areas including Neural Networks, Expert Systems, Automatic Speech Recognition, Genetic Algorithms, Intelligent Agents, Natural Language Processing, Robotics, Logic Programming, and Fuzzy Logic.

The importance of these individual areas has changed over the last two decades. These changes have been based upon the progress in each area, and the needs that each area meets. For example in the early 1980's robotics was a large thrust in artificial intelligence. At that time benefits could be seen in manufacturing applications. In the late 1990's the blossoming of the Internet pushed the importance of intelligent agents forward for performing routine tasks and complex searches. At the same time, throughout the 1980s and 1990s, orders of magnitude advances in computer processing power have allowed hurdles in speech recognition and image processing to be overcome.

AI is also defined as getting of computers to do things that seem to be intelligent. The hope is that more intelligent computers can be more helpful to us-better able to respond to our needs and wants, and more clever about satisfying them.

However, having said that, there are many tasks which we might reasonably think require intelligence - such as complex arithmetic - which computers can do very easily. Conversely, there are many tasks that people do without even thinking - such as recognising a face - which are extremely complex to automate. AI is concerned with these difficult tasks, which seem to require complex and sophisticated reasoning processes and knowledge.

People might want to automate human intelligence for a number of different reasons:-

1. To understand human intelligence better. For example, we may be able to test and refine psychological and linguistic theories by writing programs which attempt to simulate aspects of human behaviour.
2. Another reason is simply so that we have smarter programs. We may not care if the programs accurately simulate human reasoning, but by studying human reasoning we may develop useful techniques for solving difficult problems.

From literature the following are some definition of AI.

1. The exciting new effort to make computers think . . . machines with minds, in the full and literal sense" (Haugeland, 1985)
2. "[The automation of] activities that we associate with human thinking, activities such as decision-making, problem solving, learning ..."(Bellman, 1978)
3. The study of mental faculties through the use of computational models" (Charniak and McDermott, 1985)
4. "The study of the computations that make it possible to perceive, reason, and act" (Winston, 1992)
5. "The art of creating machines that perform functions that require intelligence when performed by people" (Kurzweil, 1990).
6. "The study of how to make computers do things at which, at the moment, people are better" (Rich and Knight, 1 991)
7. "A field of study that seeks to explain and emulate intelligent behavior in terms of computational processes" (Schalkoff, 1 990)
8. "The branch of computer science that is concerned with the automation of intelligent behavior" (Luger and Stubblefield, 1993)

1.2. Artificial Intelligence Terms

1.2.1. Intelligence

Intelligence relates to tasks involving mental processes, e.g: Creating, solving problems, pattern recognition, classification, learning etc. Intelligence is the computational part of the ability to achieve goals.

1.2.2. Intelligent behaviour

Turing defined intelligent behavior as the ability to achieve human-level performance in all cognitive tasks, sufficient to fool an interrogator. Characteristics of an intelligent agent;

1. Perceiving one's environment
2. Acting in complex environment
3. Learning and understanding from experience.
4. Reasoning to solve problems and discover hidden knowledge
5. Knowledge applying successfully in new situations
6. Thinking abstractly, using analogies
7. Communication with others
8. Creativity e.t.c

1.2.3. "Hard" or "Strong" Artificial Intelligence

Generally, artificial intelligence research aims to create AI that can replicate human intelligence completely.

"Strong AI" refers to a machine that approaches or supersedes human intelligence,

- if it can do typically human tasks,
- if it can apply a wide range of background knowledge and
- if it has some degree of self-consciousness.

"Strong AI" aims to build machines whose overall intellectual ability is indistinguishable from that of a human being.

1.2.4. "Soft " or "Weak" Artificial Intelligence

"Weak AI" refers to the use of software to study or accomplish specific problem solving or reasoning tasks that do not encompass the full range of human cognitive abilities.

Example : a chess program such as "Deep Blue".

"Weak AI" does not achieve

- self-awareness, or
- demonstrate a wide range of human-level cognitive abilities, and
- at best is merely an intelligent, a more specific problem-solver.

1.3. Goals of AI

AI is a field that overlaps with computer science rather than being a strict subfield. Different areas of AI are more closely related to psychology, philosophy, logic, linguistics, and even neurophysiology. The goal of AI is to develop computers that can think, as well as see, hear, walk, talk, and feel. A major thrust of AI is the development of computer functions normally associated with human intelligence, such as reasoning, learning, and problem solving. From the definition of AI; there are four possible goals;-

(A) Systems that think like humans	(B). Systems that think rationally
The exciting new effort to make computers think . . . machines with minds, in the full and literal sense" (Haugeland, 1985) "[The automation of] activities that we associate with human thinking, activities such as decision-making, problem solving, learning ..."(Bellman, 1978)	The study of mental faculties through the use of computational models" (Charniak and McDermott, 1985) "The study of the computations that make it possible to perceive, reason, and act" (Winston, 1992)
(C) Systems that act like humans	(D). Systems that act rationally
"The art of creating machines that perform functions that require intelligence when performed by people" (Kurzweil, 1990) "The study of how to make computers do things at which, at the moment, people are better" (Rich and Knight, 1 991)	"A field of study that seeks to explain and emulate intelligent behavior in terms of computational processes" (Schalkoff, 1 990) "The branch of computer science that is concerned with the automation of intelligent behavior" (Luger and Stubblefield, 1993)

Table 1. AI Categorization

These definitions vary along two main dimensions.

- The definitions on the top, (A) and (B) are concerned with reasoning, whereas those on the bottom, (C) and (D) address behavior.
- The definitions on the left, (A) and (C) measure success in terms of human performance, whereas those on the right, (B) and (D) measure ideal concept of intelligence called rationality.

A system is rational if it does the right thing.

General AI goals

- Replicate human intelligence
- Solve knowledge intensive task.
- Make an intelligent connection between perception and action.
- Enhance human-human, human-computer and computer to computer interaction/ communication

Engineering based AI Goals

- Develop concepts, theory and practice of building intelligent machines.
- Emphasis is on system building

Science based AI goals

- Develop concepts, mechanisms and vocabulary to understand biological intelligent behaviour.
- Emphasis is on understanding intelligent behaviour.

1.4. Approaches to Artificial Intelligence

The field of artificial intelligence (AI) has as its goal the development of machines that can perceive, reason, communicate, and act in complex environments much like humans can, or possibly even better than humans can. Even though the field has produced some practically useful machines with rudiments of these abilities, it is generally conceded that the ultimate goal is still distant.

The approaches used in AI are based on the goals of the computational, model, and the basis for evaluating performance of the system.

	Human Being	Rationally
Think	(1) Cognitive science approach	(2) Laws of thought approach
Act	(3) Turing test approach	4) Rational agent approach

1. Cognitive Science: Think human-like
   - Aims at making computers think; That is, the machine with minds, in full and literal sense.
   - Focus is not on behaviour and I/O, but looks at reasoning process.
   - Aims to develop, explore and evaluate theories of how the human mind work through the use of environment models.
   - It explore how things are done but not what is done. This means intelligent behaviour is not enough, the program must operate in an intelligent manner.
   - The goal is not just to produce human-like behaviour but to produce a sequence of steps of the reasoning process, similar to the steps followed by a human in solving the same task.
   - Examples: The chess program can play chess but know little about how human being play chess.

2. Laws of Thought: Think Rationally
   - Studies mental faculties through the use of computational models: It studies the computations that make it possible to perceive, reason and act.
   - Focus on inference mechanism that are provably correct and guarantee an optimal solution.
   - Develop systems of representation to allow inferences to be like "Rose is a Computer science student. All computer science students are programmers, Therefore Rose is a Programmer".
   - The issue is, not all problems can be solved by just reasoning and inference.
   - The goal is to formalize the reasoning process as a system of logical rules and procedures for inference.

(3) Turing test approach

- The art of creating machines that perform function requiring intelligence when performed by people; That is the study of how to make machines do things which at the moment people do better.
- Focus is on action not on intelligent behavior centered around representation of the world.
- A behavioral approach, is not concerned with how to get the results but to the similarity to what human results are.
- Goal is to develop systems that are human-like

Example

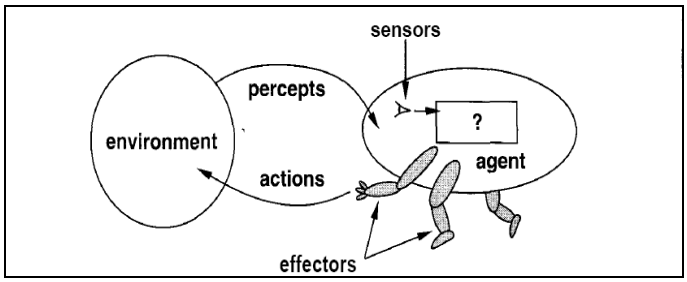
- 3 rooms contain: a person, a computer, and an interrogator.
- The interrogator can communicate with the other 2 by teletype (to avoid the machine imitate the appearance or voice of the person).
- The interrogator tries to determine which is the person and which is the machine.
- The machine tries to fool the interrogator to believe that it is the human, and the person also tries to convince the interrogator that it is the human.
- If the machine succeeds in fooling the interrogator, then conclude that the machine is intelligent.

4) Rational agent approach

Act Rationally.

- Tries to explain and emulate intelligent behavior in terms of computational processes; that is it is concerned with the automation of intelligence.
- Focus on systems that act sufficiently if not optimally in all situations.
- It is passable to have imperfect reasoning if the job gets done.
- Goal is to develop systems that are rational and sufficient.

A rational agent is one that does the right thing.



Agents interact with environment through sensors and effectors

1.5. Artificial Intelligence Techniques

Various techniques have evolved that can be applied to a variety of AI tasks. These techniques are concerned with how we represent, manipulate and reason with knowledge in order to solve problems. Two categories of techniques;

1. Techniques, not all intelligent but used to behave as intelligent

a) Describe and match	d) Goal reduction
b) Constraint satisfaction	e) Tree Searching
c) Generate and test	f) Rule based systems

2. Biology-Inspired AI Techniques

a) Neural Networks
b) Reinforcement learning
c) Genetic Algorithms

1.5.1. Intelligent Techniques

Describe and Match

- Model is a description of system behavior.
- Finite state model consists of a set of states, a set of input event and the relations between them. Given a current state and an input event, you can determine the next current state of the model.
- Computational model is a finite state machine. It includes a set of states, a set of start state, an input alphabet, and a transition.
- Representation of the computation system include start and end state.
- Transition relation: if a pair of states (S, S') is such that one move takes the system for S to S', then the transition is represented by S =) S0
- State-transition system is called deterministic if every state has a set at most one successor. it is called non-deterministic if at least one state has more than on successor.

Constraint Satisfaction

- Constraint is a logical relation among variables, e.g. "circle is inside the square" – The constraints relate objects without precisely specifying their positions; moving any one, the relation is still maintained.
- Constraint satisfaction is the process of finding a solution to a set of Constraints, – the constraints express allowed values for variables and finding solution is evaluation of these variables that satisfies all constraints.

Constraint Satisfaction Problem (CSP) and Its solution.

- Constraint satisfaction Problem consist of:
  a) Variables, a finite set X = (x_1….. x_n)
  b) Domain, a finite set of possible values which each variable x1 can take.
  c) Constraints , a set of values that a variable can simultaneously satisfy in the constraints.
- A solution to a CSP is an assignment of a value from some domain to every variable satisfying every constraint, that could be
  a) One solution, with no preference as which one.
  b) All solutions,
  c) An optimal, or a good solution- Constraint Optimization problem (COP)

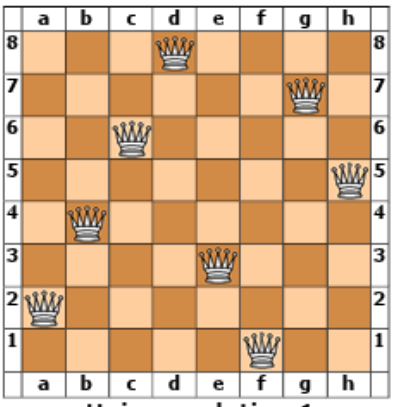
- Constraint satisfaction has application in Artificial Intelligence, Programming Languages, Symbolic Computing, Computational Logic. Examples

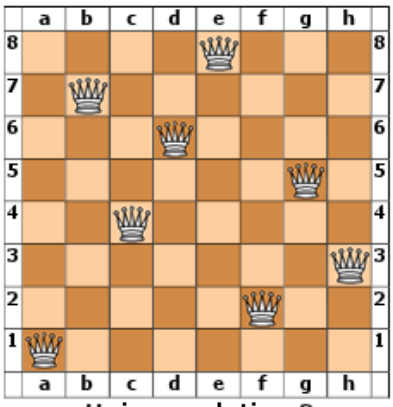
N-Queens puzzle :

Problem : Given any integer N, place N queens on N*N chessboard satisfying constraint that no two queens threaten each other (a queen threatens other queens on same row, column and diagonal).

Solution: To model this problem :

1) Assume that each queen is in different column;
2) Assign a variable Ri (i = 1 to N) to the queen in the i-th column indicating the position of the queen in the row.
3) Apply "no-threatening" constraints between each couple Ri and Rj of queens and evolve the algorithm.
   Example : 8 - Queens puzzle

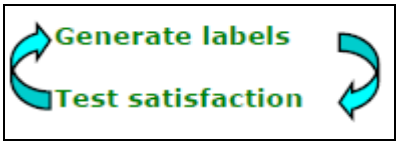

Unique solution 1


Unique solution 2

The eight queens puzzle has 92 distinct solutions. If solutions that differ only by symmetry operations (rotations and reflections) of the board are counted as one, the puzzle has 12 unique solutions, the two them are presented above.

Generate and Test (GT)

- Most algorithms for solving CSP search systematically through possible assignment of values.
  a) CSP algorithm guarantee to fins solution if one exists or to provide that the problem is unsolvable.
  b) The disadvantage is that they take a very long time to do so
- Generate and test method: It first guesses the solution and then test whether this solution is correct, means solution satisfies the constraint.
- This paradigm involves two processes: generate to enumerate possible solutions and test to evaluate each possible solution
- The algorithm is



Generate and test paradigm

Disadvantage of Generate and Test method

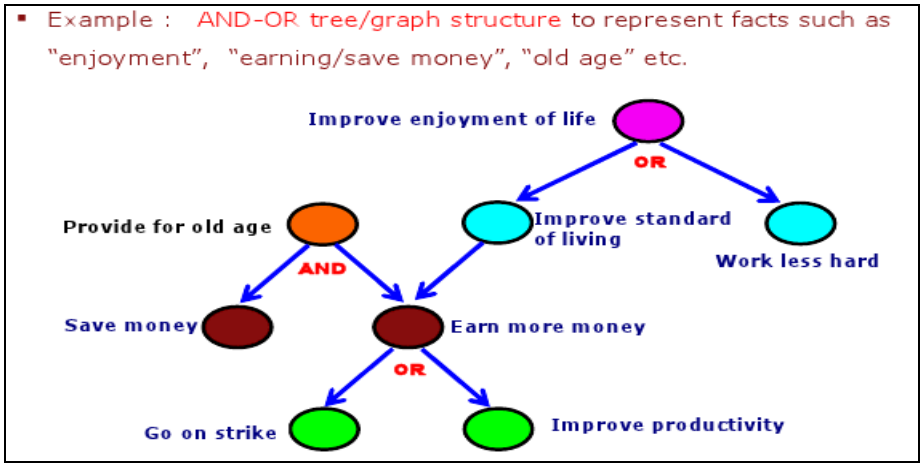
- Not very sufficient, generate many wrong assignments of values to variables which are rejected in the testing phase.
- Generate leaves conflicting instantiations and it generates other assignments independently of the conflict.
- For better efficiency GT approach need to be supported by backtracking approach.

Example: Opening a combination lock without knowing the combination.

Goal Reduction

- Goal reduction procedures are special case of the procedural representation of knowledge in AI.
- The process involves the Hierarchical sub-division of goals into sub goals, until the sub-goals which have an immediate solution are reached and are said goal has been satisfied.
- Goal- reduction process is illustrated in form a tree drawn upside down

- Goal levels: Higher-level goals are higher in the tree and lower-level goals are lower in the tree.
- Arcs are directed from a higher-to- lower level of node represent the reduction of the higher-level to lower level sub goal.
- Node at the bottom of the tree represent irreducible action goals
- A tree/graph structure can represent relations between goals and sub-goals, alternative sub-goals and conjoint sub-goals.
- Goal-reduction process is illustrated in the form of "AND/OR" tree drawn upside-down as shown below;



The above AND-OR tree/graph structure describes:

- Hierarchical relationships between goals and sub-goals: e.g.: "going on strike" is a sub-goal of "earning more money", is a sub-goal of "improving standard of living ", is a sub-goal of "improving enjoyment of life".
- Alternative ways of trying to solve a goal: e.g.: "going on strike" and "increasing productivity" are alternative ways of trying to "earn more money" (increase pay).e.g.: "improving standard of living" and "working less hard" are alternative ways of trying to "improve enjoyment of life".
- Conjoint sub-goals: e.g.: to "provide for old age", not only need to "earn more money", but as well need to "save money".

Tree Searching

Many problems can be described in the form of a search tree. A solution to the problem is obtained by finding a path through this tree. A search through the entire tree, until a satisfactory path is found, is called exhaustive search.

Tree search strategies

a) Depth-First search
b) Hill climbing
c) Breadth-first search
d) Beam search
e) Best-first search.

a). Depth-First search

The Depth-First search assumes any path is as good as any other path. At each node, pick an arbitrary path and work forward until solution is found or a dead end is reached. In the case of a dead end- back track to the last node in the three where a previously unexplored path branches of, and test this path. Backtracking can be of two types:

- Chronological Backtracking: undo everything as we move back up the tree to a suitable node.
- dependence directed backtracking: only withdraw choices that matter (those on which dead end depends)

b). Hill Climbing: Like depth first but involving some quantitative decision on the most likely path to follow at each node

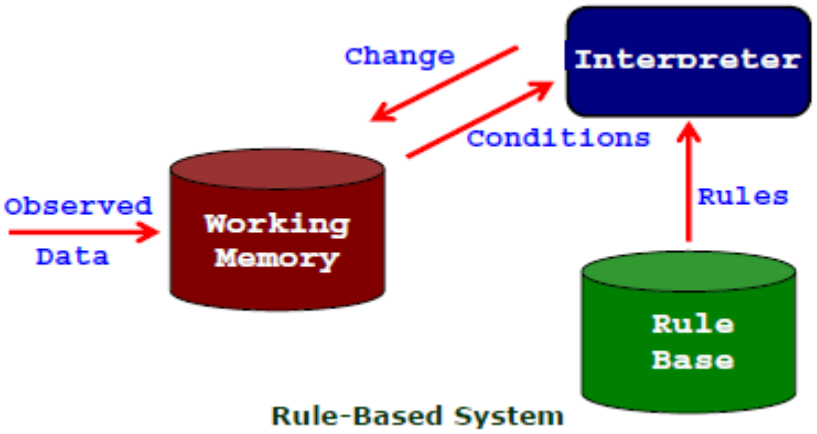
c). Breadth-First Search (BFS): Look for solution amongst all nodes at a give level before proceeding to the next.

d). Beam Search: Like breadth first but selecting only those N nodes at each level that are most likely to lead to a solution

e) Best-First Search: like beam search but only proceeding from one most likely node at each level

Rule-Based Systems (RBSs)

- These are simple and are successful AI techniques.
- Rules are of the form: IF condition THEN action
- Rules are often arranged in hierarchies (tree).
- When all the conditions of a rule are satisfied the rule is triggered.
- RBS Components: working memory, Rules Base and Interpreter



Working Memory

- Contains facts about the world observes or derives from a rule; stored as a triplet object, attribute, value e.g car, colour, red: The colour of the car is red.
- Contains temporary knowledge about problem-solving sessions.
- Can be modified by rules

Rule Base

- Contains rules; each rule is a step in problem solving.
- Rules are domain knowledge and modified only from outside
- Rules syntax is
  IF condition THEN action
- If the condition are matched to the working memory and if fulfilled then the rule may be fired.
- Rule based actions are: Add, Remove and Modify

Interpreter

- It is the domain independent reasoning mechanism for RBS.

- It selects rule from rule base and apply by performing actions.

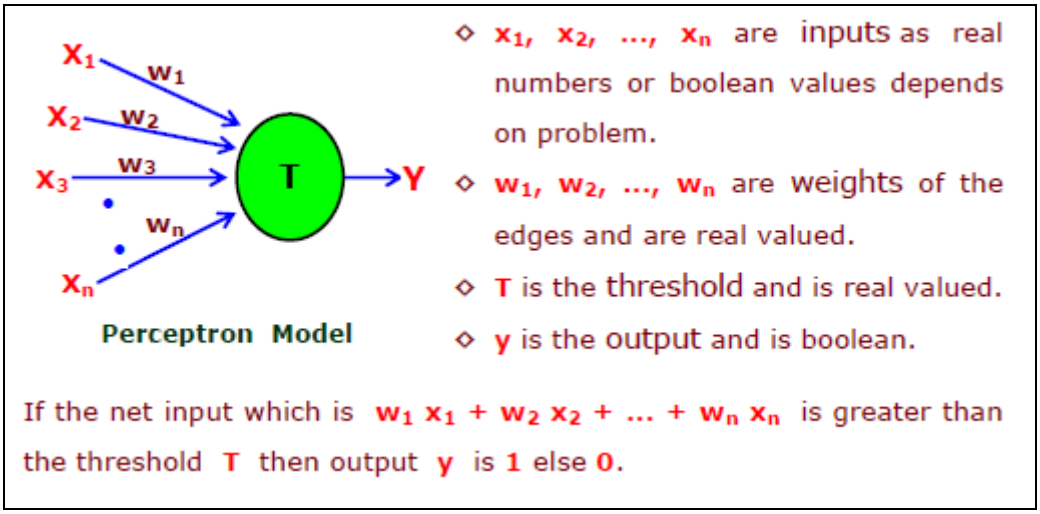
Operates on a cycle.

1. Retrieval: Finds the rule that matches the current WM.
2. Refinement: Prunes, records and resolves conflicts.
3. Executes the actions of the rules in the Conflict set and then applies the rule by performing actions.

1.5.2. Biology-Inspired Artificial Intelligence Techniques

Neural Networks (NN)

- Neural Networks model a brain learning by example.
- NN are structures trained to recognize input pattern.
- NN typical take a vector of input values and produce a vector of output values, inside they train weights of neurons.
- Perception is a model of a single trainable neuron as shown below



Perception Model

- NN use supervised learning in which inputs and outputs are known and the goal is to build a representation of a function that will approximate the input to output mapping

Genetic Algorithms (GA)

- They are part of evolutionary computing and a rapidly growing area of AI.
- They are implemented as computer simulation where techniques are inspired by evolutionary biology.
- Mechanism of biological evolution.
- Every organism has a set of rules.
- The genes are connected together into long string called chromosomes.
- Each gene represent a specific trait(feature).
- The gene and their settings are referred as an organism's, Genotype
- When two organism mate they share their genes
- A gene can mutate and form a new trait in and organism.
- GAs provide a way of solving problems by mimicking processes, the nature uses, Selection, Cross over, Mutation and accepting to evolve a solution to problem.

Steps in developing genetic Algorithms

Start: Generate random population of n Chromosomes.

- Fitness: Evaluate the fitness f (x) of each chromosome x in the population.

- New Population: Create a new population by repeating following steps until the new population is complete
  - Selection: Select two parents chromosomes from a population according to their fitness
  - Crossover: With a crossover probability cross over the parent to form new offspring.
  - Mutation: With mutation probability, mutates new offspring at each locus.
  - Accepting: place new offspring in the new population

  Replace: Use new generated population for further run of the algorithm.
  Test: If the end condition is satisfied, stop, and return the best solution in current population.
  Go to step 2'
- GA does unsupervised learning if the right answer is not known beforehand

1.6. Branches of Artificial Intelligence

Branches of AI are concerned with how we represent, manipulate and reason with knowledge in order to solve problems.

Logical Artificial Intelligence

Logic is a language for reasoning; a collection of rules used while doing logical reasoning.
Types of logic:

- Preposition logic- logic of sentences.
- Predicative logic- Logic of objects.
- Fuzzy logic- dealing with fuzziness
- Temporal logic, etc.

Preposition logic and Predicate logic are fundamental to all logic.

Preposition logic.

Prepositions are "sentences"; either true or false but not both. A sentence is smallest unit in propositional logic. If proposition is true, then truth value is "true"; else "false". Example: Sentence "Grass is green"; Truth value "true"; Proposition is "yes".

Predicate Logic.

Predicate is a function may be true or false for arguments. Predicate logical are rules that govern quantifiers. Predicative logic is propositional logic added with quantifiers.
Examples:

- The car Tom is driving is blue
- The sky is blue
- The cover of this book is blue.

  Predicate: is blue, give a name B;
  Sentence: represented as B(x); read B(x) as X is blue
  Object: represented as X.

Search in Artificial Intelligence

- Search is a problem-solving technique that systematically consider all possible action to find a path from initial to target state.
- Search techniques are many; the most fundamental are
  a) Depth first
  b) Hill climbing
  c) Breadth First
  d) Least cost

- Search components
  a) Initial state-First location.
  b) Available actions-Successor function: reachable states.
  c) Goal test- Conditions for goal satisfaction
  d) Path cost-Cost of sequence from initial state to reachable state
- Search Objective: Transform initial state into goal state-find a sequence of action.
- Search solution: Path from initial state to goal-optimal if lowest cost

Pattern Recognition (PR)

Definition: The science that concerns the description or classification (recognition) of measurement.
Pattern recognition problem

- Machine vision-Visual inspection.
- Character recognition-mail sorting, processing bank cheques.
- Computer aided diagnosis- Medical image.
- Speech recognition- Human computer interaction, access.

Approaches for Pattern recognition.

- Template Matching: Match with stored template considering translation, rotation and scale change, measure similarity based on training set. Measure "Similarity" (correlation) based on training set.
- Statistical classification: Each pattern is represented in terms of features and viewed as a point in a dimensional space. Using training sets establish decision boundaries in the feature space following "decision theoretic" or "discriminate analysis" approaches.
- Syntactic or Structural matching: Complex pattern is composed of sub-pattern and the relations; they themselves are built from simpler elementary sub-pattern called primitives.
  The patterns are viewed as sentences belonging to a language, primitives are viewed as the alphabet of the language. The sentences are generated according to a grammar.
  A large collection of complex patterns can be described by a small number of primitives and grammatical rules. The grammar for each pattern class are inferred from the training samples.
- Neural networks: They are viewed as weighted directed graphs in which the nodes are artificial neurons and directed edges (with weights) are connections between neurons input-output.
  Neural networks have the ability to learn complex nonlinear input-output relationships from the sequential training procedures, and adapt themselves to input data.

Applications requiring Pattern recognition

• Image Proc / Segmentation • Computer Vision • Medical Diagnosis • Man and Machine Diagnostics	• Seismic Analysis • Industrial Inspection • Financial Forecast

Knowledge Representation

Knowledge representation is crucial. One of the clearest results of artificial intelligence research so far is that solving even apparently simple problems requires lots of knowledge. Knowledge is a collection of facts. To manipulate these facts by program, a suitable representation is required. Different types of knowledge require different type of representation;

- Semantic networks- A graph where the nodes represent concepts and the arcs represent binary relationships between concepts.
- Frames and Scripts: Consists of a data structure which represents a sequence of events for a given type occurrence.
- Production rules: Consist of a set of rule about a behaviour, a production consists two parts: a Precondition (IF) and an action (THEN), if a production's precondition matches the current state of the world, then the production is said to be triggered.

Inference

Inference: The act or process of deriving a conclusion based on solely on what one already knows: It is deduction of new facts from old ones.

Deductive Inference

It is never false; inference is true if premise is true. A traditional logic is based on deduction; it is a method of exact inference; there is no probability of mistake if rules are followed exactly. The information required is complete, precise and consistent. A logic is monotonic, if the truth of a proposition does not change when new information are added to the system.

Inductive Inference

It may be correct or incorrect inference, because in real world the information is incomplete, inexact and inconsistent. A logic is inductive, also called induction or inductive reasoning, if the process in which the premises of an argument are believed to support the conclusion but do not ensure it.

A logic is non-monotonic, if the truth of a proposition may change when new information is added to or an old information is deleted from the system. The reasoner draw conclusions tentatively reserving the right to retract them in the light of further information. Example: when we hear of a bird, we human infer that bird can fly, but this conclusion can be reserved when we hear that it is a penguin; the bird penguin cannot fly.

Common Sense Knowledge and Reasoning

Common sense is the mental skills that most people have. It is the ability to analyze a situation based on its context.

- People can think because the brain contains vast libraries of common sense knowledge and has the means for organizing, acquire, and using the knowledge.
- Computer can not think; they lack common senses.

Researchers have divided common sense capabilities into:

- Common sense knowledge
- Common sense reasoning
- Teaching computer common sense by building a database of human common sense knowledge.

Learning

Learning: Program learn from what the facts or behaviours can represent.

A branch of artificial intelligence, is a scientific discipline concerned with the design and development of algorithms that allow computers to evolve behaviors based on empirical data, such as from sensor data or databases

Definition: Learning is making useful changes in the working of the mind.

Others include;

Planning

A plan is a representation of a course of action. Planning is a problem solving technique. It applies reasonable series of actions to accomplish a goal. Planning programs: Start with facts about the world, particularly,

- Facts about the effects of actions.
- Facts about the particular situation.
- Statement of a goal

Benefits of planning

- Reducing search
- Resolving goal conflicts.
- Providing a basis for error recovery

Strategy for planning: It is a sequence of action. From facts the program generate a strategy for achieving the goal

1.7. Applications of AI

Game playing

Interactive computer games is an emerging area in which the goals of human-level AI are pursued.

Games are made by creating human level artificially intelligent entities, e.g. enemies, partners, and support characters that act just like humans.

Game play is a search problem defined by:

- Initial state - board
- Expand function - build all successor states
- Cost function - payoff of the state
- Goal test - ultimate state with maximal payoff

Game playing is characterized by:

- “Unpredictable" opponent - specifying move for every possible opponent reply.

Time limits - games become boring if there is no action for too long a time; Opponents are unlikely to find goal, must approximate.

Speech Recognition

A process of converting a speech signal to a sequence of words. In the 1990s, computer speech recognition reached a practical level for limited purposes. Using computers recognizing speech is quite convenient, but most users find the keyboard and the mouse still more convenient. The typical usages are voice dialing (Call home), call routing (collect call), simple data entry (credit card number). The Voice verification or Speaker recognition are a related process. The spoken language interface PEGASUS in the American Airlines' EAASY SABRE reservation system, allows users to obtain flight information and make reservations over the telephone.

Understanding Natural Language

Natural language processing (NLP) does automated generation and understanding of natural human languages. Natural language generation systems convert information from computer databases into normal-sounding human language. Natural language understanding systems convert samples of human language into more formal representations that are easier for computer programs to manipulate.

Some major tasks in NLP :

- Text-to-Speech (TTS) system - converts normal language text into speech
- Speech recognition - process of converting a speech signal to a sequence of words
- Machine translation (MT) - translate text or speech from one natural language to another.
- Information retrieval (IR)- searching for information from databases such as Internet or World Wide Web or Intranets.

Computer Vision

It is a combination of concepts, techniques and ideas from Digital Image Processing, Pattern Recognition, Artificial Intelligence and Computer Graphics. The world is composed of 3-D objects, but the inputs to the human eye and computers' TV cameras are 2-D. Some useful programs can work solely in 2-D, but full computer vision requires partial 3-D information that is not just a set of two- dimensional views. At present there are only limited ways of representing 3-D information directly, and they are not as good as what humans evidently use.

Examples:

- Face recognition – the programs in use by banks
- Autonomous driving - The ALVINN system, autonomously drove a
- The van from Washington, D.C. to San Diego, averaging 63 mph day and night, and in all weather conditions.
- Other usages are handwriting recognition, baggage inspection, manufacturing inspection, photo interpretation, etc.

Expert Systems

Systems in which human expertise is held in the form of rules which enable the system to diagnose situations without the human expert being present. A Man-machine system with specialized problem-solving expertise. The "expertise" consists of knowledge about a particular domain, understanding of problems within that domain, and "skill" at solving some of these problems. A “knowledge engineer" interviews experts in a certain domain and tries to embody their knowledge in a computer program for carrying out some task. One of the first expert systems was MYCIN in 1974, which diagnosed bacterial infections of the blood and suggested treatments. Expert systems rely on knowledge of human experts, e.g.

- Diagnosis and Troubleshooting - deduce faults and suggest corrective actions for a malfunctioning device or process
- Planning and Scheduling – analyzing a set of goals to determine and ordering a set of actions taking into account the constraints; e.g. airline scheduling of flights
- Financial Decision Making - advisory programs assists bankers to make loans, Insurance companies to assess the risk presented by the customer, etc.
- Process Monitoring and Control - analyze real-time data, noticing anomalies, predicting trends, and controlling optimality and do failure correction.

1.8. Artificial Intelligence Languages

Programming languages in artificial intelligence (AI) are the major tool for exploring and building computer programs that can be used to simulate intelligent processes such as learning, reasoning and understanding symbolic information in context. These languages includes;

1. IPL: was the first language developed for artificial intelligence. It includes features intended to support programs that could perform general problem solving, including lists, associations, schemas (frames), dynamic memory allocation, data types, recursion, associative retrieval, functions as arguments, generators (streams), and cooperative multitasking.
2. Lisp: is a practical mathematical notation for computer programs based on lambda calculus. Linked lists are one of Lisp languages' major data structures, and Lisp source code is itself made up of lists. As a result, Lisp programs can manipulate source code as a data structure, giving rise to the macro systems that allow programmers to create new syntax or even new domain-specific programming languages embedded in Lisp. There are many dialects of Lisp in use today, among them are Common Lisp, Scheme, and Clojure.
3. Prolog: is a declarative language where programs are expressed in terms of relations, and execution occurs by running *queries* over these relations. Prolog is particularly useful for symbolic reasoning, database and language parsing applications. Prolog is widely used in AI today.
4. STRIPS: is a language for expressing automated planning problem instances. It expresses an initial state, the goal states, and a set of actions. For each action preconditions (what must be established before the action is performed) and post-conditions (what is established after the action is performed) are specified.
5. Planner is a hybrid between procedural and logical languages. It gives a procedural interpretation to logical sentences where implications are interpreted with pattern-directed inference.

AI applications are also often written in standard languages like C++ and languages designed for mathematics, such as MATLAB and Lush.

1.9. Features of Artificial Intelligence

1. The use of computers to do symbolic reasoning, pattern recognition, learning, or some other form of inference.
2. A focus on problems that do not respond to algorithmic solutions. Rely on heuristic search as an AI problem-solving technique.
3. A concern with problem solving using inexact, missing, or poorly defined information.
4. Reasoning about the significant qualitative feature of a situation.
5. An attempt to deal with issues of semantic meaning as well as syntactic form.
6. Answers that are neither exact nor optimal, but are in some sense “sufficient”.
7. The use of large amounts of domain- specific knowledge in solving problems.
8. The use of meta-level knowledge to effect more sophisticated control of problem solving strategies.

Chapter Review Questions

1. Define the terms (a) intelligence (b). artificial intelligence
2. There are well-known classes of problems that are intractably difficult for computers and other classes that are provably undecidable. Does this mean that AI is impossible.
3. Examine the AI literature to discover whether the following tasks can currently be solved by computers
   a) Playing a decent game of table tennis
   b) Buying a week’s worth of groceries at the market.

c) Discovering and proving new mathematical theorems
d) Giving competent legal advice in a specialized area of law.

CHAPTER TWO

Artificial Intelligence in Business

Chapter Objectives

By the end of this chapter the learner should be able to;

- Describe the Application of AI in Business

2.1. Introduction

The potential applications of Artificial Intelligence are abundant. They stretch from the military for autonomous control and target identification, to the entertainment industry for computer games and robotic pets, to the big establishments dealing with huge amounts of information such as hospitals, banks and insurances, we can also use AI to predict customer behavior and detect trends.

AI is a broad discipline that promises to simulate numerous innate human skills such as automatic programming, case-based reasoning, decision-making, expert systems, natural language processing, pattern recognition and speech recognition etc. AI technologies bring more complex data-analysis features to existing applications

Business applications utilize the specific technologies mentioned earlier to try and make better sense of potentially enormous variability (for example, unknown patterns/relationships in sales data, customer buying habits, and so on). However, within the corporate world, AI is widely used for complex problem-solving and decision-support techniques in real-time business applications. The business applicability of AI techniques is spread across functions ranging from finance management to forecasting and product

2.2. Scope of Artificial Intelligence in Business

Business Intelligence

Business Intelligence is the process of intelligence gathering applied to business.

Definition:

"Business intelligence (BI) is a broad category of applications and technologies for gathering, storing, analysing, and providing access to data to help enterprise users make better business decisions. BI applications include the activities of decision support systems, query and reporting, online analytical processing (OLAP), statistical analysis, forecasting, and data mining." (WHATIS?COM, 2001,)

Artificial Intelligence (AI) is widely used for complex problem-solving and decision-support techniques in real-time business applications. The business applicability of AI techniques is spread across functions ranging from finance management to forecasting and production.

Enterprises that utilize AI-enhanced applications are expected to become more diverse, as the needs for the ability to analyze data across multiple variables, fraud detection and customer relationship management emerge as key business drivers to gain competitive advantage.

2.3. Emergence of AI in business

Artificial Intelligence (AI) has been used in business applications since the early eighties. As with all technologies, AI initially generated much interest, but failed to live up to the hype. However, with the advent of web-enabled infrastructure and rapid strides made by the AI development community, the

application of AI techniques in real-time business applications has picked up substantially in the recent past.

2.4. Artificial Intelligence in Manufacturing

As the manufacturing industry becomes increasingly competitive, sophisticated technology has emerged to improve productivity. Artificial Intelligence in manufacturing can be applied to a variety of systems. It can recognize patterns, plus perform time consuming and mentally challenging tasks. Artificial Intelligence can optimize your production schedule and production runs. In order for organizations to meet ever increasing customer demands, and to be able to survive in an environment where change is inevitable, it is crucial that they offer more reliable delivery dates and control their costs by analyzing them on a continual basis. For businesses, being capable of delivering high quality goods at low costs and short delivery times is akin to operating in a whirlpool environment like the Devil's Triangle, and this is no easy task for any organization. Managing so that production takes place at the right time, on the right equipment, and using the right tools will minimize any deviations in delivery dates promised to the customer. Utilizing equipment, personnel and tools to their maximal efficiency will no doubt improve any organization's competitive strength. In return, proper utilization of these capabilities will result in lower costs for the organization

Optimal scheduling of jobs on equipment, without the use of computer software, is a truly difficult undertaking. Performing planning using the "Deterministic Simulation Method" will provide you with schedules that will indicate job loads per equipment. Even in the case limited to a single piece of equipment, as the number of jobs to schedule on that equipment increases, finding the right solution in the "Possible Solutions Set" becomes next to impossible. And in the real world, the difficulties arising from the large size of the solutions set due to the recipes formed by jobs, equipment and products, and shaped by the technological restrictions, as well as the complexity in finding a close to ideal solution, are readily apparent.

Research and studies are being conducted worldwide on the subject of scheduling. Software vendors working in this area follow developments closely, and they are coming out with new products to better meet demands. "Genetic Algorithms", "Artificial Intelligence", and "Neural Networks" are some of the technologies being used for scheduling

Advantages

- View your best product runs and the corresponding settings.
- Increase efficiency and quality by using optimal settings from past production.
- Artificial Intelligence can optimize your schedule beyond normal human capabilities.
- Increase productivity by eliminating downtime due to unpredictable changes in the schedule.

2.5. Artificial Intelligence in Financial services

AI has found a home in financial services and is recognized as a valuable addition to numerous business applications. Sophisticated technologies encompassing neural networks and business rules along with AI-based techniques are yielding positive results in transaction-oriented scenarios for financial services. AI has been widely adopted in such areas of risk management, compliance, and securities trading and monitoring, with an extension into customer relationship management (CRM). Tangible benefits of AI adoption include reduced risk of fraud, increased revenues from existing customers due to newer opportunities, avoidance of fines stemming from non-compliance and averted securities trade exceptions that could result in delayed settlement, if not detected.

In the field of Finance, artificial intelligence has long been used. Some applications of Artificial Intelligence are

- Credit authorization screening
- Mortgage risk assessment
- Project management and bidding strategy
- Financial and economic forecasting
- Risk rating of exchange-traded, fixed income investments
- Detection of regularities in security price movements
- Prediction of default and bankruptcy
- Security/and or Asset Portfolio Management

Artificial intelligence types used in finance include neural networks, fuzzy logic, genetic algorithms, expert systems and intelligent agents. They are often used in combination with each other.

2.6. Artificial Intelligence in Marketing

Advances in artificial intelligence (AI) eventually could turbo-boost customer analytics to give companies speedier insights into individual buying patterns and a host of other consumer habits. Artificial intelligence functions are made possible by computerized neural networks that simulate the same types of connections that are made in the human brain to generate thought. Currently, the technology is used mostly to analyze data for genetics, pharmaceutical and other scientific research.

High-tech data mining can give companies a precise view of how particular segments of the customer base react to a product or service and propose changes consistent with those findings. In addition to further exploring customers" buying patterns, analytics could help companies react much more quickly to the marketplace. Intelligent agents could let companies make real-time changes to marketing campaigns.

2.7. Artificial Intelligence in HR

It is widely believed that the role of managers is becoming a key determinant for enterprises' competitiveness in today's knowledge economy era. Owing to fast development of information technologies (ITs), corporations are employed to enhance the capability of human resource management, which is called human resource information system (HRIS). Recently, due to promising results of artificial neural networks (ANNs) and fuzzy theory in engineering, they have also become candidates for HRIS. The artificial intelligence (AT) field can play a role in this, especially; in assuring that the fuzzy neural network has the characteristics and functions of training, learning, and simulation to make an optimal and accurate judgment according to the human thinking model.

2.8. Pros and Cons of Artificial Intelligence

Pros

- Artificial intelligence finds applications in space exploration. Intelligent robots can be used to explore space. They are machines and hence have the ability to endure the hostile environment of the interplanetary space. They can be made to adapt in such a way that planetary atmospheres do not affect their physical state and functioning.
- Emotions that often intercept rational thinking of a human being are not a hindrance for artificial thinkers. Lacking the emotional side, robots can think logically and take the right decisions.
- Sentiments are associated with moods that affect human efficiency. This is not the case with machines with artificial intelligence.
- Robots can do certain laborious tasks.
- utilized in the completion of repetitive and time-consuming tasks efficiently

Cons

- Ethics and moral values. Is it ethically correct to create replicas of human beings? Do our moral values allow us to recreate intelligence? Intelligence is after all a gift of nature. It may not be right to install it into a machine to make it work for our benefit.
- Concepts such as wholeheartedness and dedication in work bear no existence in the world of artificial intelligence.
- Thinking machines lack a creative mind

CHAPTER 3

Problem Solving, Search and Control Strategies

Chapter Objectives

By the end of this chapter the learner should be able to;

- Explain problem solving in AI
- Explain the AI. Search and Control Strategies
- Exhaustive Strategies
- Heuristic Search Techniques

  Constraint Satisfaction Technique. Problem and Model

3.1 Introduction

Researchers in artificial intelligence have segregated the AI problems from the non-AI problems. Generally, problems, for which straightforward mathematical / logical algorithms are not readily available and which can be solved by intuitive approach only, are called AI problems.

For solving an AI problem, one may employ both artificial intelligence and non-AI algorithms. An obvious question is: what is an AI algorithm? Formally speaking, an artificial intelligence algorithm generally means a non-conventional intuitive approach for problem solving. The key to artificial intelligence approach is intelligent search and matching. In an intelligent search problem / sub-problem, given a goal (or starting) state, one has to reach that state from one or more known starting (or goal) states.

Problems dealt with in artificial intelligence generally use a common term called 'state'. A state represents a status of the solution at a given step of the problem solving procedure. The solution of a problem, thus, is a collection of the problem states. The problem solving procedure applies an operator to a state to get the next state. Then it applies another operator to the resulting state to derive a new state. The process of applying an operator to a state and its subsequent transition to the next state, thus, is continued until the goal (desired) state is derived.

An important aspect of intelligence is *goal-based* problem solving. The solution of many problems (e.g. noughts and crosses, timetabling, chess) can be described by finding a sequence of actions that lead to a desirable goal. Each action changes the *state* and the aim is to find the sequence of actions and states that lead from the initial (start) state to a final (goal) state. Problem solving is fundamental to many AI-based applications. Problem solving is a process of generating solutions from observe data.

Problem solving, particularly in artificial intelligence, may be characterized as a systematic search through a range of possible actions in order to reach some predefined goal or solution. Problem-solving methods divide into special purpose and general purpose. A special-purpose method is tailor-made for a particular problem and often exploits very specific features of the situation in which the problem is embedded. In contrast, a general-purpose method is applicable to a wide variety of problems.

Problem solving relates to analysis of AI. It may be characterized as a Systematic search through a range of possible actions to reach some predefined goal or solution. Problem solving methods are categorized as special purpose and general purpose

- Special purpose methods is method designed to solve a particular problem (tailor made for a particular problem). It exploits specific features of the situation in which the problem is embedded
- General purpose method applied to a wide range of problems. It is a step-by-step, or incremental, reduction of the difference between current states and final goal. Example "means-end analysis"

3.2. Problem Description

Problem descriptions involves the identifying the following components.

1. A problem consists of the description of;
   - Current state of the world
   - Actions that transform one state of the world into another.
   - Desired state of the world
2. A state space: describes everything that is needed to solve a problem and nothing is not needed to solve the problem.
   Problem State: A state is a representation of elements at a given moment. A problem is defined by its elements and relations. At each instant of a problem, the element have specific descriptors and relation; the descriptor tell- how to select elements. Among all possible states, the are two special states called
3. Initial state which is the start point
4. Goal state is the conditions it has to fulfill.
   - Description of a desired state of the world;
   - The description may be complete or partial
5. Operators are to change state. Operators consist of Preconditions and Instructions.
   - Precondition provides partial description of the state of the world that must be true in order to perform the action.
   - Instruction tell on how to create the next state.

   Operators should be as general as possible to reduce their number.
6. Element of the domain has relevance to the problem. Knowledge of the starting point.
7. Problem solving is finding an ordered sequence of operators that transform the current state into a goal state.
8. Restriction are solution quality any, optimal or all
   - Finding the shortest sequence or;
   - Finding the least expensive sequence defining cost or.
   - Finding any sequence as quickly as possible.

3.3. Steps in Solving a Problem

Problem solving has been a key area of concern for artificial Intelligence. Problem solving is a process of generating solutions from observed data.

- a “problem” is characterized by a set of goals,
- a set of objects, and
- a set of operations.

In building a system to solve a particular problem; the steps below are followed.

- Define the problem precisely- find input situations as well as final situations for accepting solution to the problem.
- Analyze the problem- find few important features that may have impact on the appropriateness of various possible techniques for solving the problem.
- Isolate and represent task knowledge necessary to solve the problem
- Choose the best problem solving techniques and apply to the particular problem

3.3.1. Problem Definition

A problem is defined by its elements and their relations. At each instant of a problem, the element have specific descriptors and relation; the descriptor tells- how to select elements. Among all possible states, are two special states called

- Initial state which is the start point
- Final state which is the goal state

To provide a form definition of the problem we need the following information

1. Initial state; the starting point for the intelligent agent.
2. Operator or successor function - A successor function is needed for state change. The successor functions moves one state to another state. A successor Function:
   - Is a description of possible actions; a set of operators
   - Is a transformation function on a state representation, which converts that state into another state.
   - Defines a relation of accessibility among states
   - Represents the conditions of applicability of a state and corresponding transformation function. Example, for any state x returns s(x), the set of states reachable from x with one action
3. State space - all states reachable from initial by any sequence of actions ie. It contains contain all the possible configuration of the relevant objects including some impossible. Together the initial state and the successor function implicitly define the state space of the problem.
   - It forms a graph/map in which the nodes are the states and the arcs between the nodes are actions.
   - In state space, a path is a sequence of states connected by a sequence of actions
   - The solution of a problem is the part of the map formed by the state space

   Structure of a state space

   The structure of a state space are trees and graphs.
   - Tree is a hierarchical structure in a graphical form.
   - Graph is a non-hierarchical structure
   - A tree has only one path to a given node.
   - A graph consists of a set of node (vertices) and a set of edges (arcs); i.e a graph has several paths to a given node.
   - Operators are directed arcs between nodes t

   Search process explores the state space. In the worst case, the search explores all possible paths between the initial state and the goal state.
4. Path - sequence of states connected by a sequence of actions through state space,
5. Path cost - function that assigns a numeric cost to each/a path. Cost of a path is the sum of costs of individual actions along the path. The problem solving agent chooses a cost function that reflects its own performance measure
6. Goal test – used to determine whether a given state is a goal state; ie. desired end has been achieved.

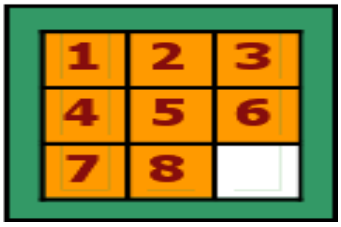
Others includes;

7. Problem Solution: Specify one or more state that would be acceptable solution to the problem called goals. In the state space, a solution is a path from initial state to a goal state or sometimes just a goal state.
   - A solution cost function assigns a numeric cost to each path;
   - It also gives the cost of applying the operator to the state.
   - A solution quality is measured by the path cost function; and an optimal solution has the lowest cost among all solution.
   - The solution may be any or optimal or all.
   - The importance of cost depends on the problem and the type of solution asked.

8. Specify a set of rules that describe the actions available.
   The problem is solved by using the rules, in combination with appropriate control strategy to move through the problem space until a path from an initial state to a goal state is found. This process is called a search. Search refers to the search for a solution in a problem space using different search control strategies. A solution is a combination of operations and objects that achieve the goals.

Examples of Problem definitions

• Example 1 : 8 – Puzzle



- State space: configuration of 8 - tiles on the board
- Initial state: any configuration
- Goal state: tiles in a specific order
- Action: "blank moves"
  - ✓ Condition: the move is within the board
  - ✓ Transformation: blank moves Left, Right, Up, Dn
- Solution: optimal sequence of operators

Examples of Problem Solving Agent

```
function SIMPLE-PROBLEM-SOLVING-AGENT(percept) returns an action
  persistent: seq, an action sequence, initially empty
              state, some description of the current world state
              goal, a goal, initially null
              problem, a problem formulation

  state ← UPDATE-STATE(state, percept)
  if seq is empty then do
      goal ← FORMULATE-GOAL(state)
      problem ← FORMULATE-PROBLEM(state, goal)
      seq ← SEARCH(problem)
      if seq = failure then return a null action
  action ← FIRST(seq)
  seq ← REST(seq)
  return action
```

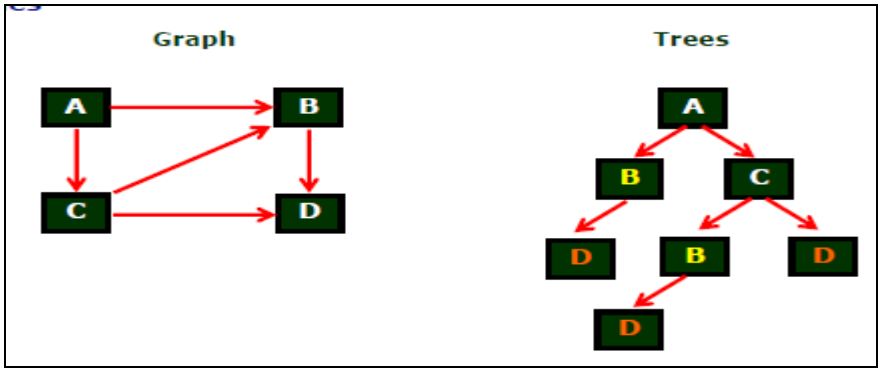
Figure 3.1 A simple problem-solving agent. It first formulates a goal and a problem, searches for a sequence of actions that would solve the problem, and then executes the actions one at a time. When this is complete, it formulates another goal and starts over.

3.3.2. Problem space

The problem space is represented by directed graph where nodes represent search state and path represent the operator applied to the change of the state. To simplify a search algorithms, it is often convenient to logically and programmatically represent a problem space as a tree. A tree decreases the complexity of a

search at a cost. Here, the cost is due to duplicating some nodes on the tree that were linked numerous times in the graph; e.g node B and node D.

A tree is a graph in which any two vertices are connected by exactly one path. Alternatively, any connected graph with no cycles is a tree. Example



3.4. Search and Control Strategies

3.4.1. Search

Search is fundamental to the problem solving process. Search is the systematic examination of states to find path from the start/initial state to the goal state. Search is a general mechanism that can be used when more direct method is not known. Search provides the framework into which more direct methods for solving subparts of a problem can be embedded. A very large number of Artificial Intelligence problems are formulated as search problems

The word search refers to finding a solution in a problem space.

- Search usually results from lack of knowledge
- Search explores knowledge alternatives to arrive at the best answer.
- Search algorithms out is a solution (path from the initial state to goal test.

For general-purpose problem solve:

- Deals with finding nodes having certain properties in a graph that represent search space.
- Search methods explore the search space , evaluating possibilities without investigating every single possibility

Example :

The search trees are multilevel indexes used to guide the search for data items, given some search criteria.

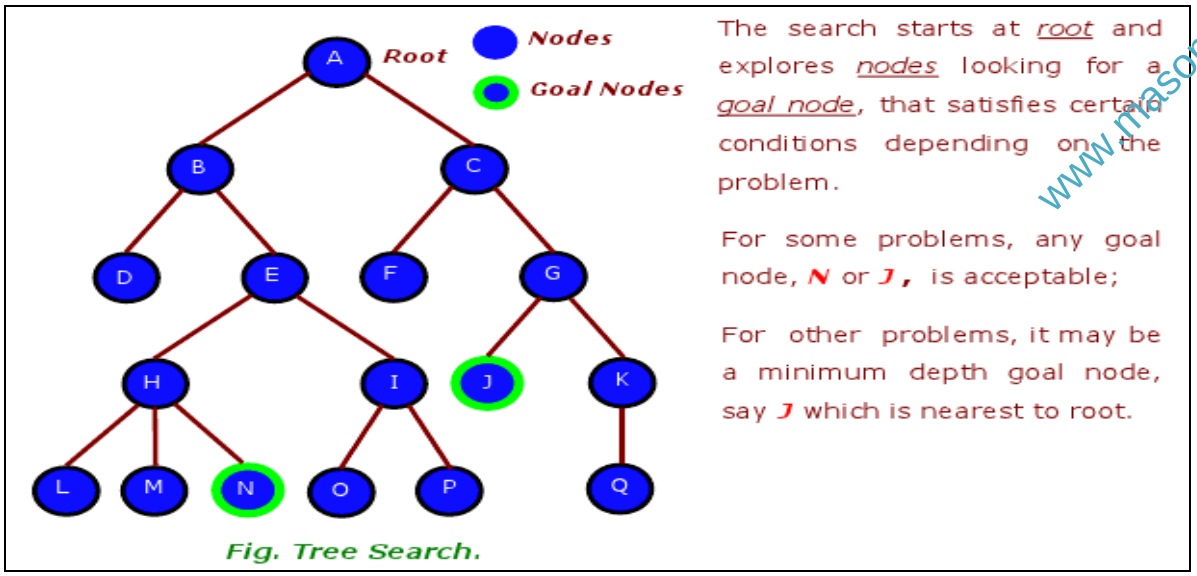


Fig. Tree Search.

The root of a search tree is a search node which corresponds to the initial state. A node is a data structure with five components

- STATE: the state in the state space to which the node corresponds.
- PARENT: the node in the search tree that generated this node.
- ACTION: the action that was applied to the parent to generate the node.
- PATH-COST: the cost, traditionally denoted by g(n), of the path from the initial state to the node, as indicated by the parent pointers and;
- DEPTH: the number of steps along the path from the initial state.

The first step is to test whether it is goal state. If it is not a goal state, then it's current state is expanding by applying the successor function to the current state, thereby generating a new set of states. The choice of which node to expand during the search is determined by the search strategy.

3.5. Search Related Terms

a) Search Space
b) Algorithm's Performance and Complexity
c) Computational Complexity
d) Tree structure
e) Stacks and queues

3.5.1. Search Space

The set of possible states, together with *operators* defining their connectivity constitute the *search space*. The output of a search algorithm is a solution, that is, a path from the initial state to a state that satisfies the goal test. In real life search usually results from a lack of knowledge. In AI too search is merely a offensive instrument with which to attack problems that we can't seem to solve any better way.

Search techniques fall into three groups:

1. Methods which find *any* start - goal path,
2. Methods which find the *best* path, and finally
3. Search methods in the face of adversaries.

3.5.2. Algorithm's Performance and Complexity

This provides a common measure to be used in comparing approaches in order to select the most appropriate algorithm for a given solution.

Performance depends on both internal and external factors

Internal factors	External Factors
• Time required to run • Space (memory) required run	• Size of input to algorithm • Speed of the computer • Quality of the compiler

- Complexity is the measure of the performance of the Algorithm, which is measured based on the internal factor of space and time.
- Computational Complexity: This measure the resources in terms of Time and space: Example If A is an algorithm that solves a decision problem f, then run time of A is the number of steps take on the input of length n
  - ✓ Time Complexity: T(n) of the decision problem f is the time of best Algorithm A for f
  - ✓ Space complexity S(n) the decision problem f is the amount of memory used by the best Algorithm A for f

3.5.3. Computational Complexity

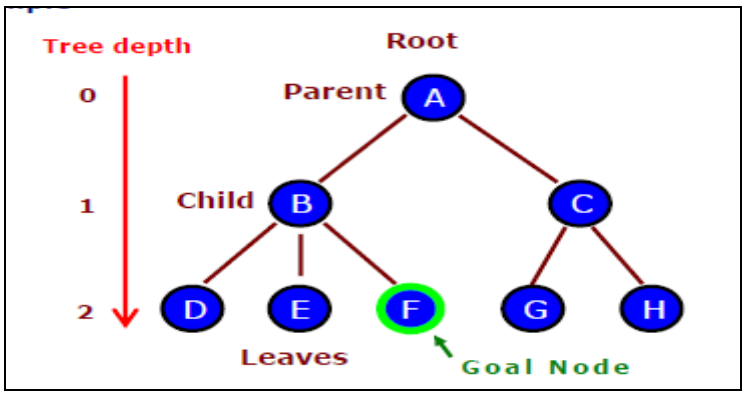
It is the measure of resources in terms of Time and Space.

- If A is an algorithm that solves a decision problem f then run time of A is the number of steps taken on the input of length n.
- Time Complexity T(n) of a decision problem f is the run time of the `best' algorithm A for f .
- Space Complexity S(n) of a decision problem f is the amount of memory used by the `best' algorithm A for f .

3.5.4. Tree Structure

Tree is a way of organizing objects, related in a hierarchical fashion. This is a type of data structure where:

- Each element is attached to one or more element directly beneath it.
- The connection between elements are called branches
- They are often called inverted trees because their roots are at the top.
- The elements that do not have elements below them are called leaves.
- A Binary tree is a special type, each element has two branches below it.



Example of a tree Structure

Properties of a tree structure

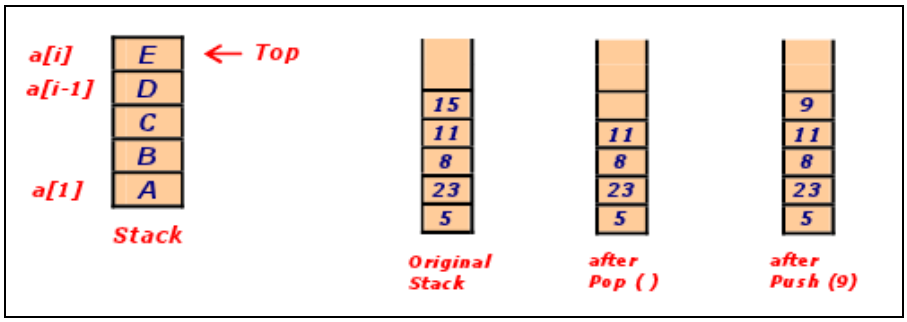
- It is a special case of a graph
- The topmost node in the tree is the root node and all operations on the tree begin at the root node.
- Each node has either zero or more child below it.
- Node at the bottom most level is the leaf node and leaf nodes do not have children.
- The node which has a child is the parent node.
- The depth of the node n is the length of the path from the root node to the node; the root is at zero depth.

3.5.5. Stacks and Queues

Stacks and Queues are data structures maintain the Last-in, first-Out and First-In First-Out respectively. Both stacks and queues are often implemented as linked lists, but that is not the only possible implementation.

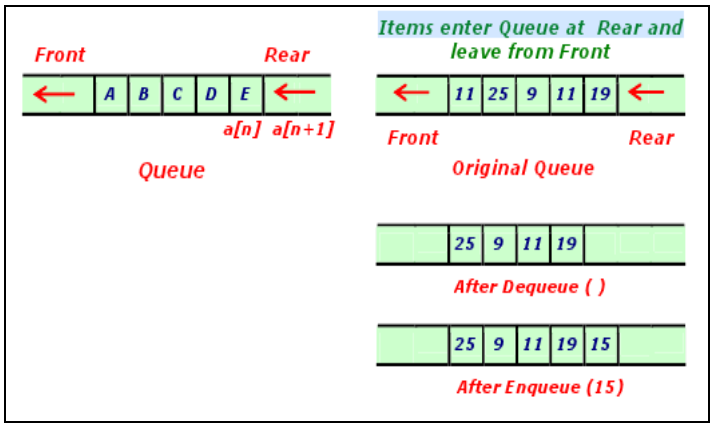
Stacks: A stack maintain the order Last-in first-Out (LIFO) The items are a sequence and piled one on top of the other.

- An ordered list ; a sequence of items, piled one on top of the other.
- The insertions and deletions are made at one end only, called Top.
- If Stack S = (a[1], a[2], ... a[n]) then a[1] is bottom most element
- Any intermediate element (a[i]) is on top of element a[i-1], 1 < i <= n.
- In Stack all operation take place on Top.
    - ✓ The Pop operation removes item from top of the stack.
    - ✓ The Push operation adds an item on top of the stack.



Queue: An ordered list which work as a First-In First-Out (FIFO).The items are in the sequence but has restrictions on how items are added to and removed from the list.

- A queue has two ends.
- All insertions (enqueue) take place at one end, called Rear or Back
- All deletions (dequeue) take place at other end, called Front.
- If Queue has a[n] as rear element then a[i+1] is behind a[i] , 1 < i <= n.
- All operation take place at one end of queue or the other.
- The Dequeue operation removes the item at Front of the queue.
- The Enqueue operation adds an item to the Rear of the queue.
- Items enter Queue at Rear and



3.6. Search Algorithms

Many traditional search algorithms are used in AI applications. For complex problems, the traditional algorithms are unable to find the solution within some practical time and space limits. Consequently, many special techniques are developed, using heuristic functions.

The algorithms that use heuristic functions are called heuristic algorithms.

- Heuristic algorithms are not really intelligent; they appear to be intelligent because they achieve better performance.
- Heuristic algorithms are more efficient because they take advantage of feedback from the data to direct the search path.

A good heuristic will make an informed search dramatically out-perform any uninformed search. For example, the Traveling Salesman Problem (TSP) where the goal is to find a good solution instead of finding the best solution. Some prominent intelligent search algorithms are stated below.

1. Generate and Test Search
2. Best-first Search
3. Greedy Search
4. A* Search
5. Constraint Search
6. Means-ends analysis
7. Hill Climbing
8. Heuristic Search

1. Generate and Test Approach: This approach concerns the generation of the state-space from a known starting state (root) of the problem and continues expanding the reasoning space until the goal node or the terminal state is reached. In fact after generation of each and every state, the generated node is compared with the known goal state. When the goal is found, the algorithm terminates. In case there exist multiple paths leading to the goal, then the path having the smallest distance from the root is preferred. The basic strategy used in this search is only generation of states and their testing for goals but it does not allow filtering of states.
2. Hill Climbing Approach: Under this approach, one has to first generate a starting state and measure the total cost for reaching the goal from the given starting state. Let this cost be f. While f = a predefined utility value and the goal is not reached, new nodes are generated as children of the current node. However, in case all the neighborhood nodes (states) yield an identical value of f and the goal is not included in the set of these nodes, the search algorithm

is trapped at a hillock or local extrema. One way to overcome this problem is to select randomly a new starting state and then continue the above search process. While proving trigonometric identities, we often use Hill Climbing, perhaps unknowingly.

3. Heuristic Search: Classically heuristics means rule of thumb. In heuristic search, we generally use one or more heuristic functions to determine the better candidate states among a set of legal states that could be generated from a known state. The heuristic function, in other words, measures the fitness of the candidate states. The better the selection of the states, the fewer will be the number of intermediate states for reaching the goal. However, the most difficult task in heuristic search problems is the selection of the heuristic functions. One has to select them intuitively, so that in most cases hopefully it would be able to prune the search space correctly. We will discuss many of these issues in a separate chapter on Intelligent Search.
4. Means and Ends Analysis: This method of search attempts to reduce the gap between the current state and the goal state. One simple way to explore this method is to measure the distance between the current state and the goal, and then apply an operator to the current state, so that the distance between the resulting state and the goal is reduced. In many mathematical theorem- proving processes, we use Means and Ends Analysis.

3.6.1. Measuring Problem Solving Performance

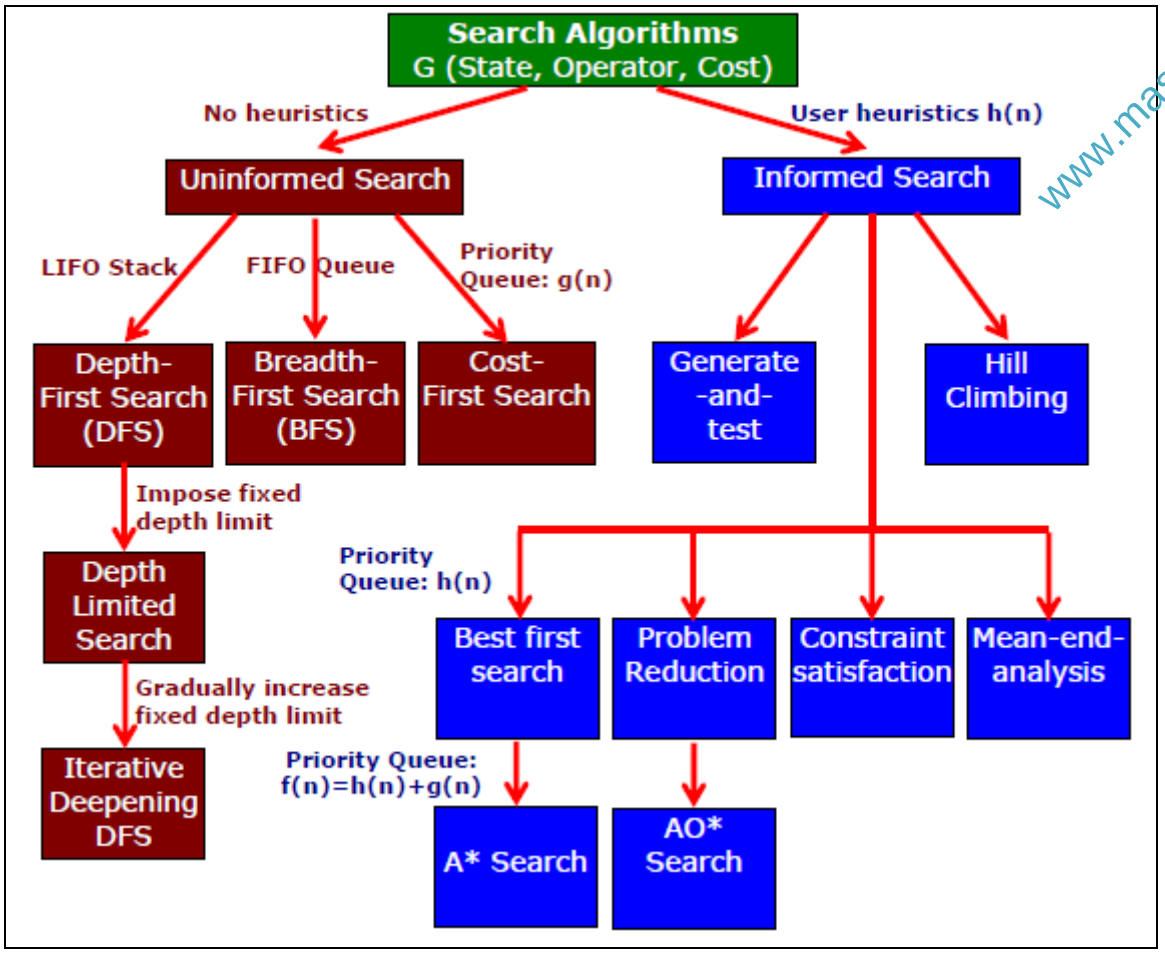
The output of a problem-solving algorithm is either failure or a solution. The search algorithm strategies are evaluated in terms of:

- Completeness: is the algorithm guaranteed to find a solution when there is one
- Time Complexity: How long does it take to find a solution
- Space complexity: How much memory is needed to perform the search and
- Optimality: Does the strategy find the optimal solution as defined?.

3.6.2. Hierarchical Representation of Search Algorithms

Two type;

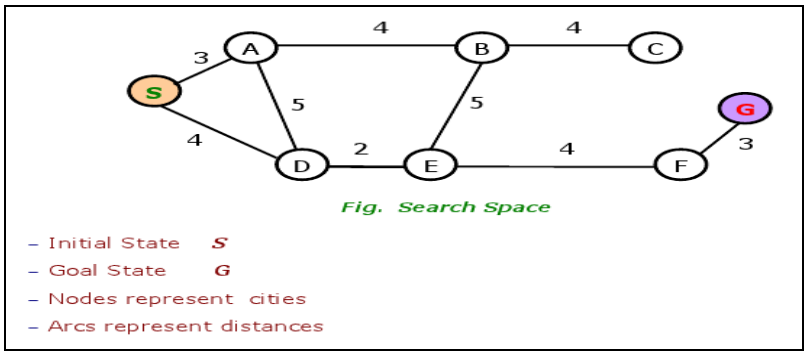
1. Uninformed Search: sometimes called blind, exhaustive or bruto-force. Methods that do not use any specific knowledge about the problem to guide the search and therefore may not be very efficient. Search through the search space all possible candidates for the solution checking whether each candidate satisfies the problem's statement. Example;
    - Depth-first
    - Breadth-first
    - Non-deterministic search
    - Iterative deepening
    - Bi-directional search
2. Informed Search: sometime called heuristics or intelligent search which 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. They are specific to the problem.



3.6.3. Search Space

A "set of all states" , which can be reached, constitute a search space.
This is obtained by applying some combination of operators defining their connectivity.

Example
Find route from Start to Goal state.
Consider the vertices as city and the edges as distances.



3.6.4. Formal Statement:

Problem solving is a set of statements describing the desired states expressed in a suitable language; e.g. first-order logic. The solution of many problems (like chess, crosses) can be described by finding a sequence of actions that lead to a desired goal.

- each action changes the state, and

- the aim is to find the sequence of actions that lead from (start) state to a final (goal) state.

A well-defined problem can be described by an example stated below.

Example

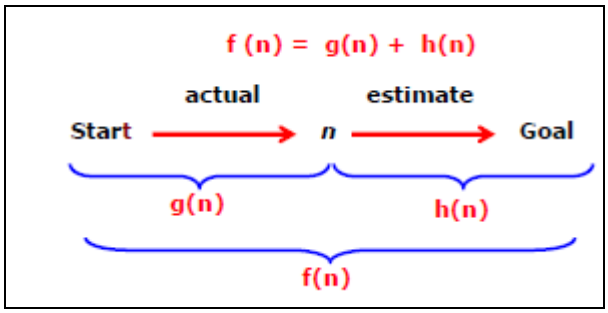
1. Initial State: (S)
2. Operator or successor function: for any state x, returns s(x), the set of states reachable from x with one action.
3. State space: all states reachable from initial by any sequence of actions.
4. Path: sequence through state space.
5. Path cost: function that assigns a cost to a path; cost of a path is the sum of costs of individual actions along the path.
6. Goal state : (G)
7. Goal test: test to determine if at goal state.

3.6.5. Search Notations

Search is the systematic examination of states to find path from the start/root state to the goal state.
The notations used to define a search are;

1. Evaluation function f (n) that estimate least cost solution through node n.
2. Heuristic function h(n) that estimates least cost path from node n to goal node
3. Cost function g(n) that estimates the least cost path from start node to node n

The relation among three parameters are expressed as



Relationships between notations

if h(n) _ actual cost of shortest path from node n to goal then h(n) is an under estimate

3.6.5.1. Estimation of Cost function g*

An estimated least cost path from start node to node n is written as g*(n)

- ✓ g* is the calculated as the actual cost of g(n)
- ✓ g* is known by summing all the path costs from start to current state

- if search space is a tree, then g* = g where g represents the optimal path since there is only one path from start node to current node.
- In general the search space is graph
- if search space is a graph, then g* ≥ g, since
  - ✓ g* can never be less than the cost of the optimal path, rather it can be only over estimate the cost.
  - ✓ g* can be equal to g in a graph if chosen properly

3.6.5.2. Estimating Heuristic Function h*

The Estimated least cost path from node n to goal node is written as h*(n)

- h* is a heuristic information, it represents a guess; "How hard it is to reach from current node to goal state ? ".
- h* may be estimated using an evaluation function f(n) that measures goodness of a node
- h* may have different values i.e between a range 0 _ h _ (n) _ h(n)
  the values lie between $0 \leq h^*(n) \leq h(n)$;
  they mean a different search algorithm.
- if h_ = h, it is a perfect heuristic which means no unnecessary nodes are ever expended.

3.7. Control Strategies

Search for a solution in a problem space, requires control strategies to control the search processes.
The search control strategies are of different types, and are realized by some specific type of control structures.

Some widely used control strategies for search are stated below.

1. Forward search: Here, the control strategies for exploring search proceeds forward from initial state to wards a solution; the methods are called data- directed.
2. Backward search: Here, the control strategies for exploring search proceeds backward from a goal or final state towards either a soluble sub problem or the initial state; the methods are called goal directed.
3. Both forward and backward search: Here, the control strategies for exploring search is a mixture of both forward and backward strategies .
4. Systematic search: Where search space is small, a systematic (but blind) method can be used to explore the whole search space. One such search method is depth-first search and the other is breath-first search.
5. Heuristic search: Many search depend on the knowledge of the problem domain. They have some measure of relative merits to guide the search. The search so guided are called heuristic search and the methods used are called heuristics.
   Heuristic search use the knowledge of the problem and provide a measure of relative merits to guide the search. A heuristic search might not always find the best solution but is guaranteed to find a good solution in reasonable time.
   Heuristic Search Algorithms :
   - First, generate a possible solution which can either be a point in the problem space or a path from the initial state.
   - Then, test to see if this possible solution is a real solution by comparing the state reached with the set of goal states.
   - Lastly, if it is a real solution, return, Otherwise repeat from the first again.

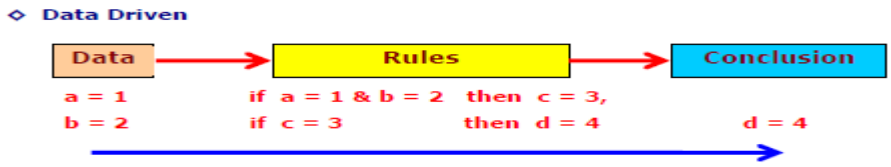
Search Chaining
Chaining refers to sharing conditions between rules, so that the same condition is evaluated once for all rules. When one or more conditions are shared between rules, they are considered "chained." Chaining are of two types:

1. Forward chaining is called data-driven and
2. Backward chaining is called query-driven.

Activation rules: Forward and Backward chaining are the two different strategies for activation of rules in the system. Forward and Backward chaining are techniques for drawing inference from rule based.

3.8. Forward Chaining Algorithm

Forward chaining is a technique for drawing inference from rules base. The algorithm proceed from a given situation to a desired goal by adding new assentation (facts) found. It compares data in a working memory against the condition in the IF parts of the rules and determines which rules to fire



◇ **Example : Forward Channing**

- Given : A Rule base contains following Rule set;

  Rule 1: If **A** and **C** Then **F**
  Rule 2: If **A** and **E** Then **G**
  Rule 3: If **B** Then **E**
  Rule 4: If **G** Then **D**

- Problem : Prove that

  If **A** and **B** true Then **D** is true

- **Solution** : ***(continued - the Forward Chaining problem)***

  (i) ∓ ***Start*** with ***A, B*** is true at ***Rule 1***
  ∓ go forward/down till a rule "fires" is found.

  First iteration :

  (ii) ∓ ***Rule 3*** fires : conclusion ***E*** is true
  ∓ new knowledge found

  (iii) ∓ No other rule fires;
  ∓ end of first iteration.

  (iv) ∓ Goal not found;
  ∓ new knowledge found at (ii);
  ∓ go for second iteration

  Second iteration :

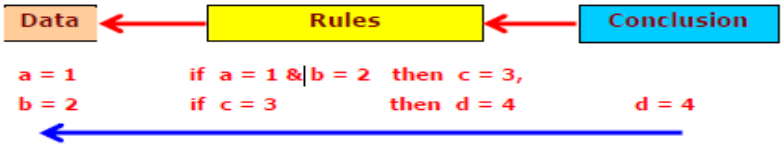
  (v) ∓ ***Rule 2*** fires : conclusion ***G*** is true
  ∓ new knowledge found

  (vi) ∓ ***Rule 4*** fires : conclusion ***D*** is true
  ∓ ***Goal found;***
  ∓ ***Proved***

3.9. Backward Chaining Algorithm

Backward Chaining is also used to draw inference from rule base. The inference is often called goal-driven. The algorithm start from the desired goal, adding new assertions found. For Backward-chaining, systems looks for actions in then THEN clause of the rules that matches the specified goals

◇ Goal Driven



◇ Example : Backward Channing

◇ Given : Rule base contains following Rule set

Rule 1: If A and C Then F
Rule 2: If A and E Then G
Rule 3: If B Then E
Rule 4: If G Then D

◇ Problem : Prove

If A and B true Then D is true

◇ Solution : *(continued - the Backward Chaining problem)*

(i) ∓ *Start* with goal ie *D* is true
∓ go backward/up till a rule *"fires"* is found.

First iteration :

(ii) ∓ *Rule 4* fires :
∓ new *sub goal* to prove *G* is true
∓ go backward

(iii) ∓ *Rule 2* "fires"; conclusion: *A* is true
∓ new sub goal to prove *E* is true
∓ go backward;

(iv) ∓ no other rule fires; end of first iteration.
∓ new sub goal found at (ii);
∓ go for second iteration

Second iteration :

(v) ∓ *Rule 3* fires :
∓ conclusion *B* is true (2nd input found)
∓ both inputs *A* and *B* *ascertained*
∓ *Proved*

3.10. Exhaustive Search/ Uninformed Search

A search is said to be exhaustive if the search is guaranteed to generate all reachable (outcomes) before it terminates with failure. A graphical representation of all possible reachable state and paths by which they may be reached is decision tree.

The most used strategies are:

1. Breadth-first Search (BSF): Is a strategy in which the highest layer of a decision tree is searched completely before proceeding to the next layer.
2. Depth-First Search (DFS): Is a strategy that extends the current path as far as possible before backtracking to the last choice point and trying the next alternative path.

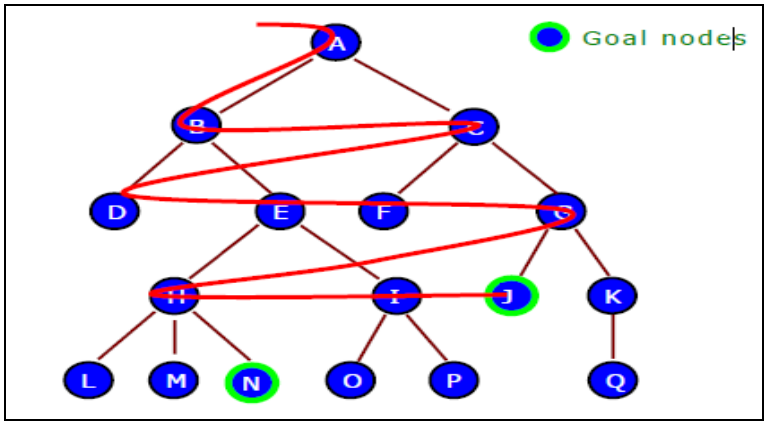
The Breadth-first search (BFS) and depth-first search (DFS) are the foundation for all other search techniques.

Other include; Depth-limited search, uniform-cost search and Bi-directional search

3.10.1. Breadth-First Search (BFS) Strategy

- The search generates/ expands all nodes at a particular level/depth in the search tree before any node at the next level is expanded.
- The search systematically proceeds tasting each node that is reachable from a parent node before it expands to any child of those nodes. The control engine guarantees that the space of possible moves is systematically examined.
- The search terminates when a solution is found and the test returns true.
- Explores nodes nearest to the root before exploring nodes that are further away.
- In this strategy, no viable solution is omitted and therefore guarantee that optimal solution is found.
- This strategy is often not feasible when the search space is large.
- Requires large memory and more time to execute

Example: Breadth-first search tree



Breadth-First search

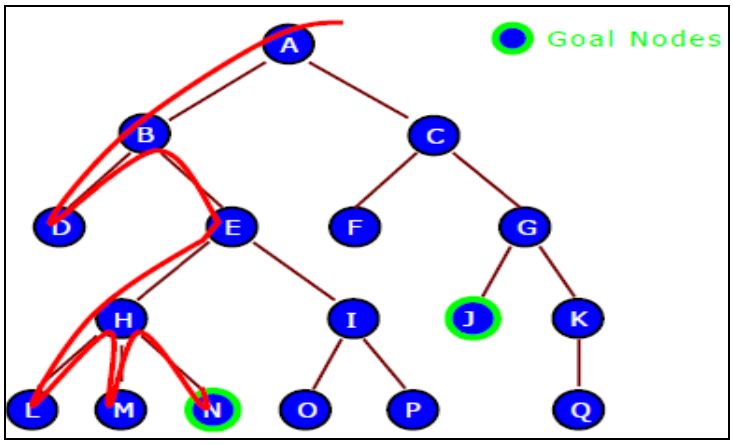
3.10.2. Depth-First Search (DFS)

The search always expands the deepest node in the current fringe of the search tree. The search proceeds immediately to the deepest level of the search tree, where the nodes have no successors. As the nodes are expanded, they are dropped from the fringe, so the search "backs up" to the next shallowest node that still has unexplored successors.

The search systematically proceeds to some depth, before another path is considered.

- If the maximum depth of the search tree is three, then if this limit is reached and if the solution has not been found, then the search backtracks to the previous level and explores any remaining alternatives at this level.
- The systematic backtracking procedure ensures that all the possibilities available are explored.
- It explores a path all the way to a leaf before backtracking and exploring another path.
- This strategy does not guarantee that the optimal solution has been found.
- In this strategy, search reaches a satisfactory solution more rapidly than breadth first, an advantage when the search space is large.
- If the tree is very deep and the maximum depth searched is less then the maximum depth of the tree, then this procedure is "exhaustive modulo of depth" that has been set.
- Requires moderate memory space

Example: Depth-first search tree



Depth-First search

Comparison between Depth-First and Breadth-First Searches

Here the Depth-first and Breadth-first search are compared at algorithm level, at feature level and how to over come the limitations.

Depth-first search	Breadth-first search
◇ **Compare Algorithms**	

Put the root node on a stack;
while (stack is not empty)
{
remove a node from the stack;
if (node is a goal node)
return success;
put all children of node
onto the stack;
}
return failure;
	Put the root node on a queue;
	while (queue is not empty)
	{
	remove a node from the queue;
	if (node is a goal node)
	return success;
	put all children of node
	onto the queue;
	}
	return failure;

◇ **Compare Features**	
‡ When succeeds, the goal node found is not necessarily minimum depth	‡ When succeeds, it finds a minimum-depth (nearest to root) goal node
‡ Large tree, may take excessive long time to find even a nearby goal node	‡ Large tree, may require excessive memory

How to over come limitations of DFS and BFS ?

- Requires, how to combine the advantages and avoid
- Requires, how to combine the advantages and avoid the disadvantages ?
- The answer is “Depth-limited search” . This means, Depth-first searches may be performed with a depth limit.

3.10.3. Depth-First Iterative-Deepening (DFID)

DFID is a another kind of exhaustive search procedure which is a blend of depth first and breadth first search.

Algorithm: Steps

- First perform a Depth-first search (DFS) to depth one.
- Then, discarding the nodes generated in the first search, starts all over and does a DFS to level two.
- Then three ... . until the goal state is reached.

3.11. Heuristic Search Strategies / Informed Search Strategies

Depth-first and Breadth-first algorithm require a lot of time and space (memory) for practice solutions. Therefore special techniques area developed using heuristics functions to overcome these problem.

- Heuristics are rules of thumb. They do not guarantee for a solution to a problem.
- They use problem-specific knowledge beyond the definition of the problem itself. They can find solutions more efficiently than an Uniformed search strategy.
- Heuristics search is a weak techniques but can be effective if applied correctly, they require domain specific information

3.11.1. Characteristics of Heuristic Search

- They require knowledge about a domain which help search and reason in that domain.
- They incorporate domain knowledge to improve efficiency over the blind search.
- They are functions that when applied to a state return values as estimated merits of state, with respect to goal.
  - ✓ Heuristics might (for reasons) under estimate or over estimate the merit of a state with respect to goal.
  - ✓ ■ Heuristics that under estimates are desirable and called admissible.
- The evaluation function estimates likelihood of given state leading to goal state.
- Heuristic search function estimate cost from current state to goal presuming function is efficient.

Examples

- Best-first search: an instance of the general TREE-SEACH or GRAPH-SEARCH algorithm in which a node is selected for expansion based on an evaluation function f (n). The node with the lowest evaluation is selected for expansion, because the evaluation measures distance to the goal.
- Greedy Best-first search: the search tries to expand the node that is closest to the goal, on the grounds that this is likely to lead to a solution quickly. It evaluates the modes by using the heuristics function: f(n) = h(n).
- A* search: minimizes the total estimated solution cost. It evaluates nodes by combining g(n), the cost to reach the node, and h(n), the cost to get from the node to the goal.

Heuristic search compared with other search

Brute force/ Blind search	Heuristic search
Only have knowledge about already explored nodes	Estimate 'distance' to goal state
No knowledge about how far a node is from goal state	Guides search process toward goal state
	Prefer nodes that lead close to and not away from goal state

Examples: 8-puzzle

Stat space: Configuration of 8-tiles on the board.
Initial state: any Configuration
Goal: tiles in a specific order.

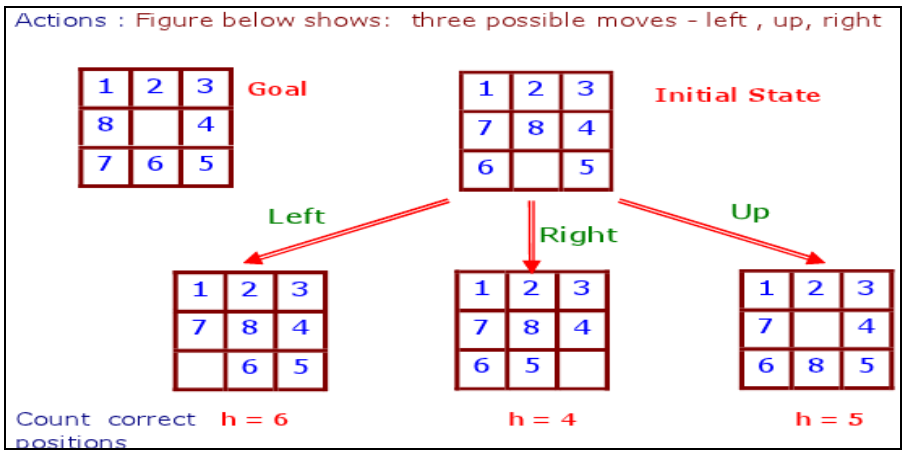
8-puzzle

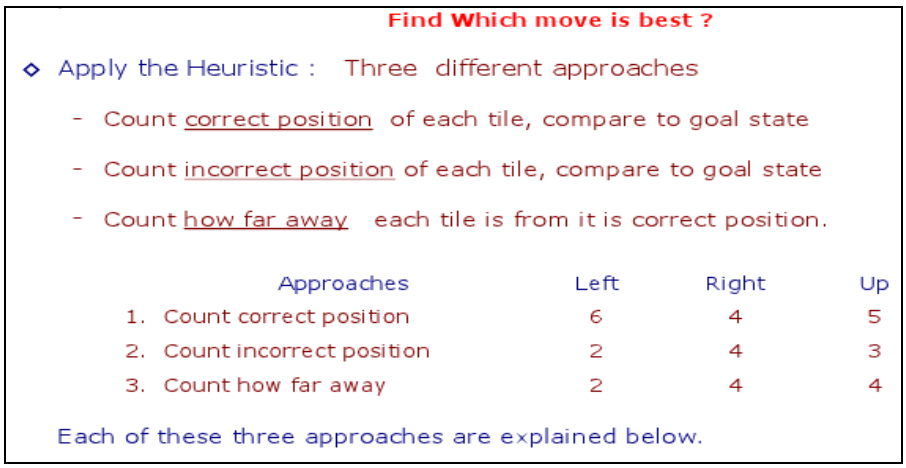
Action: Blank moves

- Condition: the move is within the board.
- Directions: Left, Right, UP Down

Problem:

- Which 8-puzzle move is best?
- What heuristic(s) can decide?
- Which move is 'best' (worth considering first)?





Heuristic: 3 - different approaches

1st approach :

Count correct position of each tile, compare to goal state.

- The higher the number the better it is.
- Easy to compute (fast and takes little memory).
- Probably the simplest possible heuristic.

2nd approach

Count incorrect position of each tile, compare to goal state

- The lower the number the better it is.

- The “best” move is where lowest number returned by heuristic.

3rd approach

Count how far away each tile is from it’s correct position

- Count how far away (how many tile movements) each tile is from it’s correct position
- Sum up this count over all the tiles.
- The “best” move is where lowest number returned by heuristic.

3.12. Constraint Satisfaction Problems (CSPs) and Models

Constraint processing is an efficient alternative to searching. Constraints arise in most areas of human endeavor. Constraints are a natural medium for people to express problems in many fields. Like most problems in artificial intelligence (AI), CSPs are solved through search.

Examples of constraints:

1. The sum of three angles of a triangle is 180 degrees,
2. The sum of the currents floating into a node must equal zero.

Constraint is a logical relation among variables.The constraints relate objects without precisely specifying their positions; moving any one, the relation is still maintained.

3.12.1. Constraint Satisfaction

- The Constraint satisfaction is a process of finding a solution to a set of constraints.
- The constraints articulate allowed values for variables, and
- Finding solution is evaluation of these variables that satisfies all

3.12.2. Constraint Satisfaction Models

Humans solve the puzzle problems stated above: trying with different configurations and using various insights about the problem to explore only a small number of configurations. It is not clear what these insights are?

For n = 8 queens on a standard chess board (8 × 8), such that no queen can attack any other queen, the puzzle has 92 distinct solutions. Humans would find it hard to solve N-Queens puzzle while N becomes more. Example: The possible number of configurations are :

1. For 4-Queens there are 256 different configurations.
2. For 8-Queens there are 16,777,216 configurations.
3. For 16-Queens there are 18,446,744,073,709,551,616 configurations.
4. In general, for N - Queens there are we have N configurations.
5. For N = 16, This would take about 12,000 years on a fast modern machine.

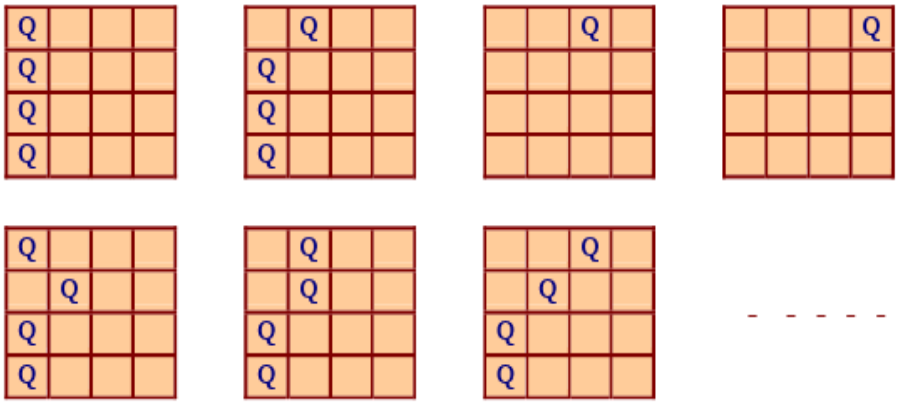
How do we solve such problems? Three computer based approaches or models are stated below. They are:

Generate and Test (GT) ,

Generate and Test (GT) : n = 4 Queens puzzle

One possible solution is to systematically try every placement of queens until we find a solution. The process is known as "Generate and Test".

Examples of Generate and Test conditions for solutions:



Backtracking (BT)

Backtracking (BT): n = 4 Queens puzzle

The Backtracking method is based on systematic examination of the possible solutions. The algorithms try each possibility until they find the right one. It differs from simple brute force, which generates all solutions, even those arising from infeasible partial solutions.

Backtracking is similar to a depth-first search but uses even less space, keeping just one current solution state and updating it.

- During search, if an alternative does not work, the search backtracks to the choice point, the place which presented different alternatives, and tries the next alternative.
- When the alternatives are exhausted, the search returns to the previous choice point and tries the next alternative there.
- If there are no more choice points, the search fails.

This is usually achieved in a recursive function where each instance takes one more variable and alternatively assigns all the available values to it, keeping the one that is consistent with subsequent recursive calls.

Note: feasible, infeasible, pruning the solution tree.

- The partial solutions are evaluated for feasibility.
- A partial solution is said to be feasible if it can be developed by further choices without violating any of the problem's constraints.
- A partial solution is said to be infeasible if there are no legitimate options for any remaining choices.
- The abandonment of infeasible partial solutions is called pruning the solution tree.

Backtracking is most efficient technique for problems, like n – Queens problem. An example below illustrates to solve N = 4 Queens problem.

Backtracking Algorithm: to solve N Queens problem.

The problem proceeds either by rows or by columns.

- for no particularly good reason, select columns to proceed.
- for each column, select a row to place the queen.
- a partial solution is feasible if no two queens can attack each other;

Note : No feasible solution can contain an infeasible partial solution.

1. Move "left to right" processing one column at a time.

2. For column J, select a row position for the queen. Check for feasibility.
   a) If there are one or more attacks possible from queens in columns 1 through (J – 1), discard the solution.
   b) For each feasible placement in column J, make the placement and try placement in column (J + 1).
   c) If there are no more feasible placements in column J, return to column (J – 1) and try another placement.
3. Continue until all N columns are assigned or until no feasible solution is found.

3.12.3. Constraint Satisfaction Problems (CSPs)

The CSPs are all around us while managing work, home life, budgeting expenses and so on;
Where we do not get success in finding solution, there we run into problems.

We need to find solution to such problems satisfying all constraints. Like most problems in artificial intelligence (AI), the constraint Satisfaction problems (CSPs) are solved through search.

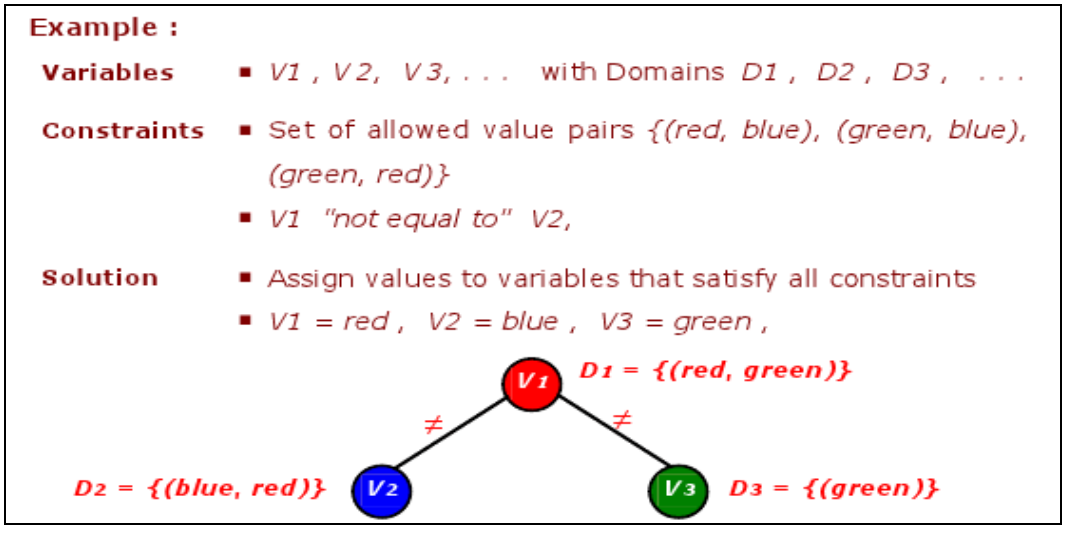
A constraint problem (or consistent labeling problem) consists of:

- a finite set of variables: z1, z2, … , zn
- for each variable: an associated finite domain of possible values di = {ai_1,ai_2, … , ai_{ni}}
- for each two variables zi, zj, i < j, a constraint c(zi, zj)

The formal definition of a CSP involves variables, their domains, and constraints. A constraint network is defined by

- a set of variables, V1 , V 2, ... , Vn,
- a domain of values, D1 , D2, ... , Dn for each variable; all variables Vi have a value in their respective domain Di.
- a set of constraints, C1 , C2, ... , Cm ; a constraint Ci restricts the possible values in the domain of some subset of the variables.
- Problem : Is there a solution of the network, i.e., an assignment of values to the variables such that all constraints are satisfied ?
- A solution to a CSP is an assignment of every variable some value in its domain such that every constraint is satisfied. Each assignment of a variable to a variable must be consistent i.e must not violate any of the constraints.

Example :



Properties of CSPs

1. Constraints are used sense. The constraints enjoy several interesting properties.
2. Constraints may specify partial information; constraint need not uniquely specify the values of its variables;
3. Constraints are non-directional, typically a constraint on (say) two variables V1, V2 can be used to infer a constraint on V1 given a constraint on V2 and vice versa;
4. Constraints are declarative; they specify what relationship must hold without specifying a computational procedure to enforce that relationship;
5. Constraints are additive; does not matter, all that matters at the end is that the conjunction of constraints is in effect;
6. Constraints are rarely independent; typically constraints in the constraint store share variables.
7. to guide reasoning of everyday common the order of imposition of constraints

Algorithms for CSPs : n = 8 Queens puzzle
As stated above, a CSP consists of 3 components :

1. A set of variables,
2. A set of values for each of the variables and
3. A set of constraints imposed between the variables.

Find a value for each variables that satisfies all the constraints.

1. Constraints
A constraint is a relation between a local collection of variables.
The constraints restrict the values that these variable can simultaneously have

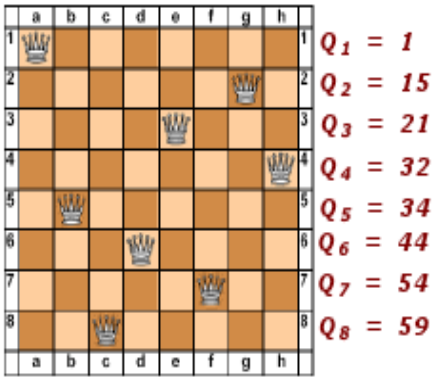
Examples : All-diff(X1, X2, X3).

This constraint says that X1, X2, X3 must take on different values.
If {1, 2, 3} is the set of values for each of these variables then:
X1 = 1, X2 = 2, X3 = 3 is ok and X1 = 1, X1 = 1, X1 = 3 not ok

2. Finding a Solution
Finding a global assignment to all of the variables that satisfies all the constraints is hard : NP-Complete
The solution techniques work by cleverly searching through the space of possible assignments of values to variables
If each variable has d values and there are n variables, then we have d possible assignments.

Representations : N = 8 Queens as a CSP
A problem can be represented as a CSP in different ways.

First Representations :
1. We need to know where to place each of the N queens.
2. We could have N variables each of which has as a value 1. . . N^2.
3. The values represent where we will place the i th variable.



This representation has

$64^8 = 281,474,976,710,656$

different possible assignments in the search space.

Second Representation

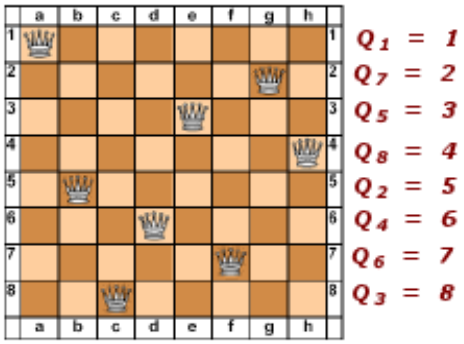
Here we know, we can not place two queens in the same column.

Assign one queen to each column, and then find out the rows where each of these queens is to placed.

We can have N variables: Q1 , ... , QN .

The set of values for each of these variables are {1 , 2 , . . . , N}.

The representation has



$8^8 = 16,777,216$

different possible assignments in the search space.

It is still too large to examine all of them, but definitely a big improvement .

Constraints :

Translate each of individual conditions into a separate constraint.

First Condition : Queens Cannot attack each other

- The Q_i cannot attack Q_j for ($i \neq j$)
- The Q_i is a queen to be placed in column i ,
- The Q_j is a queen to be placed in column j .
- The value of Q_i and Q_j, are the rows the queens are to be placed.

Note : The translation is dependent on the representation we chose.

Second Condition : Queens can attack each other

- Vertically, if they are in the same column . . . ; this is impossible as Q_i and Q_j are placed in different columns
- Horizontally, if they are in the same row . . . ; the we need constraint $Q_i \neq Q_j$.
- Along a diagonal . . . ; they cannot be the same number of columns apart as they are rows apart: we need the constraint j we need the constraint
  $|i - j| \neq |Q\,i - Q\,j|$; (|.| is absolute value)

Representing the Constraints

Between every pair of variables (Q i , Q j) for ($i \neq j$) , the constraint is C_{ij}.

For each C_{ij}. tan assignment of the values to the variables
$Q_i = A$ and $Q_j = B$ satisfies this constant if and only if
$A \neq B$ and $|A - B| \neq |i - j|$.

Solutions

1. A solution to *N - Queens* problem is any assignment of values to the variables $Q_i, \ldots, Q_n$ Constraints can be over any collection of variables.
2. The N – Queens problems, need only binary constraints, ie. Constraints over pair of variables.
3. By simply enumerating and testing all possible assignments, we can recognize a subset of the variables that make a solution impossible. Thus we can largely improve upon different possible assignments in the search space
4. Finally, by expressing the problem as a CSP we have a systematic way of achieving extra efficiency.

Examples of CSPs

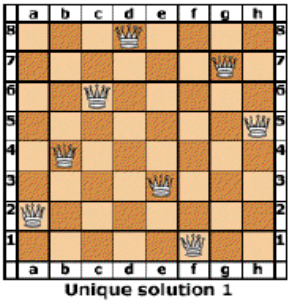
Some poplar puzzles like, the Latin Square, the Eight Queens, and Sudoku are stated below.

- Latin Square Problem : How can one fill an n × n table with n different symbols such that each each symbol occurs exactly once in each row and column ?
  Solutions : The Latin squares for n = 1, 2, 3 and 4 are :

Eight Queens Puzzle Problem

◇ **Eight Queens Puzzle Problem :** How can one put 8 queens on a (8 × 8) chess board such that no queen can attack any other queen ?

Solutions: The puzzle has 92 distinct solutions. If rotations and reflections of the board are counted as one, the puzzle has 12 unique solutions.



Unique solution 1

Sudoku Problem

◇ **Sudoku Problem :** How can one fill a partially completed (9 × 9) grid such that each row, each column, and each of the nine (3 × 3) boxes contains the numbers from 1 to 9.

Problem

	2	6				8	1	
3			7		8			6
4				5				7
	5		1		7		9	
		3	9		5	1		
	4		3		2		5	
1				3				2
5			2		4			9
	3	8				4	6	

Solution

7	***2***	***6***	4	9	3	***8***	***1***	5
3	1	5	***7***	2	***8***	9	4	***6***
4	8	9	6	***5***	1	2	3	***7***
8	***5***	2	***1***	4	***7***	6	***9***	3
6	7	***3***	***9***	8	***5***	***1***	2	4
9	***4***	1	***3***	6	***2***	7	***5***	8
1	9	4	8	***3***	6	5	7	***2***
5	6	7	***2***	1	***4***	3	8	***9***
2	***3***	***8***	5	7	9	***4***	***6***	1

Chapter Review Questions

1. Define the following terms: state, state space, search tree, search node, goal, successor function.
2. Explain why problem formulation must follow goal formulation.
3. Suppose that LEGAL-ACTIONS(s) denotes the set of actions that are legal in a state s, and RESULTS (a, s) denotes the state that results from performing a legal action a in state s. define SUCCESSOR-FN in terms of LEGAL-ACTIONS and RESULTS, and vice versa.
4. Does a finite state space always lead to a finite search tree. What types of state spaces always lead to finite search trees
5. Prove each of the following statements;
   a) Breadth-first search is a special case of uniform-cost search
   b) Breadth-first search, depth-first search and uniform-cost search are special cases of best-first search.
   c) Uniform-cost search is a special case of A* search.
6. Define the following terms; constraint satisfaction problem, constraints, backtracking search, nin-conflicts.

CHAPTER FOUR

Knowledge Representation: Issues

Chapter Objectives

By the end of this chapter the learner should be able to;

- Explain Knowledge Representation.
- Describe the Knowledge Representation Models, typology etc
- Explain Knowledge representation Issues
- Explain Knowledge Representation using Predicate Logic and Rules

4.1. Knowledge and Representation

Knowledge is a description of the world. It determines a system's competence by what it knows.
Types:

- Procedural Knowledge : knowing "how to do something". e.g. "how to drive a car"
- Declarative knowledge ; knowing "that something is true or false". e.g. "that is the speed limit for a car on a motorway"

Knowledge and Representation are distinct entities that play a central but distinguishable roles in intelligent system. Knowledge is a description of the world. It determines a system's competence by what it knows. Representation is the way knowledge is encoded. It defines the performance of a system in doing something. Different types of knowledge require different kinds of representation. The Knowledge Representation models/mechanisms are often based on:

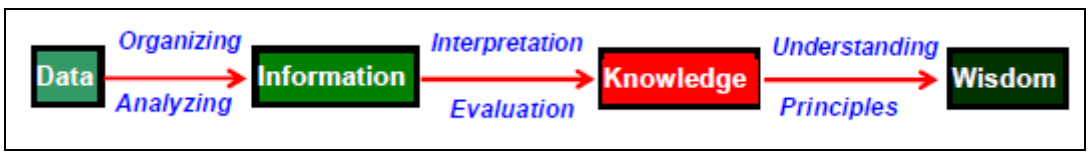
- Logic
- Rules
- Frames
- Semantic Net

4.2. Knowledge

Knowledge is a progression that starts with data which is of limited utility.

- By organizing or analyzing the data, we understand what the data means, and this becomes information.
- The interpretation or evaluation of information yield knowledge.
- An understanding of the principles embodied within the knowledge is wisdom

Knowledge Progression



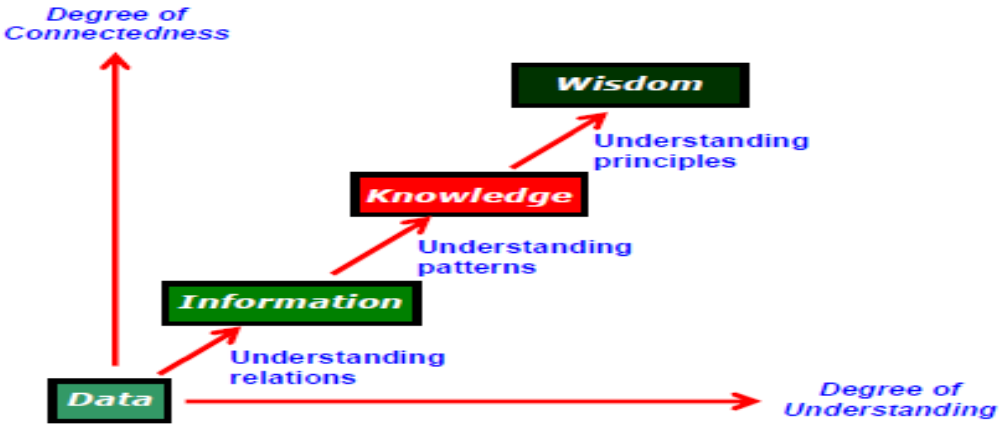
Knowledge Progression

- Data is viewed as collection of disconnected facts. :Example : It is raining.

- Information emerges when relationships among facts are established and understood; Provides answers to "who", "what", "where", and "when". : Example: The temperature dropped 15 degrees and then it started raining.
- Knowledge emerges when relationships among patterns are identified and understood; Provides answers as "how": Example : If the humidity is very high and the temperature drops substantially, then atmospheres is unlikely to hold the moisture, so it rains.
- Wisdom is the pinnacle of understanding, uncovers the principles of relationships that describe patterns. Provides answers as "why": Example : Encompasses understanding of all the interactions that happen between raining, evaporation, air currents, temperature gradients, changes, and raining.

4.2.1. Knowledge Model (Bellinger 1980)

The model tells, that as the degree of connectedness and understanding increase, we progress from data through information and knowledge to wisdom.



Knowledge model

The model represents transitions and understanding.

- the transitions are from data, to information, to knowledge, and finally to wisdom;
- the understanding support the transitions from one stage to the next stage.

The distinctions between data, information, knowledge, and wisdom are not very discrete. They are more like shades of gray rather than black and white (Shedroff, 2001).

- Data and information deal with the past; they are based on the gathering of facts and adding context.
- Knowledge deals with the present that enable us to perform.
- Wisdom deals with the future, acquire vision for what will be, rather than for what is or was.

4.2.3. Knowledge Type

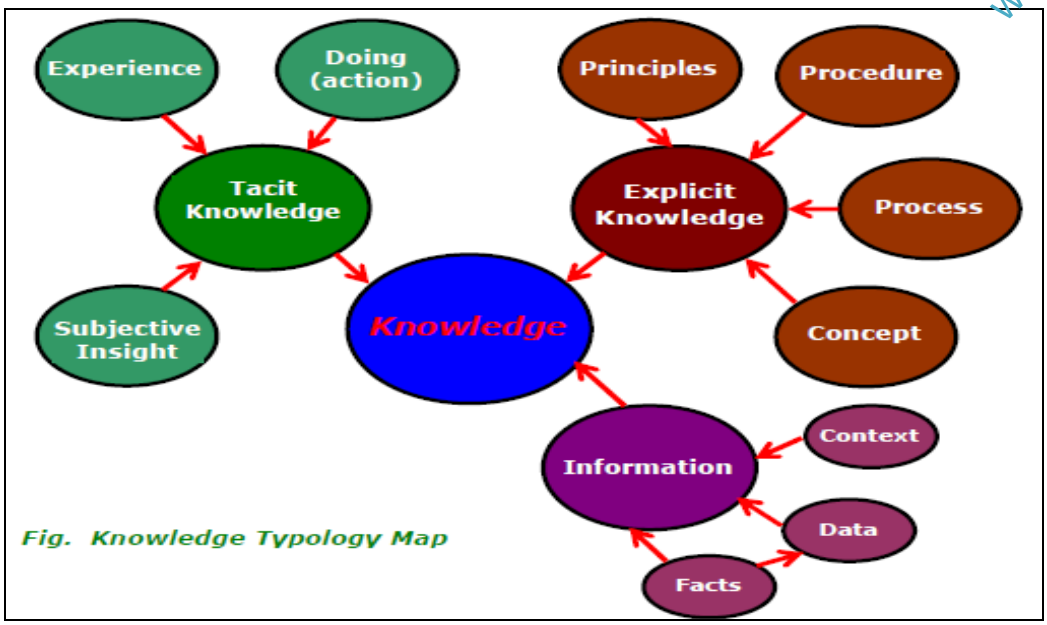
Knowledge is categorized into two major types:

1. Tacit corresponds to informal or implicit type of knowledge,
2. Explicit corresponds to formal type of knowledge.

Tacit knowledge	**Explicit knowledge**
Exists within a human being; it is embodied	Exists outside a human being
Difficult to articulate formally	Can be articulated formally.
Difficult to share/communicate	Can be shared, copied, processed and stored.
Hard to steal or copy	Easy to steal or copy
Drawn from experience, action, subjective insight.	Drawn from artifact of some type as principle, procedure, process, concepts.

4.2.4. Knowledge Typology Map

The map shows that, Tacit knowledge comes from experience, action, subjective insight and Explicit knowledge comes from principle, procedure, process, concepts, via transcribed content or artifact of some type.



Knowledge typology map

- Facts : are data or instance that are specific and unique.
- Concepts : are class of items, words, or ideas that are known by a common name and share common features.
- Processes : are flow of events or activities that describe how things work rather than how to do things.
- Procedures : are series of step-by-step actions and decisions that result in the achievement of a task.
- Principles : are guidelines, rules, and parameters that govern; principles allow to make predictions and draw implications; principles are the basic building blocks of theoretical models (theories).

These artifacts are used in the knowledge creation process to create two types of knowledge: declarative and procedural

4.2.5. Procedural Knowledge and Declarative Knowledge

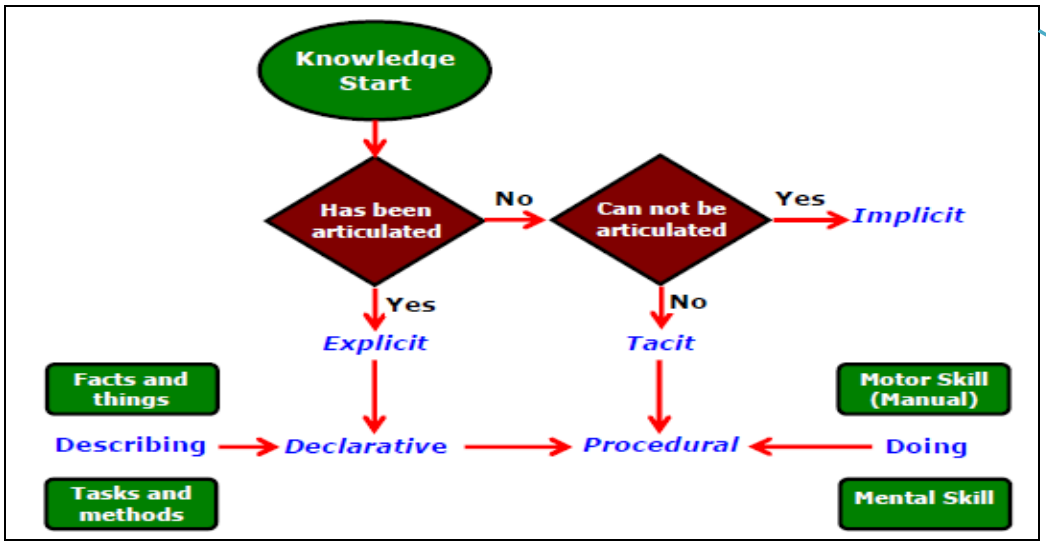
Cognitive psychologists sort knowledge into Declarative and Procedural category and some researchers added Strategic as a third category.

Procedural knowledge Declarative knowledge	Procedural knowledge Declarative knowledge
Procedures, rules; strategies, agendas, models	concepts, objects, facts, propositions, assertions, semantic nets; logic and descriptive models
focuses on tasks that must be performed to reach a particular objective or goal.	refers to representations of objects and events; knowledge about facts. and relationships;
Knowledge about "how to do, something"	Knowledge about "that something is true or false.

All declarative knowledge are explicit knowledge; it is knowledge that can be and has been articulated. The strategic knowledge is thought as a subset of declarative knowledge.

4.2.6. Relationship among Knowledge Type

The relationship among explicit, implicit, tacit, declarative and procedural knowledge are illustrated below.



Relationship among types of knowledge

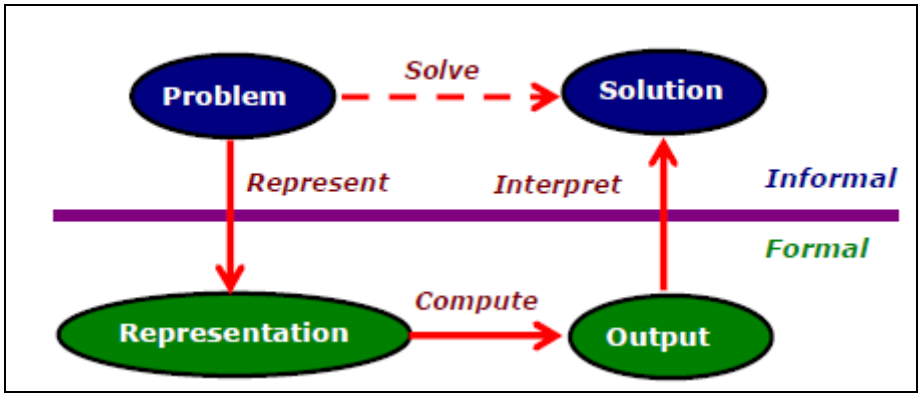
The figure above shows,;

- Declarative knowledge is tied to "describing" and Procedural knowledge is tied to "doing."
- The arrows connecting explicit with declarative and tacit with procedural, indicate the strong relationships exist among them.
- The arrow connecting declarative and procedural indicates that we often develop procedural knowledge as a result of starting with declarative knowledge. i.e., we often "know about" before we "know how".

Therefore, we may view all procedural knowledge as tacit, and all declarative knowledge as explicit.

4.3. Framework of Knowledge Representation

Computer requires a well-defined problem description to process and also provide well-defined acceptable solution. To collect fragments of knowledge we need: first to formulate description in our spoken language and then represent it in formal language so that computer can understand. The computer can then use an algorithm to compute an answer. This process is illustrated below.



Knowledge Representation Framework

The steps are

- The informal formalism of the problem takes place first.

- It is then represented formally and the computer produces an output.
- This output can then be represented in a informally described solution that user understands or checks for consistency.

The Problem solving requires formal knowledge representation, and conversion of informal (implicit) knowledge to formal (explicit) knowledge.

4.3.1. Knowledge Representation

Problem solving mechanism requires large amount of knowledge and some mechanism for manipulating that knowledge.

The Knowledge and the Representation are distinct entities, play a central but distinguishable roles in intelligent system.

- Knowledge is a description of the world; it determines a system's competence by what it knows.
- Representation is the way knowledge is encoded; it defines the system's performance in doing something.

Knowledge representation involves

- need to know about things we want to represent , and
- need some means by which things we can manipulate.

know things to represent	1. Objects – 2. Events 3. Performance 4. Meta-knowledge-	1. facts about objects in the domain 2. actions that occur in the domain 3. knowledge about how to do things 4. knowledge about what we know
need means to manipulate	Requires some formalism	what we represent

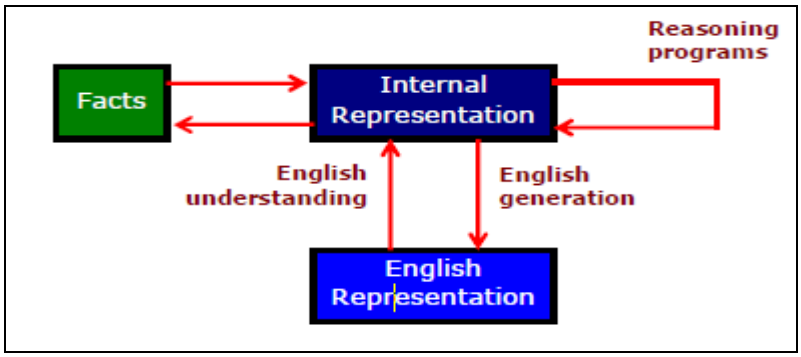
Knowledge representation can be considered at two levels

1. Knowledge level at which facts are described, and
2. Symbol level at which the representations of the objects, defined in terms of symbols, can be manipulated in the programs.

A good representation enables fast and accurate access to knowledge and understanding of the content.

4.3.2. Mapping between Facts and Representation

We need a representation of facts that can be manipulated by a program. Normal English is insufficient, too hard currently for a computer program to draw inferences in natural languages. Thus some symbolic representation is needed. Example Consider an English sentence.



Mapping between Facts and Representation

Facts and representation

Facts	Representation
Spot is a dog	A fact represented in English sentence
dog (Spot)	Using **forward mapping function** the above fact is represented in logic
$\forall x : dog(x) \rightarrow hastail(x)$	A **logical representation** of the fact that "all dogs have tails

Using deductive mechanism we can generate a new representation of Object as;

Hastail (Spot)	A new object representation
Spot has a tail	Using backward mapping function to generate English sentence

4.4. Knowledge Representation System Requirements

A good knowledge representation enables fast and accurate access to knowledge and understanding of the content. A knowledge representation system should have following properties.

1. Representational Adequacy: The ability to represent all kinds of knowledge that are needed in that domain.
2. Inferential Adequacy: The ability to manipulate the representational structures to derive new structure corresponding to new knowledge inferred from old .
3. Inferential Efficiency: The ability to incorporate additional information into the knowledge structure that can be used to focus the attention of the inference mechanisms in the most promising direction.
4. Acquisitional Efficiency: The ability to acquire new knowledge using automatic methods wherever possible rather than reliance on human intervention.

4.5. Knowledge Representation Schemes

There are five types of Knowledge representation which include

1. Relational Knowledge: provides a framework to compare two objects based on equivalent attributes.
2. Inheritable Knowledge is obtained from associated objects. It prescribes a structure in which new objects are created which may inherit all or a subset of attributes from existing objects.
3. Inferential Knowledge: is inferred from objects through relations among objects.
4. Declarative Knowledge: a statement in which knowledge is specified, but the use to which that knowledge is to be put is not given. e.g. laws, people's name; these are facts which can stand alone, not dependent on other knowledge;
5. Procedural Knowledge: a representation in which the control information, to use the knowledge, is embedded in the knowledge itself. e.g. computer programs, directions, and recipes; these indicate specific use or implementation;

4.5.1. Relational Knowledge

Relational knowledge: associates elements of one domain with another. Used to associate elements of one domain with the elements of another domain or set of design constrains.

- Relational knowledge is made up of objects consisting of attributes and their corresponding associated values.
- The results of this knowledge type is a mapping of elements among different domains.

Question: Who is the heaviest player basing on the data in table below?

player	Height	Weight	Bats-Throw
Aaron	6	180	Right-Right
Mays	10	170	Right-Right
Ruth	5	215	Left-Left

4.5.2 Inheritable Knowledge

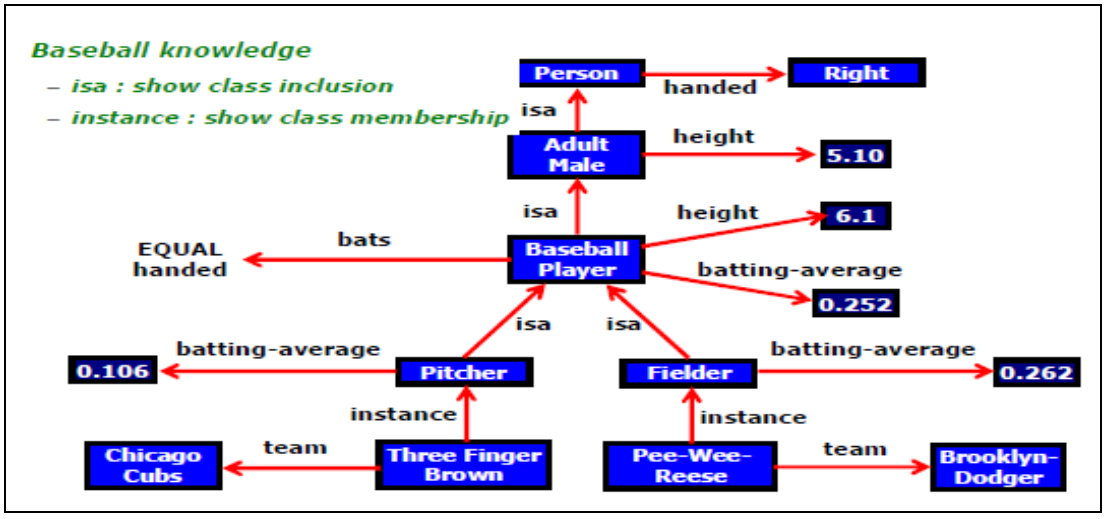
Inheritable knowledge: elements inherit attributes from their parents.

The knowledge is embodied in the design hierarchies found in the functional, physical and process domains. Within the hierarchy, elements inherit attributes from their parents, but in many cases, not all attributes of the parent elements be prescribed to the child elements.

- The basic KR needs to be augmented with inference mechanism, and
- Inheritance is a powerful form of inference, but not adequate.

The KR in hierarchical structure, in figure 8, is called semantic network or a collection of frames or slot-and-filler structure". It shows property inheritance and way for insertion of additional knowledge.

- Property inheritance : Objects/elements of specific classes inherit attributes and values from more general classes.
- Classes are organized in a generalized hierarchy.



Inheritable knowledge

- The directed arrows represent attributes (isa, instance, and team) originating at the object being described and terminating at the object or its value.
- the box nodes represents objects and values of the attributes.

Example: Viewing a node as a frame Baseball-player

isa	Adult-Male
Bates	EQUAL handed
Height	v6.1
Batting-average	0.252

Algorithm: Property Inheritance:
Retrieve a value V for an attribute A of an instance object O. The steps are as follows

- Find object O in the knowledge base.
- If there is a value for the attribute A then report that value.
- Else, see if there is a value for the attribute instance; If not, then fail.
- Else, move to the node corresponding to that value and look for a
- Value for the attribute A; If one is found, report it.
- Else, do until there is no value for the is a attribute or until an answer is found :

4.5.3. Inferential Knowledge

Generates new information from the given information. This new information does not require further data gathering form source, but does require analysis of the given information to generate new knowledge.

- Search usually results from lack of knowledge
- Search explores knowledge alternatives to arrive at the best answer.
- Search algorithms out is a solution (path from the initial state to goal test.

For general-purpose problem solve:

- Given a set of relations and values, one may infer other values or relations.
- In addition to algebraic relations, a predicate logic (mathematical deduction) is used to infer from a set of attributes.
- Inference through predicate logic uses a set of logical operations to relate individual data.
  The symbols used for the logic operations are

" → " (implication), " ¬ " (not), " ∨ " (or), " ∧ " (and), " ∀ " (for all), " ∃ " (there exists).

Example of predicate logic statements

Wonder is a name of a dog	dog (wonder)
All dogs belong to the class of animals :	∀ x : dog (x) → animal(x
All animals either live on land or in water	: ∀ x : animal(x) → live (x, land) V live (x, water)

4.5.4. Declarative/Procedural Knowledge

The difference between Declarative/Procedural knowledge is not very clear.

- Declarative knowledge: Here, the knowledge is based on declarative facts about axioms and e.g axioms are assumed to be true unless a counter example is found to invalidate them. Domains represent the physical world and the perceived functionality. Then the axiom and domains thus simply exists and serve as declarative statements that can stand alone.
- Procedural knowledge: Here the knowledge is a mapping process between domains that specifies what to do when and the representation is of how to make it rather than what it is. The procedural knowledge for example may have inferential efficiency, but no inferential adequacy and acquisitions efficiency. They are represented as small programs that know how to do specific things, how to proceed.

Example: a parser in a natural language has the knowledge that a noun phrase may contain articles, adjectives and nouns. It thus accordingly call routines that know how to process articles, adjectives and nouns.

4.5. Issues in Knowledge Representation

The fundamental goal of Knowledge Representation is to facilitate inferencing (conclusions) from knowledge. The issues that arise while using KR techniques are many. Some of these are explained below

1. Important Attributes: Any attribute of objects so basic that they occur in almost every problem domain?
2. Relationship among attributes: Any important relationship that exists among object attributes ?
3. Choosing Granularity: At what level of detail should the knowledge be represented ?
4. Set of objects: How sets of objects be represented ?
5. Finding Right structure : Given a large amount of knowledge stored, how can relevant parts be accessed

4.6. Important Attributes

There are two attributes "instance" and "isa", that are of general significance. These attributes are important because they support property inheritance.

4.6.1. Relationship among attributes

The attributes we use to describe objects are themselves entities that we represent. The relationship between the attributes of an object independent of specific knowledge they encode, may hold properties like: Inverses, existence in an isa hierarchy, techniques for reasoning about values and single valued attributes.

Inverse

This is about consistency check, while a value is added to one attribute. The entities are related to each other in many different ways. The figure shows attributes (isa, instance, and team), each with a directed arrow, originating at the object being described and terminating either at the object or its value.

There are two ways of realizing this:

1. first, represent both relationships in a single representation; e.g., a logical representation, team(Pee-Wee-Reese, Brooklyn–Dodgers), that can be interpreted as a statement about Pee-Wee-Reese or Brooklyn–Dodger.
2. second, use attributes that focus on a single entity but use them in pairs, one the inverse of the other; for e.g., one, and the other, team = Pee-Wee-Reese

Existence in an isa hierarchy:

This is about generalization-specialization, like, classes of objects and specialized subsets of those classes, there are attributes and specialization of attributes. Example, the attribute height is a specialization of general attribute physical-size which is, in turn, a specialization of physical-attribute. These generalization-specialization relationships are important for attributes because they support inheritance.

Techniques for reasoning about values :

This is about reasoning values of attributes not given explicitly. Several kinds of information are used in reasoning, like,

- height : must be in a unit of length,
- age: of person cannot be greater than the age of person's parents.

The values are often specified when a knowledge base is created.

CHAPTER FIVE

Knowledge Representation Using Predicate Logic

Chapter Objectives

By the end of this chapter the learner should be able to;

- Explain Predicate Logic
- Describe the Knowledge Representation Using Predicate Logic
- Explain Knowledge Representation using Predicate Logic and Rules
- Explain how knowledge may be represented as symbol structures that characterize bits of knowledge about objects, concepts.

5.1. Logic

Logic is concerned with the truth of statements about the world. It includes: Syntax, Semantics and Inference Procedure.

1. Syntax: Specifies the symbols in the language about how they can be combined to form sentences. The facts about the world are represented as sentences in logic
2. Semantic: Specifies how to assign a truth value to a sentence based on its meaning in the world. It Specifies what facts a sentence refers to. A fact is a claim about the world, and it may be TRUE or FALSE.
3. Inference Procedure: Specifies methods for computing new sentences from an existing sentences.

5.2. Logic as a KR Language

- Logic is a language for reasoning, a collection of rules used while doing logical reasoning. Logic is studied as KR languages in artificial intelligence
- Logic is a formal system in which the formulas or sentences have true or false values and the trade off is between which is
    - Expressive enough to represent important objects and relations in a problem domain.
    - Efficient enough in reasoning and answering questions about implicit information in a reasonable amount of time.
- Logics are of different types :
    a) Propositional logic,
    b) Predicate logic,
    c) Temporal logic,
    d) Modal logic,
    e) Description logic etc;

Propositional logic and Predicate logic are fundamental to all logic.

- Propositional Logic is the study of statements and their connectivity.
- Predicate Logic is the study of individuals and their properties.

Assumptions about KR

- Intelligent Behavior can be achieved by manipulation of symbol structures.
- KR languages are designed to facilitate operations over symbol structures, have precise syntax and semantics;

- Syntax tells which expression is legal ?, e.g., red1(car1), red1 car1, car1(red1), red1(car1 & car2) ?
- Semantic tells what an expression means? e.g., property dark red applies to my car.
- Make Inferences, draw new conclusions from existing facts.

5.3. Logic Representation

- Facts are claims about the world that are True or False.
- Representation is an expression (sentence), stands for the objects and relations.
- Sentences can be encoded in a computer program.

To build a Logic-based representation

- User defines a set of primitive symbols and the associated semantics.
- Logic defines ways of putting symbols together so that user can define legal sentences in the language that represent TRUE facts.
- Logic defines ways of inferring new sentences from existing ones
- Sentences - either TRUE or false but not both are called propositions.
- A declarative sentence expresses a statement with a proposition as content; example: the declarative "snow is white" expresses that snow is white; further, "snow is white" expresses that snow is white is TRUE.

5.4. Propositional Logic

A proposition is a statement, which in English would be a declarative sentence. Every proposition is either TRUE or FALSE.

Examples: (a) The sky is blue. (b) Snow is cold, (c) 12 * 12=144

- propositions are sentences , either true or false but not both.
- a sentence is smallest unit in propositional logic.
- if proposition is true, then truth value is "true"
- if proposition is false, then truth value is "false"

Example :

Sentence	*Truth value*	*Proposition (Y/N)*
"Grass is green"	"true"	Yes
"2 + 5 = 5"	"false"	Yes
"Close the door"	-	No
"Is it hot out side ?"	-	No
"x > 2" where is variable	-	No (since x is not defined)
"x = x"	-	No (don't know what is "x" and "="; "3 = 3" or "air is equal to air" or "Water is equal to water" has no meaning)

Propositional logic is also called Propositional calculus, Sentential calculus, or Boolean algebra. Propositional logic tells the ways of joining and/or modifying entire propositions, statements or sentences to form more complicated propositions, statements or sentences, as well as the logical relationships and properties that are derived from the methods of combining or altering statements.

5.5. Statement, Variables and Symbols

- Statement Simple statements (sentences), TRUE or FALSE, that does not contain any other statement as a part, are basic propositions; lower-case letters, p, q, r, are symbols for simple statements. Large, compound or complex statement are constructed from basic propositions by combining them with connectives.
- Connective or Operator: The connectives join simple statements into compounds, and joins compounds into larger compounds.

Connective	*Symbols*					Read as
assertion	P					"p is true"
negation	¬p	~	!		NOT	"p is false"
conjunction	p ∧ q	·	&&	&	AND	"both p and q are true"
disjunction	P v q	||	|		OR	"either p is true, or q is true, or both "
implication	p → q	⊃	⇒		if ..then	"if p is true, then q is true" " p implies q "
equivalence	↔	≡	⇔		if and only if	"p and q are either both true or both false"

Connectives and Symbols in decreasing order of operation priority

- Truth value: The truth value of a statement is its TRUTH or FALSITY.

Example :

p is either *TRUE* or *FALSE*,

~p is either *TRUE* or *FALSE*,

p v q is either *TRUE* or *FALSE*, and so on.

use " T " or " 1 " to mean *TRUE*.

use " F " or " 0 " to mean *FALSE*

- Truth Table : defining the basic connectives
  Example

p	q	¬p	¬q	p ∧ q	p v q	p→q	p ↔ q	q→p
T	T	F	F	T	T	T	T	T
T	F	F	T	F	T	F	F	T
F	T	T	F	F	T	T	F	F
F	F	T	T	F	F	T	T	T

- Tautologies A proposition that is always true is called a tautology. e.g., (P V ┌: P) is always true regardless of the truth value of the proposition P.
- Contradictions: A proposition that is always false is called a contradiction. e.g., (P _ : P) is always false regardless of the truth value of the proposition P.
- Contingencies: A proposition is called a contingency, if that proposition is neither a tautology nor a contradiction e.g., (P _ Q) is a contingency.
- Antecedent, Consequent: In the conditional statements, p ! q ,the
- 1st statement or "if - clause" (here p) is called antecedent ,
  2nd statement or "then - clause" (here q) is called consequent.

- Argument: Any argument can be expressed as a compound statement. Take all the premises, conjoin them, and make that conjunction the antecedent of a conditional and make the conclusion the consequent. This implication statement is called the corresponding conditional of the argument.

Note :
Every argument has a corresponding conditional and every implication statement has a corresponding argument.

Because the corresponding conditional of an argument is a statement,- it is therefore either a tautology, or a contradiction, or a contingency.

- An argument is valid "if and only if" its corresponding conditional is a tautology.
- Two statements are consistent "if and only if" their conjunction is not a contradiction.
- Two statements are logically equivalent "if and only if" their truth table columns are identical; "if and only if" the statement of their equivalence using " ≡ " is a tautology.

The truth tables are adequate to test validity, tautology, contradiction, contingency, consistency, and equivalence.

5.6. Predicate Logic

The propositional logic, is not powerful enough for all types of assertions. We need languages that allow us to describe properties (predicates) of objects, or a relationship among objects represented by the variables

Example:
The assertion "x > 1", where x is a variable, is not a proposition because it is neither true nor false unless value of x is defined.
Consider example:
- All men are mortal.
- Socrates is a man.
- Then Socrates is mortal

These cannot be expressed in propositional logic as a finite and logically valid argument (formula).

In Predicate Logic every complete sentence contains two parts: a subject and a predicate.
- The subject is what (or whom) the sentence is about and
- The predicate tells something about the subject.
- A Predicate is a verb phrase template that describes a property of objects, or a relation among objects represented by the variables. Example: the car Tom is driving is blue. Predicate is blue, describes property of the car
- Predicates are given names; example let B be the name for predicate is blue. The sentence is represented as B(x) , read as x is blue; x represents an arbitrary Object .

Example
A sentence "Judy {runs}".
- The subject is Judy
- Predicate is runs

Predicate, always includes verb, which tells something about the subject. Predicate is a verb phrase template that describes a property of objects, or a relation among objects represented by the variables

5.6.1. Predicate Logic Expressions

The propositional operators combine predicates, like

If (p(....) && (!q(....) || r (....)))

Examples of logic operators : disjunction (OR) and conjunction (AND).

Consider the expression with the respective logic symbols || and &&

x < y || (y < z && z < x)

Which is *true || (true && true) ;*

Applying truth table, found *True*

Assignment for < are 3, 2, 1 for x, y, z and then

the value can be *FALSE* or *TRUE*

3 < 2 || (2 < 1 && 1 < 3)

It is *False*

5.6.2. Predicate Logic Quantifiers

A Predicate with variables (is called atomic formula) can be made a proposition by applying one of the following two operations to each of its variables :

1. Assign a value to the variable; e.g., x > 1, if 3 is assigned to x becomes 3 > 1 , and it then becomes a true statement, hence a proposition.
2. Quantify the variable using a quantifier on formulas of predicate logic (called wff), such as x > 1 or P(x), by using Quantifiers on variables.

Apply Quantifiers on Variables

Variable x: x > 5 is not a proposition, its truth depends upon the value of variable x to reason such statements, x need to be declared

Declaration

format; x : a: declares variable x and

x : a (read as x is an element of set a)

Statement p is a statement about x

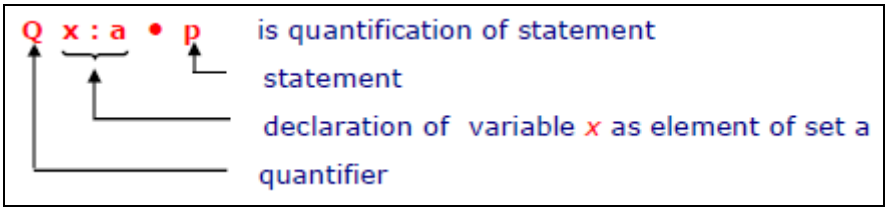


Figure: statement quantifier

Quantifiers are two types:

- universal quantifiers, denoted by ∀ symbol and
- existential quantifiers , denoted by ∃ symbol

5.7. Universe of Discourse

The universe of discourse, also called domain of discourse or universe.

This indicates

- A set of entities that the quantifiers deal.
- Entities can be set of real numbers, set of integers, set of all cars on a parking lot, the set of all students in a classroom etc.
- Universe is thus the domain of the (individual) variables.
- Propositions in the predicate logic are statements on objects of a universe.

Example: About natural numbers for All x, y (x < y or x = y or x > y), there is no need to be more precise and say for All x, y in N, because N is implicit, being the universe of discourse.
About a property that holds for natural numbers but not for real numbers,
it is necessary to qualify what the allowable values of x and y are.

5.8. Universal quantifier ∀ For All "
Universal Quantification allows us to make a statement about a collection of objects.
Universal quantification: ∀ x : a p

- read for all x in a , p holds
- a is universe of discourse
- x is a member of the domain of discourse.
- p is a statement about x

In propositional form it is written as: ∀ x p(x)
read for all x, P(x) holds
for each x, P(x) holds or
for every x, P(x) holds
where P(x) is predicate,
∀ x means all the objects x in the universe
P(x) is true for every object x in the universe

How can you express the English language preposition "All cars have wheels" in preposition logic ?
** "All cars have wheels"*
∀x : car • x has wheel
x P(x)
where p (x) is predicate tells : 'x has wheels'
x is variable for object 'cars' that populate universe of discourse

5.9. Existential quantifier (There Exists)
Existential Quantification allows us to state that an object does exist without naming it.
Existential quantification: x : a p

- there exists an x such that p holds
- a is universe of discourse
- x is a member of the domain of discourse.
- p is a statement about x

In propositional form it is written as : x
read there exists an x such that P(x)
there exists at least one x such that P(x)
where P(x) is predicate,
x means at least one object x in the universe
P(x) is true for at least one object x in the universe

How can you express the English language preposition "Someone loves you" in preposition logic ?

5.10. Formula
In mathematical logic, a formula is a type of abstract object, a token of which is a symbol or string of symbols which may be interpreted as any meaningful unit in a formal language.

- Terms: Defined recursively as variables, or constants, or functions like f (t1, ……., tn), where f is an n-array function symbol, and t1, ……., tn are terms. Applying predicates to terms produce atomic formulas.
- Atomic formulas: An atomic formula (or simply atom) is a formula with no deeper propositional structure, i.e., a formula that contains no logical connectives or a formula that has no strict sub-formulas.
  - Atoms are thus the simplest well-formed formulas of the logic.
  - Compound formulas are formed by combining the atomic formulas using the logical connectives.
  - Well-formed formula ("wi_") is a symbol or string of symbols (a formula) generated by the formal grammar of a formal language
- Example of atomic formula is t1 = t2.
- Example of a compound formula is (((a ∧ b) ∧c) _ ((¬a ∧b) ∧ c)) _ ((a ∧ ¬:b) ∧c))

5.11. Representing IsA and Instance Relationships

The ways, attributes "instance" and "isa", are logically expressed are :
Example : A simple sentence like "Joe is a musician"

- Here "is a" (called IsA) is a way of expressing what logically is called a class-instance relationship between the subjects represented by the terms "Joe" and "musician".
  - ✓ "Joe" is an instance of the class of things called "musician".
  - ✓ "Joe" plays the role of instance,
  - ✓ "musician" plays the role of class in that sentence
- Note : In such a sentence, while for a human there is no confusion, but for computers each relationship have to be defined explicitly.
- This is specified as: [Joe] IsA [Musician] i.e [Instance] IsA [Class]

Computable Functions and Predicates
The objective is to define class of functions C computable in terms of F. This is expressed as C F is explained below using an example : "evaluate factorial n"
Example (1) : A conditional expression to define factorial n ie n!

- Expression: if p1 then e_1 else if p2 then e2 . . . else if pn then en .
  ie. (p1 →e1; p2 →e2, …….pn → en)
  Here p1, p2…… pn are propositional expressions taking the values T or F for true and false respectively.
- The value of (p1 → e1; p2 → e2; ……pn → en) is the value of the e corresponding to the first p that has value T.
- Derive an expressions defining n! , n= 5, recursively by use of the condition expressions

5.12. Computable Functions and Predicates

Example (2) : A conditional expression for triangular functions

- The graph of a well known triangular function is shown below

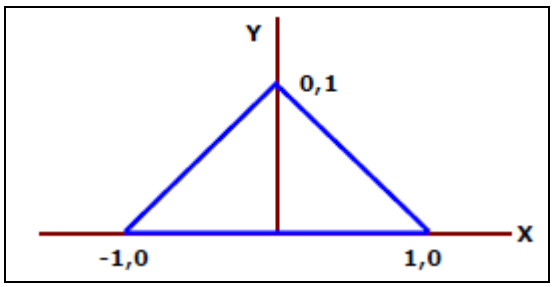


Figure: triangular function

- the conditional expressions for triangular functions are p x p= ($x < 0 \rightarrow -x$; $x \geq 0 \rightarrow x$)
- The triangular function of the above graph is represented by the conditional expression is
  tri(x) = ($x \leq -1 \rightarrow 0$, $x \leq 0 \rightarrow -x$, $x \leq 1 \rightarrow x$; $x > 1 \rightarrow 0$)

5.13. Resolution

- Resolution is a procedure used in proving that arguments which are expressible in predicate logic are correct.
- Resolution lead to refute a theorem-proving technique for sentences in propositional logic and first-order logic
- Resolution is a rule of inference.
- Resolution is a computerized theorem prover.
- Resolution is so far only defined for Propositional Logic. The strategy is that the Resolution techniques of Propositional logic be adopted in Predicate Logic.

CHAPTER SIX

Knowledge Representation Using Rules

Chapter Objectives

By the end of this chapter the learner should be able to;

- Explain rules used in Rule –based production systems
- Explain Logic Programming
- Describe logic programming program components
- Explain Programming paradigms : Models of Computation

6.1. Introduction

The other most popular approach to Knowledge representation is to use production rules, sometimes called IF-THEN rules. The production rules are simple but powerful forms of knowledge representation providing the flexibility of combining declarative and procedural representation for using them in a unified form.

Examples of production rules

- IF condition THEN action
- IF premise THEN conclusion
- IF proposition p1 and proposition p2 are true THEN proposition p3 is true

The advantages of production rules

- They are modular
- Each rule define a small and independent piece of knowledge
- New rules may be added and old ones deleted
- Rules are usually independently of other rules.

The production rules as knowledge representation mechanism are used in the design of many "Rule-based systems" also called "Production systems" .

6.2. Types of Rules

Three major types of rules used in the Rule-based production systems

1. Knowledge Declarative Rules : These rules state all the facts and relationships about a problem.
   Example IF inflation rate declines
   THEN the price of gold goes down.
   These rules are a part of the knowledge base.
2. Inference Procedural Rules: These rules advise on how to solve a problem, while certain facts are known.
   Example IF the data needed is not in the system
   THEN request it from the user.
   These rules are part of the inference engine.
3. Meta rules: These are rules for making rules. Meta-rules reason about which rules should be considered for ring.

Example IF the rules which do not mention the current goal in their premise,
AND there are rules which do mention the current goal in their premise,
THEN the former rule should be used in preference to the latter.

- Meta-rules direct reasoning rather than actually performing reasoning.
- Meta-rules specify which rules should be considered and in which order they should be invoked.

6.3. Procedural Knowledge, Declarative Knowledge and Rules

Procedural Knowledge: knowing 'how to do'

- Includes : Rules, strategies, agendas, procedures, models. These explains what to do in order to reach a certain conclusion. e.g., Rule: To determine if Peter or Robert is older, first and their ages.
- It is knowledge about how to do something. It manifests itself in the doing of something, e.g., manual or mental skills cannot reduce to words. It is held by individuals in a way which does not allow it to be communicated directly to other individuals.
- Accepts a description of the steps of a task or procedure. It Looks similar to declarative knowledge, except that tasks or methods are being described instead of facts or things.

Declarative Knowledge: knowing 'what', knowing 'that'

- Includes : Concepts, objects, facts, propositions, assertions, models. wisdom;
- It is knowledge about facts and relationships, that
  - can be expressed in simple and clear statements,
  - can be added and modified without difficulty.
- Declarative knowledge and explicit knowledge are articulated knowledge and may be treated as synonyms for most practical purposes.
- Declarative knowledge is represented in a format that can be manipulated, decomposed and analyzed independent of its content.

Comparison between Procedural and Declarative Knowledge:

Procedural Knowledge	Declarative Knowledge
Hard to debug	Easy to validate
Black box	White box
Obscure	Explicit
Process oriented	Data - oriented
Extension may effect stability	Extension is easy
Fast , direct execution Simple data type can be used	Slow (requires interpretation) May require high level data type
Representations in the form of sets of rules, organized into routines and subroutines.	Representations in the form of production system, the entire set of rules for executing the task.

6.4. Procedural and Declarative Language

Procedural Language	Declarative Language
Basic, C++, Cobol, etc.	SQL
Most work is done by interpreter of the languages	Most work done by Data Engine within the DBMS
For one task many lines of code	For one task one SQL statement
Programmer must be skilled in translating the objective into lines of procedural code	Programmer must be skilled in clearly stating the objective as a SQL statement
Requires minimum of management around the actual data	Relies on SQL-enabled DBMS to hold the data and execute the SQL statement .
Programmer understands and has access to each step of the code	Programmer has no interaction with the execution of the SQL statement
Data exposed to programmer during execution of the code	Programmer receives data at end as an entire set
More susceptible to failure due to changes in the data structure	More resistant to changes in the data structure
Traditionally faster, but that is changing	Originally slower, but now setting speed records
Code of procedure tightly linked to front end	Same SQL statements will work with most front ends Code loosely linked to front end.
Code tightly integrated with structure of the data store	Code loosely linked to structure of data; DBMS handles structural issues
Programmer works with a pointer or cursor	Programmer not concerned with positioning

6.5. Logic Programming

Logic programming offers a formalism for specifying a computation in terms of logical relations between entities.

- logic program is a collection of logic statements
- programmer describes all relevant logical relationships between the various entities
- computation determines whether or not, a particular conclusion follows from those logical statements.

6.5.1. Characteristics of Logic program

- Logic program is characterized by set of relations and inferences
- The program consists of a set of axioms and a goal statement.
- The Rules of inference determine whether the axioms are sufficient to ensure the truth of the goal statement.
- The execution of a logic program corresponds to the construction of a proof of the goal statement from the axioms.
- the Programmer specify basic logical relationships, does not specify the manner in which inference rules are applied. Thus Logic + Control = Algorithms

Examples of Logic Statements

- Statement: A grand-parent is a parent of a parent.
- Statement expressed in more closely related logic terms as: A person is a grand-parent if she/he has a child and that child is a parent.
- Statement expressed in first order logic as
  (forall)x : grand parent(x) ← (there exist)y; z : parent(x; y)&parent(y; z)

6.5.2. Logic programming Language

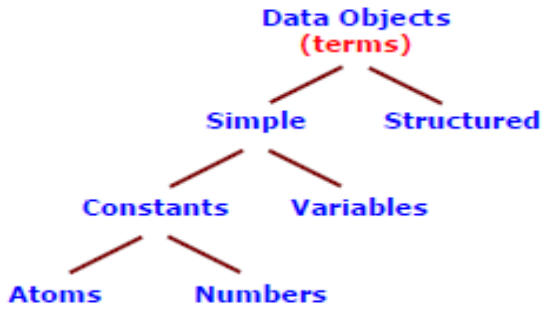
- A programming language includes
  - the syntax
  - the semantics of programs and
  - the computational model
- It mostly uses the procedural paradigm. The program specifies a computation by saying "how" it is to be performed. FORTRAN, C, and object-oriented languages fall under this general approach.
- Another paradigm is declarative. The program specifies a computation by giving the properties of a correct answer. Prolog and logic data language (LDL) are examples of declarative languages, emphasize the logical properties of a computation.
- PROLOG is the most popular Logic programming system.

6.6. Syntax and Terminology (relevant to Prolog programs)

In any language, the formation of components (expressions, statements, etc.), is guided by syntactic rules. The components are divided into two parts: (A) data components and (B) program components.

6.6.1. Data Components

Data components are collection of data objects that follow hierarchy.



Prolog Data Component

Data objects
The data objects of any kind is called a term.
Term : examples

- Constants: denote elements such as integers, floating point, atoms.
- Variables: denote a single but unspecified element; symbols for variables begin with an uppercase letter or an underscore.
- Compound terms: comprise a function and sequence of one or more compound terms called arguments
- Ground and non-ground: Terms are ground if they contain no variables; otherwise they are non-ground. Goals are atoms or compound terms, and are generally non-ground.

Simple data objects: Atoms, Numbers, Variables
Atoms

- a lower-case letter, possibly followed by other letters (either case), digits, and underscore character
- a string of special characters such as: +;□; _; =; =;&
- a string of any characters enclosed within single quotes. e.g. 'ABC'
- following are also atoms ! ; [] {} !

Numbers

- applications involving heavy numerical calculations are rarely written in Prolog.
- Integer representation
- a string of any characters enclosed within single quotes. e.g. 'ABC'
- real numbers written in standard or scientific notation,

Variables.

- begins by a capital letter, possibly followed by other letters (either case), digits, and underscore character.

Structured data objects: General Structures, Special Structures
General Structures

- a structured term is syntactically formed by a function and a list of arguments
- function is an atom.
- list of arguments appears between parentheses.
- arguments are separated by a comma.
- each argument is a term (i.e., any Prolog data object).
- the number of arguments of a structured term is called its arity. e.g greaterThan(9, 6) f(a, g(b, c), h(d)) plus(2, 3, 5)
- Note : a structure in Prolog is a mechanism for combining terms together, like integers 2, 3, 5 are combined with the function plus.

Special Structures:

- In Prolog this is an ordered collection of terms is called a list . I Lists are structured terms and Prolog offers a convenient notation to represent them:
- Empty list is denoted by the atom [ ].
- Non-empty list carries element(s) between square brackets, separating elements by comma. e.g [bach, bee] [apples, oranges, grapes]

6.6.2. Program Components

A Prolog program is a collection of predicates or rules. A predicate establishes a relationships between objects.
a). Clause, Predicate, Sentence, Subject

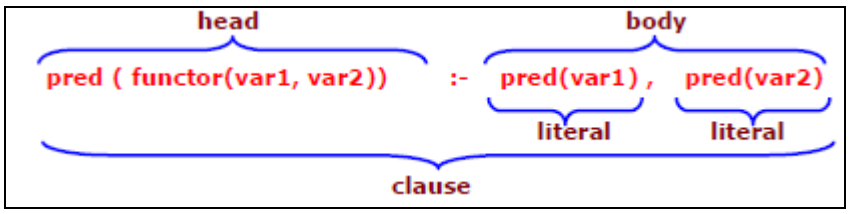
- Clause is a collection of grammatically-related words.
- Predicate is composed of one or more clauses. Clauses are the building blocks of sentences; every sentence contains one or more clauses.
- A Complete Sentence has two parts: subject and predicate. subject is what (or whom) the sentence is about. predicate tells something about the subject.
- Example 1: "cows eat grass". It is a clause, because it contains the subject "cows" and the predicate "eat grass."
- Example 2: "cows eating grass are visible from highway". This is a complete clause. The subject "cows eating grass" and the predicate "are visible from the highway" makes complete thought.

b). Predicates & Clause
Syntactically a predicate is composed of one or more clauses.

- The general form of clauses is : < left □ hand □ side > :- < right □ hand □ side >.

where LHS is a single goal called "goal" and RHS is composed of one or more goals, separated by commas, called "sub-goals" of the goal on left-hand side.

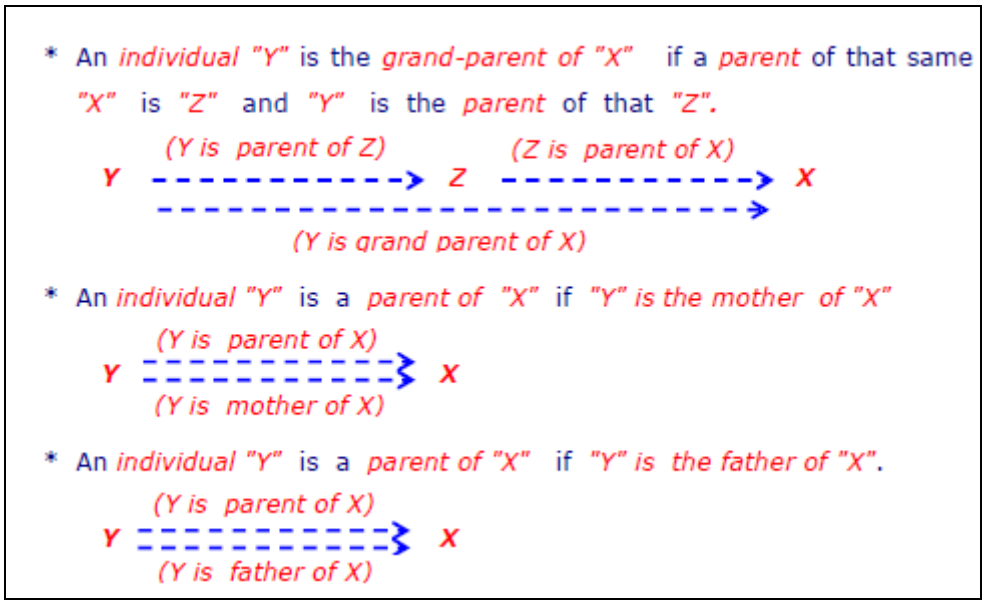
- The structure of a clause in logic program



Structure of a clause

Example
grand parent(X;Y) : parent(X; Z); parent(Z;Y).
parent(X; Y) : mother(X; Y).
parent(X; Y) : father (X;Y).

Interpretation:
A clause specifies the conditional truth of the goal on the LHS; i.e., goal on LHS is assumed to be true if the sub-goals on RHS are all true.



c). Unit Clause - a special Case
Unlike the previous example of conditional truth, one often encounters unconditional relationships that hold.

- Prolog the clauses that are unconditionally true are called unit clause or fact
  Example : Unconditionally relationships say 'Y' is the father of 'X' is unconditionally true. This relationship as a Prolog clause is :father (X;Y) : true.
  Interpreted as relationship of father between Y and X is always true; or simply stated as Y is father of X
- Goal true is built-in in Prolog and always holds.
- Prolog offers a simpler syntax to express unit clause or fact father(X, Y) i.e the : true part is simply omitted.

d). Queries

In Prolog the queries are statements called directive. A special case of directives, are called queries. Syntactically, directives are clauses with an empty left-hand side.
Example : *? - grandparent(X;W).*

This query is interpreted as :
Who is a grandparent of X ?
By issuing queries, Prolog tries to establish the validity of specific relationships. The result of executing a query is either success or failure

- Success, means the goals specified in the query holds according to the facts and rules of the program
- Failure, means the goals specified in the query does not hold according to the facts and rules of the program.

6.7. Programming paradigms: Models of Computation

A complete description of a programming language includes the computational model, syntax, semantics, and pragmatic considerations that shape the language.

- Models of Computation : A computational model is a collection of values and operations, while computation is the application of a sequence of operations to a value to yield another value.
- There are three basic computational models :
  (a) Imperative,
  (b) Functional, and
  (c) Logic.

In addition to these, there are two programming paradigms (concurrent and object-oriented programming). While, they are not models of computation, they rank in importance with computational models.

6.7.1. Imperative Model

- The Imperative model of computation, consists of a state and an operation of assignment which is used to modify the state. Programs consist of sequences of commands. The computations are changes in the state.
  Example 1 : *Linear function: A linear function y = 2x + 3 can be written as Y := 2:X + 3*
- The implementation determines the value of X in the state and then create a new state, which differs from the old state. The value of Y in the new state is the value that 2:X + 3 had in the old state.
- The imperative model is closest to the hardware model on which programs are executed, that makes it most efficient model in terms of execution time.

6.7.2. Functional model

- The Functional model of computation, consists of a set of values, functions, and the operation of functions. The functions may be named and may be composed with other functions. They can take other functions as arguments and return results. The programs consist of definitions of functions. The computations are application of functions to values.
  - ✓ Example 1 : Linear function y = 2x + 3 can be defined as : f (x) = 2:x + 3
  - ✓ Example 2 : Determine a value for Circumference. Assigned a value to Radius, that determines a value for Circumference.
  - ✓ Circumference = 2piradius where pi = 3:14
- Functional models are developed over many years. The notations and methods form the base upon which problem solving methodologies rest.

6.7.3. Logic model :

The logic model of computation is based on relations and logical inference.
Programs consist of definitions of relations. Computations are inferences (is a proof).

- Example 1 : Linear function
  A linear function y = 2x + 3 can be represented as : f (X; Y) if Y is 2 _ X + 3.

- Example 2: Determine a value for Circumference.
  The earlier circumference computation can be represented as:
  Circle(R; C) if Pi = 3:14 and C = 2 _ pi _ R.
  The function is represented as a relation between radius R and circumference C.

6.8. Forward versus Backward Reasoning

Rule-Based system architecture consists a set of rules, a set of facts, and an inference engine. The need is to find what new facts can be derived.
Given a set of rules, there are essentially two ways to generate new knowledge: one, forward chaining and the other, backward chaining.

- Forward chaining: also called data driven. It starts with the facts, and sees what rules apply.
- Backward chaining: also called goal driven. It starts with something to find out, and looks for rules that will help in answering it.

- **Example 1 :**

```
Rule R1 :   IF hot AND smoky   THEN fire
Rule R2 :   IF alarm_beeps     THEN smoky
Rule R3 :   IF fire            THEN switch_on_sprinklers
Fact F1 :   alarm_beeps   [Given]
Fact F2 :   hot           [Given]
```

- **Example 2 :**

```
Rule R1 :   IF hot AND smoky   THEN ADD fire
Rule R2 :   IF alarm_beeps     THEN ADD smoky
Rule R3 :   IF fire            THEN ADD switch_on_sprinklers
Fact F1 :   alarm_beeps   [Given]
Fact F2 :   hot           [Given]
```

- **Example 3 : A typical Forward Chaining**

Rule R1 :	IF hot AND smoky	THEN ADD fire
Rule R2 :	IF alarm_beeps	THEN ADD smoky
Rule R3 :	If fire	THEN ADD switch_on_sprinklers
Fact F1 :	alarm_beeps	[Given]
Fact F2 :	hot	[Given]
Fact F4 :	smoky	[from F1 by R2]
Fact F2 :	fire	[from F2, F4 by R1]
Fact F6 :	switch_on_sprinklers	[from F4 by R3]

- **Example 4 : A typical Backward Chaining**

Rule R1 :	IF hot AND smoky	THEN fire
Rule R2 :	IF alarm_beeps	THEN smoky
Rule R3 :	If _re	THEN switch_on_sprinklers
Fact F1 :	hot	[Given]
Fact F2 :	alarm_beeps	[Given]
Goal :	Should I switch sprinklers on?	

Chapter Review Question

1. Using examples explain the following terms as applied to rule based programming
   a) Forward chaining
   b) Conflict Resolution Strategy
   c) Alternative to Conflict Resolution
   d) Backward chaining
   e) Control Knowledge

2. Explain the following conflict resolution mechanism as applied to rule based logic programming
   a) Refractory
   b) Recency
   c) Specificity

3. Read and Make a brief notes on the following Knowledge Representation schemes
   a) Semantic Networks
   b) Frames

CHAPTER SEVEN

Reasoning Systems

Chapter Objectives

By the end of this chapter the learner should be able to;

- Describe Reasoning, Formal Logic and Uncertainty
- Explain Symbolic Reasoning
- Explain Statistical Reasoning

7.1 Reasoning

Reasoning is the act of deriving a conclusion from certain premises using a given methodology. It is a process of thinking, and it is about logical arguing and drawing inference. A knowledge system must reason, if it is requires to do something which has not been told explicitly. For reasoning, the system must find out what it needs to know from what it already knows.

Human Reasoning

A human being can reason in three different areas;

1. Mathematical reasoning
2. Logical reasoning - deduction, inductive and abduction.
3. Non-logical reasoning- linguistic, languages

- These areas are in every human being but the ability level depends on education, environment and genetics
- The IQ (Intelligence Quotient) is the summation of mathematical reasoning skills and logical reasoning.
- The EQ (Emotional Quotient) depends mostly on non-logical reasoning capability

7.2. Logical Reasoning

- Logic reasoning is the process of drawing conclusions from facts using rules of inferences.
- Logic is divided into formal (symbolic) logic and informal logic.
  - symbolic logic is the study of symbolic abstraction (construct) that capture the formal features of logical inference by a formal system
    Formal systems has two components, a formal language and a set of inference rule
    Formal systems has axioms; An axiom is a sentence which is always true within the system.
    Sentences are derived from the system's axioms.
    Theorems are rules of derivations
  - Informal logic: Informal logic is the study of natural language arguments

7.2.1 Formal logic

- It is the study of inference with purely formal content.
  Examples- Proposition logic and Predicate logic
- Here logic arguments are set of rules for manipulating symbol and two types of rules exists;
  - ✓ Syntax rules: how to build meaningful expressions
  - ✓ Inference rules; how to obtain true formulas from other true formulas

- Logic also need semantics which explain how to assign meaning to expressions

7.2.2. Informal logic

- Informal logic is the study of natural language arguments.
- It involves the analysis of arguments structures in ordinary language.
- The focus lies in distinguishing good arguments (valid) from bad arguments(invalid).
- Properties of Formal systems and its elements

Properties of formal systems

- Consistency: Systems theorem do not contradict.
- Soundness: systems' rules of derivation will never infer anything false as long as we start with only true facts.
- Completeness: There is no true sentences in the system that cannot be proved using the derivation rules of the system

Formal systems consist of the following elements

- A finite set of symbols for constructing formulae
- A grammar providing a way of constructing well-formed formulae (wff) A set of axioms; each axioms has to be a wff.
- A set of inference rules.
- A set of theorems
- A well-formed formulae (wff) is any string generated by a grammar.

7.2.3. Formal Language

- A formal language is a collection of words or collection of sentences. In computer science a formal language is defined by a precise mathematical or machine process able formula.
- A formal language L is defined by a set F of finite-length sequences of elements drawn from a specified finite set A of symbols
- If A is words then the set A is called an Alphabet of L and the elements of F are called words
- If A is sentence then the set A is called the Lexicon or vocabulary of F and the elements of F are then called sentences.

7.3. Uncertain Reasoning

- The world is an uncertain place which often makes knowledge imperfect which causes uncertainty and therefore reasoning must be able to operate under uncertainty.
- AI systems must have the ability to reason under conditions of uncertainty

Uncertainties	Desired action
Incompleteness of Knowledge	Compensate for lack of knowledge
Inconsistencies in Knowledge	Resolve ambiguities and contradictions
Changing knowledge	Update the knowledge base over time

7.3.1. Monotonic Logic

- A logic is monotonic if the truth of a proposition does not change when new information (axioms) are added
- Most formal logics have a monotonic consequence relation meaning that adding a formula to a theory never produces a reduction of its set of consequences.

7.3.2. Non-Monotonic logic

- A monotonic logic cannot handle
  - ✓ Reasoning by default because consequences may be derived only because of lack of evidence of the contrary.
  - ✓ Abductive reasoning because consequences are only deduced as most likely explanation.
  - ✓ Belief revision because new knowledge may contradict old beliefs
- A non-monotonic logic is a formal logic whose consequence relation is not monotonic.
- A logic is non-monotonic if the truth of a proposition may change when new information (axioms) are added.
  - ✓ It allows a statement to be retracted;
  - ✓ Used to formalize believable reasoning
- Non-monotonic reasoning is concerned with consistency and inconsistency is resolved by removing the relevant conclusion(s) derived by default rules

7.4. Methods of Reasoning

1. Deduction: In this kind of reasoning, when determining a conclusion, we use the rules and its precondition to make a conclusion e.g When it rains, the grass gets wet. It rains, thus the grass is wet. Deduction involves;
   - ✓ Applying a general principle to a special case.
   - ✓ Using theory to make predictions
   - ✓ Usage: Inference engines, Theorem provers
2. Induction: Determining the rule by learning the rule after numerous examples of conclusions following the precondition. e.g The grass has been wet every time it has rained. Thus when it rains, the grass gets wet. I involve;
   - ✓ Deriving a general principle from a special cases.
   - ✓ From observations to generalization to knowledge.
   - ✓ Usage: Neural nets, Bayesian Nets, Pattern recognition

3. Abduction: Means determining the precondition; It is using the conclusion and the rule to support that the precondition could explain the conclusion. e.g "When it rains, the grass gets wet. The grass is wet, it must have rained". It involves;
   - Guessing that some general principles that relate a given pattern of cases.
   - Extract hypotheses to form a tentative theory
   - Usage: Knowledge discovery, Statistical methods, data mining

4. Analogy: Illustration of an idea by means of a more familiar idea that is similar to it in some significant features.
   - ✓ Finding a common pattern in different cases.
   - ✓ Usage: Matching labels, matching transformations.

7.5. Sources of Uncertainty in Reasoning

In many problem domains, it is not possible to create complete, consistent models of the world and therefore agents (people) must act in uncertain worlds. The agent must make rational decisions even when there is not enough information to prove that an action will work.

1. Uncertainty is omnipresent because of
   - ✓ Incompleteness
   - ✓ Incorrectness.

2. Uncertainty in Data or Expert knowledge
   - ✓ Data derived from default/ assumptions
   - ✓ Inconsistency between knowledge from different experts
   - ✓ Best Guesses
3. Uncertainty in Knowledge representation.
   - ✓ Restricted model of the real system
   - ✓ Limited expressiveness of the representation mechanism.
4. Uncertainty in Rules or Inference process
   - ✓ Incomplete because too many conditions to be explicitly enumerated.
   - ✓ Incomplete because some conditions are unknown.
   - ✓ Conflict resolution.

7.6. Reasoning and Knowledge Representation

Reasoning to some extent depends on the way knowledge is represented.
A good knowledge representation scheme allows easy, natural and credible reasoning.
Reasoning methods are broadly identified as:

1. Formal reasoning: using basic rules of inference with knowledge representation.
2. Procedural reasoning: uses procedures that specify how to perhaps solve a sub problem.
3. Reasoning by analogy: This is what Human do, but more difficult for AI systems.
4. Generalization and abstraction: This is also as Human do, are basically learning and understanding methods.
5. Meta-level reasoning: Uses knowledge about what we know and ordering them as per importance.

Note: What ever may be the reasoning methods, the AI model must be able to reason under condition of uncertainty.

Reasoning Approaches

1. Symbolic Reasoning:
2. Statistical reasoning
3. Fuzzy logic reasoning

7.7. Symbolic Reasoning

It involves integrating the power of symbolic mathematical tools with suitable proof technology. A Mathematical Reasoning which states that if a conclusion follows from a given premises A, B, C, ... then it also follows from any larger set of premises, as long as the original premises A, B, C, ... are included.
Human Reasoning is not monotonic, people reach to conclusions only tentatively based on partial or incomplete information, which reserve the right to retract those conclusions while they learn new facts.

7.7.1. Non-Monotonic Reasoning

Non-Monotonic reasoning attempts to formalize reasoning with
incomplete information
Non-Monotonic reasoning is of three different types

- Default reasoning
- Circumscription
- Truth Maintenance Systems.

7.7.2. Default Reasoning

In this type of reasoning a conclusion is drawn based on what is most likely to be true. There are two types of default reasoning that is Non-Monotonic logic and Default logic

- Non-Monotonic logic: The truth of a proposition may change when new information are added and the logic may be built to allow the statement to be retracted.
  Non-Monotonic logic: is predicate logic with one extension called model operator (M) which means consistent with everything we know
- Default logic: Default logic initiates a new inference rule :A:BC where
  - A is known as the prerequisite.
  - B as the justification.
  - C as the consequent

  Read the above inference rule as: If A, and if it is consistent with the rest of what is known to assume that B, then conclude that C.
  The rule says that given the prerequisite, the consequent can be inferred, provided it is consistent with the rest of the data
  Example: Rule that " birds typically y" would be represented as

  bird(x):flies(x)
  flies(x)

  which says if x is a bird and the claim that x flies is consistent with what we know, then infer that x flies.

  The idea behind non-monotonic reasoning is to reason with first order logic, and if an inference can not be obtained then use the set of default rules available within the first order formulation.

Applying default rules

When applying default rules, first check their justification for consistency, with not only the initial data but also with the consequents of any other rules that may be applied. The application of one rule may thus block the application of other rule.
This problem is solved by applying the concept of default theory. Default Theory: It consists of a set of W and a set of default rules D.
An extension from W by applying as many rues of D as possible without (together with the rules of deductive inference) inconsistency.

Default Theory

- Note: D the set of default rules has a unique syntax of the form

$$\frac{\alpha(\overrightarrow{x}):E\beta(\overrightarrow{x})}{\gamma(\overrightarrow{x})} \text{ where}$$

$\alpha(\overrightarrow{x})$: is the prerequisite of the default rule
$E\beta(\overrightarrow{x})$: is the consistency test of the default rule.
$\gamma(\overrightarrow{x})$: is the consequent of the default rule.

- The rule can be read as.
  For all individual $x_1 \ldots x_m$ if $\alpha(\overrightarrow{x})$ is a believed and if each of $E\beta(\overrightarrow{x})$ is consistent with the beliefs Then $\gamma(\overrightarrow{x})$ may be believed.

Example
A Default rule says "Typically an African adult owns a house.

African(x)∧Adult(x):M((∃y):hous(y) ∧owns(x;y))
((∃y):house(y)_∧owns(x;y))

- The above rule is only accessed if we want to know whether or not Rose owns a house then an answer can not be deduced from our current beliefs
- The default rule is applicable if we can prove that from our beliefs that Rose is an African and an adult, and the believing that there is some car that is owned by Rose does not lead to and inconsistency.
- If these two sets of premises are satisfied, then the rule states that we can conclude that Rose owns a car.

7.7.3. Circumscription

Circumscription is non-monotonic logic which fomalizes the common sense assumption. Circumscription is a formalized rule of conjecture (guess) that can be used along with the rules of inference of first order logic. The idea is to specify particular predicates that are assumed to be " as false a possible" that is, false for every object except those for which they are known to be true. Circumscription involves formulating rules of thumb with "abnormality" predicates and then restricting the extension of these predicates Circumscribing them, so that they apply to only those things to which they are currently known.

Example: Take the case of a Bird Tweety

- The rule of thumb is that "birds typically y" is the condition. The predicate "abnormal" signifies abnormality with respect to flying ability.
- Observe that the rule ∇ x(Bird(x)& ¬abnormal(x) →Flies(x)) does not allow us to infer that Tweety flies, since we do not know that it is abnormal with respect to flying ability.
- If we add rules which circumscribe the abnormality predicate to which they are currently known say "Bird Tweety" then the inference can be drawn.

7.8. Reasoning Maintenance Systems

- Reasoning Maintenance systems (RMS) is a critical part of a reasoning system.
- Its purpose is to ensure that inference made by the reasoning systems (RS) are valid.
- The RS provides the RMS with information about each inference it perform, and in return the RMS provides the RS with information about the whole set of inference.
- Implementation of RMS include Truth Maintenance systems (TMS) and Assumption-based Truth Maintenance systems (ATMS)
- The TMS maintain the consistency of a knowledge base as soon as new knowledge is added. It considers one state at a time so it is not possible to manipulate environment.
- ATMS is intended to maintain multiple environments.

7.9. Truth Maintenance Systems

Support default reasoning In the absence of any firm knowledge, in many situations, we want to reason from default.

Example: If Tweety is a bird, then until told otherwise we assume that Tweety flies and for justification we use the fact that Tweety is a bird and the assumption that birds fly.

A truth maintenance system maintains consistency in knowledge representation of a knowledge base.
The functions of TMS are to :

- **Provide justifications for conclusions**
  When a problem solving system gives an answer to a user's query, an explanation of that answer is required;
  Example : An advice to a stockbroker is supported by an explanation of the reasons for that advice. This is constructed by the Inference Engine (IE) by tracing the justification of the assertion.
- **Recognize inconsistencies**
  The Inference Engine (IE) may tell the TMS that some sentences are contradictory. Then, TMS may find that all those sentences are believed true, and reports to the IE which can eliminate the inconsistencies by determining the assumptions used and changing them appropriately.
  Example : A statement that either Abbott, or Babbitt, or Cabot is guilty together with other statements that Abbott is not guilty, Babbitt is not guilty, and Cabot is not guilty, form a contradiction.

7.10. Implementation Issues

The issues and weaknesses related to implementation of non-monotonic reasoning in problem solving

- How to derive exactly those non-monotonic conclusion that are relevant to solving the problem at hand while not
- How to update our knowledge incrementally as problem solving progresses
- How to over come the problem where more than one interpretation of the known fact is qualified or approved by the available inference rules
- In general the theories are not computationally effective, decidable or semi decidable

7.11. Statistic Reasoning

- The issues and weaknesses related to implementation of non-monotonic reasoning in problem solving
- In logic based approaches, we have assumed that every thing is either believed false or believed true.
- However, it is often useful to represent the fact that we believe such that something is probably true, or true with some probability
- This is useful for dealing with problems where there is randomness and unpredictability and also dealing with problems where we could, if we had sufficient information, work out exactly what is true.
- To do all this we require techniques for probabilistic reasoning

Term Used

- **Probabilities :**
  Usually, are descriptions of the likelihood of some event occurring (ranging from 0 to 1).
- **Event :**
  One or more outcomes of a probability experiment .
- **Probability Experiment :**
  Process which leads to well-defined results call outcomes.
- **Sample Space :**
  Set of all possible outcomes of a probability experiment.
- **Independent Events :**
  Two events, E_1 and E_2, are independent if the fact that E_1 occurs does not affect the probability of E_2 occurring.
- **Mutually Exclusive Events :**
  Events E_1, E_2, ..., E_n are said to be mutually exclusive if the occurrence of any one of them automatically implies the non-occurrence of the remaining $n - 1$ events.
- **Disjoint Events :**
  Another name for mutually exclusive events.

- **Classical Probability :**
  Also called a priori theory of probability.
  The probability of event A = no of possible outcomes f divided by the total no of possible outcomes n ; ie., P(A) = f / n.
  Assumption: All possible outcomes are equal likely.
- **Empirical Probability :**
  Determined analytically, using knowledge about the nature of the experiment rather than through actual experimentation.
- **Conditional Probability :**
  The probability of some event A, given the occurrence of some other event B. Conditional probability is written P(A|B), and read as "the probability of A, given B ".
- **Joint probability :**
  The probability of two events in conjunction. It is the probability of both events together. The joint probability of A and B is written $P(A \cap B)$; also written as P(A, B).
- **Marginal Probability :**
  The probability of one event, regardless of the other event. The marginal probability of A is written P(A), and the marginal probability of B is written P(B).

7.12. Probability and Bayes Theorem

In probability theory, Bayes' theorem relates the conditional and marginal probabilities of two random events.

Probability : The Probabilities are numeric values between 0 and 1 (both inclusive) that represent ideal uncertainties (not beliefs).

- **Probability of event A is P(A)**

$$P(A) = \frac{\text{instances of the event A}}{\text{total instances}}$$

P(A) = 0	indicates total uncertainty in A,
P(A) = 1	indicates total certainty and
0< P(A) < 1	values in between tells degree of uncertainty

Probability Rules :

- All probabilities are between 0 and 1 inclusive 0 <= P(E) <= 1.
- The sum of all the probabilities in the sample space is 1.
- The probability of an event which must occur is 1.
- The probability of the sample space is 1.
- The probability of any event which is not in the sample space is zero.
- The probability of an event not occurring is P(E') = 1 - P(E)

7.12.1 Conditional Probability

- **Conditional probability P(A|B)**

A conditional probability is the probability of an event given that another event has occurred.

Example : Roll two dices.

What is the probability that the total of two dice will be greater than 8 given that the first die is a 6 ?

First List of the **joint possibilities** for the two dices are:

(1, 1)	(1, 2)	(1, 3)	(1, 4)	(1, 5)	(1, 6)
(2, 1)	(2, 2)	(2, 3)	(2, 4)	(2, 5)	(2, 6)
(3, 1)	(3, 2)	(3, 3)	(3, 4)	(3, 5)	(3, 6)
(4, 1)	(4, 2)	(4, 3)	(4, 4)	(4, 5)	(4, 6)
(5, 1)	(5, 2)	(5, 3)	(5, 4)	(5, 5)	(5, 6)
(6, 1)	(6, 2)	**(6, 3)**	**(6, 4)**	**(6, 5)**	**(6, 6)**

There are 6 outcomes for which the first die is a 6, and of these, there are 4 outcomes that total more than 8 are (6,3; 6,4; 6,5; 6,6).

The probability of a total > 8 given that first die is 6 is therefore 4/6 = 2/3 .

This probability is written as: $P(\underbrace{\text{total}>8}_{\text{event}} \mid \underbrace{\text{1st die} = 6}_{\text{condition}}) = 2/3$

Read as "The probability that the total is > 8 given that die one is 6 is 2/3."

7.12.2. Probability of A and B

- **Probability of A and B is P(A and B)**

The probability that events A and B both occur.

Note : Two events are independent if the occurrence of one is unrelated to the probability of the occurrence of the other.

‡ **If A and B are independent**

then probability that events A and B both occur is:

$$P(A \text{ and } B) = P(A) \times P(B)$$

ie product of probability of A and probability of B.

‡ **If A and B are not independent**

then probability that events A and B both occur is:

$$P(A \text{ and } B) = P(A) \times P(B|A) \quad \text{where}$$

$P(B|A)$ is **conditional probability** of B given A

7.12.3. Probability of A or and B

- **Probability of A or B is P(A or B)**

The probability of either event A or event B occur.

Two events are mutually exclusive if they cannot occur at same time.

‡ **If A and B are mutually exclusive**

then probability that events A or B occur is:

$$P(A \text{ or } B) = p(A) + p(B)$$

ie sum of probability of A and probability of B

‡ **If A and B are not mutually exclusive**

then probability that events A and B both occur is:

$$P(A \text{ or } B) = P(A) \times P(B|A) - P(A \text{ and } B) \quad \text{where}$$

$P(A \text{ and } B)$ is probability that events A and B both occur while events A and B are independent and $P(B|A)$ is conditional probability of B given A.

7.13. Summary of Symbols and Notations

A U B	(A union B)	'Either A or B occurs or both occur'
A ∩ B	(A intersection B)	'Both A and B occur'
A ⊆ B	(A is a subset of B)	'If A occurs, so does B'
A'	$\bar{A}$	'Event A does not occur'
Φ	(the empty set)	An impossible event
S	(the sample space)	An event that is certain to occur
A ∩ B = Φ		Mutually exclusive Events
P(A)		Probability that event A occurs
P(B)		Probability that event B occurs
P(A U B)		Probability that event A or event B occurs
P(A ∩ B)		Probability that event A and event B occur
P(A ∩ B) = P(A) . P(B)		Independent events
P(A ∩ B) = 0		Mutually exclusive Events
P(A U B) = P(A) + P(B) – P(AB)		Addition rule;
P(A U B) = P(A) + P(B) – P(A) . P(B)		Addition rule; independent events
P(A U B) = P(A) + P(B) – P(A ∩ B)		
P(A U B) = P(A) + P(B) – P(B|A).P(A)		
P(A U B) = P(A) + P(B)		Addition rule; mutually exclusive Events
A|B	(A given B)	"Event A will occur given that event B has occurred"
P(A|B)		Conditional probability that event A will occur given that event B has occurred already

7.14. Bayes Theorem

Bayesian view of probability is related to degree of belief.

It is a measure of the plausibility of an event given incomplete knowledge.

Bayes' theorem is also known as Bayes' rule or Bayes' law, or called Bayesian reasoning.

The probability of an event **A** conditional on another event **B** ie **P(A|B)** is generally different from probability of **B** conditional on **A** ie **P(B|A)**.

- There is a definite relationship between the two, P(A|B) and P(B|A), and Bayes' theorem is the statement of that relationship.
- Bayes theorem is a way to calculate P(A|B) from a knowledge of P(B|A).
- Bayes' Theorem is a result that allows new information to be used to update the conditional probability of an event.

Let S be a sample space.

Let A1, A2, ... , An be a set of mutually exclusive events from S.

Let B be any event from the same S, such that P(B) > 0.

Then Bayes' Theorem describes following two probabilities :

$$P(A_k|B) = \frac{P(A_k \cap B)}{P(A_1 \cap B) + P(A_2 \cap B) + - - - - + P(A_n \cap B)} \quad \text{and}$$

by invoking the fact $P(A_k \cap B) = P(A_k).P(B|A_k)$ the probability

$$P(A_k|B) = \frac{P(A_k).P(B|A_k)}{P(A_1).P(B|A_1) + P(A_2).P(B|A_2) + - - - - + P(A_n).P(B|A_n)}$$

7.14.1. Bayes Theorem application

Applying Bayes' Theorem :

Bayes' theorem is applied while following conditions exist.

- the sample space S is partitioned into a set of mutually exclusive events {A1, A2, ... , An }.
- within S, there exists an event B, for which P(B) > 0.
- the goal is to compute a conditional probability of the form : P(Ak|B).
- you know at least one of the two sets of probabilities described below
  - $P(A_k \cap B)$ for each Ak
  - P(Ak) and P(B|Ak) for each Ak

Example

Example 1: Applying Bayes' Theorem

Problem : Marie's marriage is tomorrow.

- in recent years, each year it has rained only 5 days.
- the weatherman has predicted rain for tomorrow.
- when it actually rains, the weatherman correctly forecasts rain 90% of the time.
- when it doesn't rain, the weatherman incorrectly forecasts rain 10% of the time.

The question : What is the probability that it will rain on the day of Marie's wedding?

Solution : The sample space is defined by two mutually exclusive events – "it rains" or "it does not rain". Additionally, a third event occurs when the "weatherman predicts rain".

The events and probabilities are stated below.

- Event A1 : rains on Marie's wedding.
- Event A2 : does not rain on Marie's wedding
- Event B : weatherman predicts rain.
- P(A1) = 5/365 =0.0136985 [Rains 5 days in a year.]
- P(A2) = 360/365 = 0.9863014 [Does not rain 360 days in a year.]
- P(B|A1) = 0.9 [When it rains, the weatherman predicts rain 90% time.]
- P(B|A2) = 0.1 [When it does not rain, weatherman predicts rain 10% time.]

We want to know **P(A1|B)**, the probability that it will rain on the day of Marie's wedding, given a forecast for rain by the weatherman.

The answer can be determined from Bayes' theorem, shown below.

$$P(A1|B) = \frac{P(A1).P(B|A1)}{P(A1).P(B|A1)+P(A2).P(B|A2)} = \frac{(0.014)(0.9)}{[(0.014)(0.9)+(0.986)(0.1)]}$$

$$= 0.111$$

So, despite the weatherman's prediction, there is a good chance that Marie will not get rain on at her wedding.

Thus Bayes theorem is used to calculate conditional probabilities.

7.15. Certainty in Rule-based systems

The certainty-factor model was one of the most popular model for the representation and manipulation of uncertain knowledge in the early (1980s) Rule-based expert systems.

The model was criticized by researchers in artificial intelligence and statistics being ad-hoc-in nature. Researchers and developers have stopped using the model.

Its place has been taken by more expressive formalisms of Bayesian belief networks for the representation and manipulation of **uncertain knowledge**.

The manipulation of uncertain knowledge in the Rule-based expert systems is illustrated in the next three slide before moving to Bayesian Networks.

7.15.1. Certainty Factors in Rule based systems

Rule Based Systems

Rule based systems have been discussed in previous lectures.
Here it is recalled to explain uncertainty.

- A rule is an expression of the form **"if A then B"**
  where **A** is an assertion and **B** can be either an action or another assertion.
  Example : Trouble shooting of water pumps
  1. If pump failure then the pressure is low
  2. If pump failure then check oil level
  3. If power failure then pump failure
- Rule based system consists of a library of such rules.
- Rules reflect essential relationships within the domain.
- Rules reflect ways to reason about the domain.
- Rules draw conclusions and points to actions, when specific information about the domain comes in. This is called inference.

- The inference is a kind of chain reaction like :

  If there is a power failure then (see rules 1, 2, 3 mentioned above)

  Rule 3 states that there is a pump failure, and

  Rule 1 tells that the pressure is low, and

  Rule 2 gives a (useless) recommendation to check the oil level.

- It is very difficult to control such a mixture of inference back and forth in the same session and resolve such uncertainties.

  How to deal such uncertainties ?

7.15.2. Certainty Handling in Rule based systems

How to deal uncertainties in rule based system?

A problem with rule-based systems is that often the connections reflected by the rules are not absolutely certain (i.e. deterministic), and the gathered information is often subject to uncertainty.

In such cases, a certainty measure is added to the premises as well as the conclusions in the rules of the system.

A rule then provides a function that describes : how much a change in the certainty of the premise will change the certainty of the conclusion.

In its simplest form, this looks like :

If A (with certainty x) then B (with certainty f(x))

This is a new rule, say rule 4, added to earlier three rules.

7.16. Certainty Schemes

Used for handling uncertainty

- There are many schemes for treating uncertainty in rule based systems.

The most common are :

- Adding certainty factors.
- Adoptions of Dempster-Shafer belief functions.
- Inclusion of fuzzy logic.

In these schemes, uncertainty is treated locally, means action is connected directly to incoming rules and uncertainty of their elements.

Example : In addition to rule 4 , in previous slide, we have the rule

If C (with certainty x) then B (with certainty g(x))

Now If the information is that **A** holds with certainty **a** and **C** holds with certainty **c**, Then what is the certainty of **B** ?

Note : Depending on the scheme, there are different algebras for such a combination of uncertainty. But all these algebras in many cases come to incorrect conclusions because combination of uncertainty is not a local phenomenon, but it is strongly dependent on the entire situation (in principle a global matter).

7.17. Bayesian Networks

Bayesian Networks and Certainty Factors

A Bayesian network (or a belief network) is a probabilistic graphical model that represents a set of variables and their probabilistic independencies. For example, a Bayesian network could represent the **probabilistic relationships** between diseases and symptoms. Given symptoms, the network can be used to compute the probabilities of the presence of various diseases.

Bayesian Networks are also called : Bayes nets, Bayesian Belief Networks (BBNs) or simply Belief Networks. Causal Probabilistic Networks (CPNs).

A Bayesian network consists of :

- a set of nodes and a set of directed edges between nodes.
- the edges reflect cause-effect relations within the domain.
- The effects are not completely deterministic (e.g. disease -> symptom).
- the strength of an effect is modeled as a probability.

If patient has virus, the test is **+ve** with probability **0.95**.

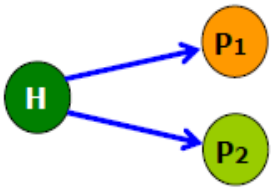
If the patient does not have the virus, the test is **+ve** with probability **0.02**.

This means given : **P(H) = 0.15 ; P(P|H) = 0.95 ; P(P|¬H) = 0.02**

Imagine, the patient is given a second test independently of the first; means the second test is done at a later date by a different person using different equipment. So, the error on the first test does not affect the probability of an error on the second test.

In other words the two tests are independent. This is depicted using the diagram below :

A simple example of a Bayesian Network.



Event **H** is the cause of the two events **P1** and **P2**.

The arrows represent the fact that **H** is driving **P1** and **P2**.

The network contained 3 nodes.

If both **P1** and **P2** are **+ve**

then find the probability that patient has the virus ?

In other words asked to find **P(H|P1 ∩ P2)** .

How to find ?

$$P(H|P1 \cap P2) = \frac{P(P1 \cap P2|H) \cdot P(H)}{P(P1 \cap P2)}$$

Here there are two quantities which we do not know.

The first is $P(P1 \cap P2|H)$ and the second is $P(P1 \cap P2)$

- Find $P(P1 \cap P2|H)$

  Since the two tests are independent, so

  $$P(P1 \cap P2|H) = (P1|H)P(P2|H)$$

- Find $P(P1 \cap P2)$

  As worked before for $P(P)$ which is the probability of a +ve result, here again break this into two separate cases:

  - patient has virus and both tests are +ve
  - patient not having virus and both tests are +ve

- As before use the second axiom of probability

  $$P(P1 \cap P2) = P(P1 \cap P2 |H)\, P(H) + P(P1 \cap P2 |\neg H)\, P(\neg H)$$

- Because the two tests are independent given **H** we can write :

  $$P(P1 \cap P2) = P(P1|H)\, P(P2|H)\, P(H) + P(P1|\neg H)\, P(P2|\neg H)\, P(\neg H)$$
  $$= 0.95 \times 0.95 \times 0.15 + 0.02 \times 0.02 \times 0.85$$
  $$= 0.135715$$

- Substitute this into Bayes Theorem above and obtain

  $$P(H|P1 \cap P2) = \frac{P(P1 \cap P2|H) \cdot P(H)}{P(P1 \cap P2)}$$
  $$= (0.95 \times 0.95 \times 0.15) / 0.135715 = 0.99749$$

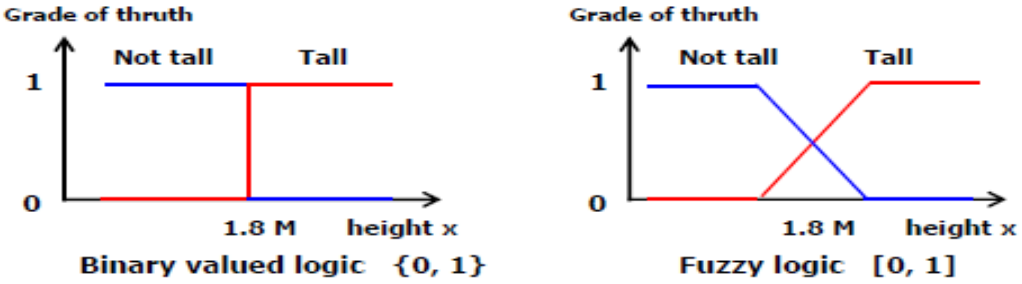
- Note : The results while two independent HIV tests performed
  - Previously we calculated the probability, that the patient had HIV given **one +ve test**, as 0.8934.
  - Later second HIV test was performed. After **two +ve tests**, we see that the probability has gone up to 0.99749.
  - So after two +ve tests it is more certain that the patient does have the HIV virus.

Fuzzy Logic

Description of Fuzzy Logic

With fuzzy logic an element could partially belong to a set represented by the **set membership**. Example, a person of height **1.79 m** would belong to both tall and not tall sets with a particular degree of membership.

Difference between binary logic and fuzzy logic



A fuzzy logic system is one that has at least one system component that uses fuzzy logic for its internal knowledge representation.

Fuzzy system communicate information using fuzzy sets.

Fuzzy logic is used purely for internal knowledge representation and externally it can be considered as any other system component.

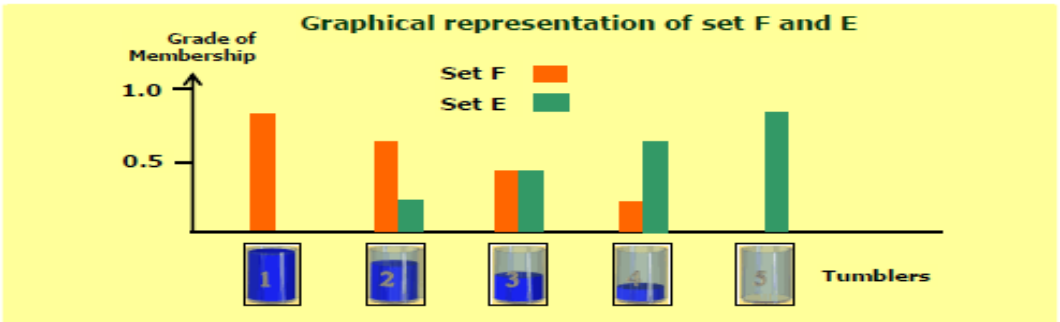
Membership

Example : Five tumblers

- Consider two sets: **F** and **E**.
- **F** is set of all tumblers belong to the class full, and
- **E** is set of all tumblers belong to the class empty.

Definition of the set F and E

Tumblers	1	2	3	4	5
Grade of membership to set *F*	100%	75%	50%	25%	0%
Grade of membership to set *E*	0%	25%	50%	75%	100%



The sets **F** and **E** have some elements, having partial membership.
Such kind of non-crisp sets are called **fuzzy sets.**
The set "all tumblers" here is the basis of the fuzzy sets **F** and **E**, is called the **base set.**

CHAPTER 8

Learning Systems: Machine Learning

8.1. Introduction

To solve problems computers require intelligence. Learning is central to intelligence. As intelligence requires knowledge, it is necessary for the computers to acquire knowledge. Machine learning serves this purpose. The general aim of Machine Learning is to produce intelligent programs, often called agents, through a process of learning and evolving.

- Machine learning, a branch of artificial intelligence, is a scientific discipline concerned with the design and development of algorithms that allow computers to evolve behaviors based on empirical data, such as from sensor data or databases.
- Machine learning refers to a system capable of acquiring and integrating the knowledge automatically. The capability of the systems to learn from experience, training, analytical observation, and other means, results in a system that can continuously self-improve and thereby exhibit efficiency and effectiveness.
- Machine learning usually refers to the changes in systems that perform tasks associated with artificial intelligence (AI). Such tasks involve recognition, diagnosis, planning, robot control, prediction, etc. The changes" might be either enhancements to already performing systems or ab initio synthesis of new systems.

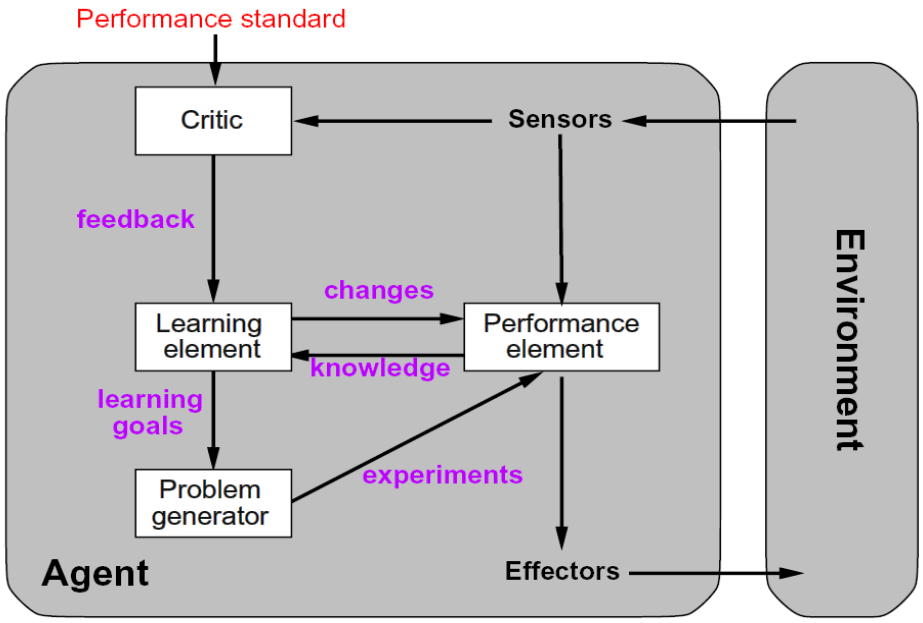
Machine learning focuses on the prediction, based on *known* properties learned from the training data. A machine learning system usually starts with some knowledge and a corresponding knowledge organization so that it can interpret, analyze, and test the knowledge acquired. Machine learning Modifies decision mechanism to improve performance.

8.2. Why Machine Learning

One might ask Why should machines have to learn? Why not design machines to perform as desired in the first place?" There are several reasons why machine learning is important.

1. Some tasks cannot be defined well except by example; that is, we might be able to specify input/output pairs but not a concise relationship between inputs and desired outputs. We would like machines to be able to adjust their internal structure to produce correct outputs for a large number of sample inputs and thus suitably constrain their input/output function to approximate the relationship implicit in the examples.
2. It is possible that hidden among large piles of data are important relationships and correlations. Machine learning methods can often be used to extract these relationships (data mining).
3. Human designers often produce machines that do not work as well as desired in the environments in which they are used. In fact, certain characteristics of the working environment might not be completely known at design time. Machine learning methods can be used for on-the-job improvement of existing machine designs.
4. The amount of knowledge available about certain tasks might be too large for explicit encoding by humans. Machines that learn this knowledge gradually might be able to capture more of it than humans would want to write down.
5. Environments change over time. Machines that can adapt to a changing environment would reduce the need for constant redesign.
6. New knowledge about tasks is constantly being discovered by humans. Vocabulary changes. There is a constant stream of new events in the world. Continuing redesign of AI systems to conform to new knowledge is impractical, but machine learning methods might be able to track much of it.

8.3. Components of a Learning System



The figure shown above is a typical learning system model. It consists of the following components.

1. Learning element
2. Knowledge base
3. Performance element
4. Feedback element
5. Standard system.

1. Learning element: It receives and processes the input obtained from a person (i.e. a teacher), from reference material like magazines, journals, etc, or from the environment at large.
2. Knowledge base: This is somewhat similar to the database. Initially it may contain some basic knowledge. Thereafter it receives more knowledge which may be new and so be added as it is or it may replace the existing knowledge.
3. Performance element: It uses the updated knowledge base to perform some tasks or solves some problems and produces the corresponding output.
4. Feedback element: It is receiving the two inputs, one from learning element and one from standard (or idealized) system. This is to identify the differences between the two inputs. The feedback is used to determine what should be done in order to produce the correct output.
5. Standard system: It is a trained person or a computer program that is able to produce the correct output. In order to check whether the machine learning system has learned well, the same input is given to the standard system. The outputs of standard system and that of performance element are given as inputs to the feedback element for the comparison. Standard system is also called idealized system.

The sequence of operations described above may be repeated until the system gets the desired perfection. There are several factors affecting the performance. They are,

- Types of training provided
- The form and extent of any initial background knowledge
- The type of feedback provided
- The learning algorithms used.

Training is the process of making the system able to learn. It may consist of randomly selected examples that include a variety of facts and details including irrelevant data. The learning techniques can be characterized as a search through a space of possible hypotheses or solutions. Background knowledge can be used to make learning more efficient by reducing the search space. The feedback may be a simple yes or no type of evaluation or it may contain useful information describing why a particular action was good or bad. If the feedback is always reliable and carries useful information, the learning process will be faster and the resultant knowledge will be correct.

The success of machine learning system also depends on the algorithms. These algorithms control the search to find and build the knowledge structures. The algorithms should extract useful information from training examples. There are several machine learning techniques available.

8.4. Disciplines that have contributed to Machine Learning;

- Statistics: A long-standing problem in statistics is how best to use samples drawn from unknown probability distributions to help decide from which distribution some new sample is drawn. A related problem is how to estimate the value of an unknown function at a new point given the values of this function at a set of sample points. Statistical methods for dealing with these problems can be considered instances of machine learning because the decision and estimation rules depend on a corpus of samples drawn from the problem environment.
- Brain Models: Non-linear elements with weighted input shave been suggested as simple models of biological neurons. Brain modellers are interested in how closely these networks approximate the learning phenomena of living brains. We shall see that several important machine learning techniques are based on networks of nonlinear elements often called neural networks. Work inspired by this school is sometimes called connectionism, brain-style computation, or sub-symbolic processing.
- Adaptive Control Theory: Control theorists study the problem of controlling a process having unknown parameters which must be estimated during operation. Often, the parameters change during operation, and the control process must track these changes. Some aspects of controlling a robot based on sensory inputs represent instances of this sort of problem.
- Psychological Models: Psychologists have studied the performance of humans in various learning tasks. An early example is the EPAM net-work for storing and retrieving one member of a pair of words when given another. Related work led to a number of early decision tree and semantic network methods.
- Some of the work in reinforcement learning can be traced to efforts to model how reward stimuli influence the learning of goal-seeking behavior in animals. Reinforcement learning is an important theme in machine learning research.
- Artificial Intelligence: From the beginning, AI research has been concerned with machine learning. AI researchers have also explored the role of analogies in learning and how future actions and decisions can be based on previous exemplary cases. Recent work has been directed at discovering rules for expert systems using decision-tree methods and inductive logic programming.
- Evolutionary Models: In nature, not only do individual animals learn to perform better, but species evolve to be better fit in their individual niches. Since the distinction between evolving and learning can be blurred in computer systems, techniques that model certain aspects of biological evolution have been proposed as learning methods to improve the performance of computer programs. Genetic algorithms and genetic programming are the most prominent computational techniques for evolution.

8.5. Applications of Machine Learning

Machine Learning is most useful when the structure of the task is not well understood but can be characterized by a dataset with strong statistical regularity. Machine Learning is also useful in adaptive or dynamic situations when the task (or its parameters) are constantly changing.

- Automatic speech recognition & speaker verification
- Printed and handwritten text parsing
- Face location and identification
- Tracking/separating objects in video
- Search and recommendation (e.g. google, amazon)
- Financial prediction, fraud detection, pricing (e.g. credit cards)
- Medical diagnosis/image analysis (e.g. pneumonia, pap smears)
- Game playing (e.g. backgammon)
- Scientific analysis/data visualization (e.g. galaxy classification)

8.6. Major Paradigms of Machine Learning

- Rote Learning: Learning by memorization. One-to-one mapping from inputs to stored representation. Association-based storage and retrieval.
- Induction: Learning from example, A form of supervised learning, use specific examples to reach general conclusions
- Clustering: Discovering similar groups, unsupervised, Inductive learning in which natural classes are found for data instances as well as classifying them.
- Analogy: determine correspondence between two different representations that come from inductive learning in which a system transfers knowledge from one database to another of a different domain Determine correspondence between two different representations
- Discovery: Learning without a teacher, learning is both inductive and deductive.
  It is deductive if it proves theorem and discovers concepts about these theorems.
  It is inductive when it raises conjecture (guesses).
  Unsupervised, specific goal not given
- Genetic Algorithms: Inspired by natural evolution. In a natural world, organisms that are not environmentally suited die off, while those that are suited for it prosper.
- Reinforcement: learning from feedback (positive or negative reward) given at end of a sequence of steps. Requires assigning reward to steps by solving the credit assignment problem—which steps should receive credit or blame for a final result?

Rote Learning

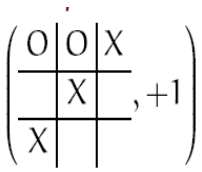
- Rote learning, also known as learning by repetition, is a method of learning by memorizing information. This memorization is usually achieved through the repetition of activities such as reading or recitation, and the use of flashcards and other learning aids.

Inductive Learning

Inductive reasoning, also known as induction or inductive logic, is a kind of reasoning that constructs or evaluates propositions that are abstractions of observations of individual instances of members of the same class. It is commonly construed as a form of reasoning that makes generalizations based on individual instances. In this sense it is often contrasted with deductive reasoning.

Simplest form: learn a function from examples

- f is the target function
- An example is a pair (x; f(x)), e.g.



- Highly simplified model of real learning:
- Ignores prior knowledge
- Assumes a deterministic, observable environment
- Assumes examples are given
- Assumes that the agent wants to learn f — why?

Different types of inductive learning:

- *Supervised Learning*: The program attempts to infer an association between attributes and their inferred class.
  - Concept Learning
  - Classification
- *Unsupervised Learning*: The program attempts to infer an association between attributes but no class is assigned.:
  - Reinforced learning.
  - Clustering
  - Discovery

- *Online vs. Batch Learning*

8.7. Types of Machine Learning

Machine learning algorithms can be organized into a taxonomy based on the desired outcome of the algorithm.

1. Supervised learning: generates a function that maps inputs to desired outputs (also called labels, because they are often provided by human experts labeling the training examples). For example, in a classification problem, the learner approximates a function mapping a vector into classes by looking at input-output examples of the function.
2. Unsupervised learning models a set of inputs, like clustering. See also data mining and knowledge discovery.
3. Semi-supervised learning combines both labeled and unlabeled examples to generate an appropriate function or classifier.
4. Reinforcement learning learns how to act given an observation of the world. Every action has some impact in the environment, and the environment provides feedback in the form of rewards that guides the learning algorithm.
5. Transduction Learning tries to predict new outputs based on training inputs, training outputs, and test inputs.

8.7.1. Supervised Learning

Supervised learning (often also called directed data mining) is the machine learning task of inferring a function from supervised (labeled) training data. The training data consist of a set of training examples. In supervised learning, each example is a pair consisting of an input object (typically a vector) and a desired output value (also called the supervisory signal). A supervised learning algorithm analyzes the training data and produces an inferred function, which is called a classifier (if the output is discrete, see classification) or a regression function (if the output is continuous, see regression). The inferred function

should predict the correct output value for any valid input object. This requires the learning algorithm to generalize from the training data to unseen situations in a "reasonable" way.

In supervised learning the variables under investigation can be split into two groups: explanatory variables and one (or more) dependent variables. The target of the analysis is to specify a relationship between the explanatory variables and the dependent variable as it is done in regression analysis.

Types of Supervised Learning

Depending on the type of the outputs, classification learning, preference learning and function learning are distinguished.

- Classification Learning: If the output space has no structure except whether two elements of the output space are equal or not, this is called the problem of *classification learning*. Each element of the output space is called a *class*. This problem emerges in virtually any pattern recognition task. Of particular importance is the problem of binary classification, i.e., the output space contains only two elements, one of which is understood as the positive class and the other as the negative class. Although conceptually very simple, the binary setting can be extended to multi-class classification by considering a series of binary classifications.
- Preference Learning: If the output space is an order space -that is, we can compare whether two elements are equal or, if not, which one is to be preferred—then the problem of supervised learning is also called the problem of *preference learning*. The elements of the output space are called *ranks*.
- Function Learning: If the output space is a metric space such as the real numbers then the learning task is known as the problem of *function learning*. One of the greatest advantages of function learning is that by the metric on the output space it is possible to use gradient descent techniques whenever the functions value *f(x)* is a differentiable function of the object *x* itself. This idea underlies the *back-propagation algorithm*, which guarantees the finding of a local optimum. An interesting relationship exists between function learning and classification learning when a probabilistic perspective is taken. Considering a binary classification problem, it suffices to consider only the probability that a given object belongs to the positive class. Thus, whenever we are able to learn the function from objects to [0,1] (representing the probability that the object is from the positive class), we have learned implicitly a classification function by thresholding the real-valued output at 1/2. Such an approach is known as *logistic regression* in the field of statistics, and it underlies the support vector machine classification learning algorithm.

Supervised Learning Problem Solving

In order to solve a given problem of supervised learning, one has to perform the following steps:

1. Determine the type of training examples. Before doing anything else, the user should decide what kind of data is to be used as a training set. In the case of handwriting analysis, for example, this might be a single handwritten character, an entire handwritten word, or an entire line of handwriting.
2. Gather a training set. The training set needs to be representative of the real-world use of the function. Thus, a set of input objects is gathered and corresponding outputs are also gathered, either from human experts or from measurements.
3. Determine the input feature representation of the learned function. The accuracy of the learned function depends strongly on how the input object is represented. Typically, the input object is transformed into a feature vector, which contains a number of features that are descriptive of the object. The number of features should not be too large, because of the curse of dimensionality; but should contain enough information to accurately predict the output.
4. Determine the structure of the learned function and corresponding learning algorithm. For example, the engineer may choose to use support vector machines or decision trees.

5. Complete the design. Run the learning algorithm on the gathered training set. Some supervised learning algorithms require the user to determine certain control parameters. These parameters may be adjusted by optimizing performance on a subset (called a *validation* set) of the training set, or via cross-validation.
6. Evaluate the accuracy of the learned function. After parameter adjustment and learning, the performance of the resulting function should be measured on a test set that is separate from the training set.

8.7.2. Unsupervised Learning

Unsupervised learning refers to the problem of trying to find hidden structure in unlabeled data. Unsupervised learning is based on the similarities and differences among the input patterns. It does not result directly in differences in overt behavior because its "outputs" are really internal representations.
Approaches to unsupervised learning include:

- clustering (e.g., k-means, mixture models, hierarchical clustering),
- blind signal separation using feature extraction techniques for dimensionality reduction (e.g., Principal component analysis, Independent component analysis, Non-negative matrix factorization, Singular value decomposition).

8.7.3. Reinforcement Learning

Reinforcement learning is an area of machine learning in computer science, concerned with how an agent ought to take actions in an environment so as to maximize some notion of cumulative reward. In machine learning, the environment is typically formulated as a Markov decision process (MDP), and many reinforcement learning algorithms for this context are highly related to dynamic programming techniques.

Reinforcement learning differs from standard supervised learning in that correct input/output pairs are never presented, nor sub-optimal actions explicitly corrected. Further, there is a focus on on-line performance, which involves finding a balance between exploration (of uncharted territory) and exploitation (of current knowledge). The exploration vs. exploitation trade-off in reinforcement learning has been most thoroughly studied through the multi-armed bandit problem and in finite MDPs.

The basic reinforcement learning model consists of:

1. a set of environment states *S*;
2. a set of actions *A*;
3. rules of transitioning between states;
4. rules that determine the *scalar immediate reward* of a transition; and
5. rules that describe what the agent observes.

The rules are often stochastic. The observation typically involves the scalar immediate reward associated to the last transition. In many works, the agent is also assumed to observe the current environmental state, in which case we talk about full observability, whereas in the opposing case we talk about partial observability. Sometimes the set of actions available to the agent is restricted (e.g., you cannot spend more money than what you possess).

A reinforcement learning agent interacts with its environment in discrete time steps. At each time t, the agent receives an observation o_t, which typically includes the reward r_t. It then chooses an action a_t from the set of actions available, which is subsequently sent to the environment. The environment moves to a new state s_{t+1} and the reward r_{t+1} associated with the *transition* (s_t,a_t,s_{t+1}) is determined. The goal of a

reinforcement learning agent is to collect as much reward as possible. The agent can choose any action as a function of the history and it can even randomize its action selection.

Uses for Reinforcement Learning

A variety of different problems can be solved using Reinforcement Learning. Because RL agents can learn without expert supervision, the type of problems that are best suited to RL are complex problems where there appears to be no obvious or easily programmable solution. Two of the main ones are:

1. Game playing- determining the best move to make in a game often depends on a number of different factors, hence the number of possible states that can exist in a particular game is usually very large. To cover this many states using a standard rule based approach would mean specifying an also large number of hard coded rules. RL cuts out the need to manually specify rules, agents learn simply by playing the game. For two player games such as backgammon, agents can be trained by playing against other human players or even other RL agents.
2. Control problems- such as elevator scheduling. Again, it is not obvious what strategies would provide the best, most timely elevator service. For control problems such as this, RL agents can be left to learn in a simulated environment and eventually they will come up with good controlling policies. Some advantages of using RL for control problems is that an agent can be retrained easily to adapt to environment changes, and trained continuously while the system is online, improving performance all the time.

CHAPTER NINE

Expert Systems

Chapter Objectives

By the end of this chapter the learner should be able to;

- Describe and Explain the components, characteristics and features of an Expert System
- Explain knowledge acquisition and techniques
- Explain application of Expert Systems

9.1. Introduction

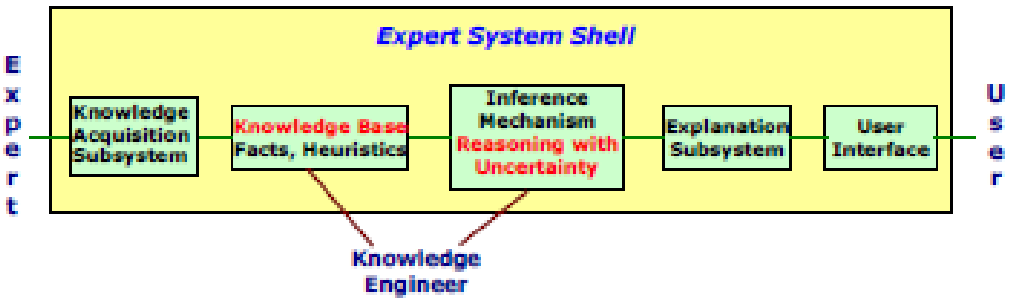
AI programs that achieve expert-level competence in solving problems in task areas by bringing to bear a body of knowledge about specific tasks are called *knowledge-based* or *expert systems*.

Definition from literature

- An expert system is an interactive computer-based decision tool that uses both facts and heuristics to solve difficult decision making problems based on knowledge acquired from an expert.
- An expert system is a model and associated procedure that exhibit within a specified domain a degree of expertise in problem solving that is comparable to that of a human expert.
- An expert system compared with traditional computer Inference Engine + Knowledge = Expert system. Algorithm + Data structures = Program in traditional computer
- Expert systems are computer applications which embody some non-algorithmic expertise for solving certain types of problems

9.2. Expert System Shell

Many expert systems are built with products called expert system shells. The expert shell is a software development environment that contains the basic components of an expert system; user interface, a format for declarative knowledge in the knowledge base, and the inference engine.



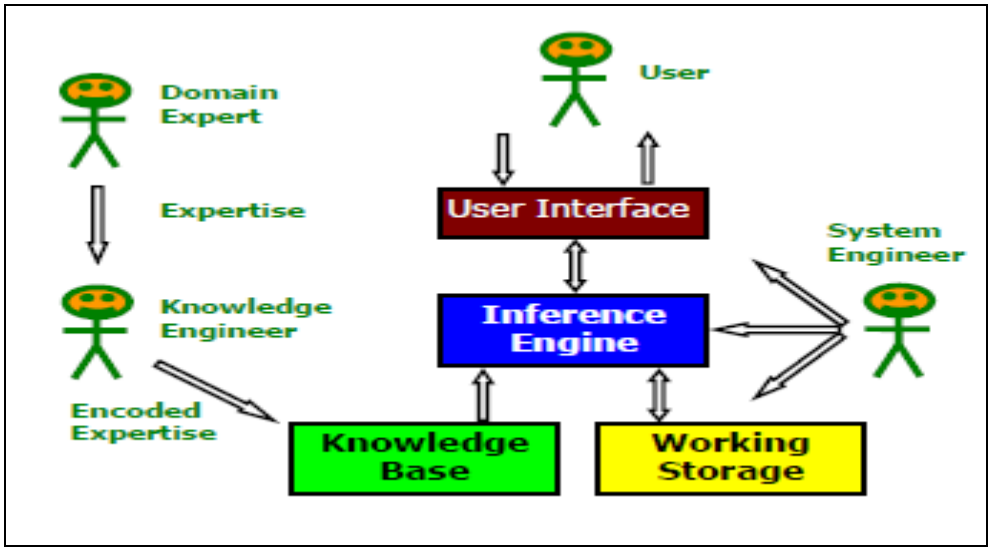
Shell Components

- Knowledge base sub-system: Expert system tool provides one or more knowledge representation schemes for expressing knowledge about the application domain
- Reasoning Engine sub-system : Inference mechanism for manipulating the symbolic information and knowledge in the knowledge base form a line of reasoning in the solving a problem.
- Knowledge Acquisition subsystems: a subsystem to help experts in building knowledge base.

- Explanation subsystem: A subsystem that explains the system's action. The explanation can range from how the final or intermediate solution was arrived at justifying the need for additional data.
- User Interface: Provides a means of communication with the user.

9.3. Expert System Components

Expert systems have a number of major system components and interface with individuals who interact with the system in various roles.



Components and Interfaces

- Knowledge Base: A declarative representation of the expertise often in the IF THEN rules.
- Working Storage: The data which is specified to a problem being solved
- Interface Engine: The code at the core of the system which derives recommendations from the knowledge base and problem specific data in working storage
- User Interface: The code that controls the dialog between the user and the system.

9.3.1 Knowledge Base

Expert systems contain a formal representation of the information provided by the expert domain
This information may be in the form of problem-solving rules, procedures, or data intrinsic to the domain. The *knowledge base* of expert systems contains both factual and heuristic knowledge.

- *Factual knowledge* is that knowledge of the task domain that is widely shared, typically found in textbooks or journals, and commonly agreed upon by those knowledgeable in the particular field.
- *Heuristic knowledge* is the less rigorous, more experiential, more judgmental knowledge of performance. In contrast to factual knowledge, heuristic knowledge is rarely discussed, and is largely individualistic. It is the knowledge of good practice, good judgment, and plausible reasoning in the field. It is the knowledge that underlies the "art of good guessing."

Knowledge Base & Knowledge Management: To incorporate these information into the system, it is necessary to make use of one or more knowledge representation methods. *Knowledge representation* formalizes and organizes the knowledge. One widely used representation is the *production rule*, or simply *rule*. A rule consists of an IF part and a THEN part (also called a *condition* and an *action*). The IF part lists a set of conditions in some logical combination. The piece of knowledge represented by the production rule is relevant to the line of reasoning being developed if the IF part of the rule is satisfied; consequently, the THEN part can be concluded, or its problem-solving

action taken. Expert systems whose knowledge is represented in rule form are called *rule-based systems*.

Another widely used representation, called the *unit* (also known as *frame*, *schema*, or *list structure*) is based upon a more passive view of knowledge. The unit is an assemblage of associated symbolic knowledge about an entity to be represented. Typically, a unit consists of a list of properties of the entity and associated values for those properties.

Expert Systems Knowledge Management Rules

The rules includes; IF-THEN rules, semantic networks and frames.

IF-THEN Rules

Human experts tend to think along:
Condition ⇨ action or situation ⇨nclusion.

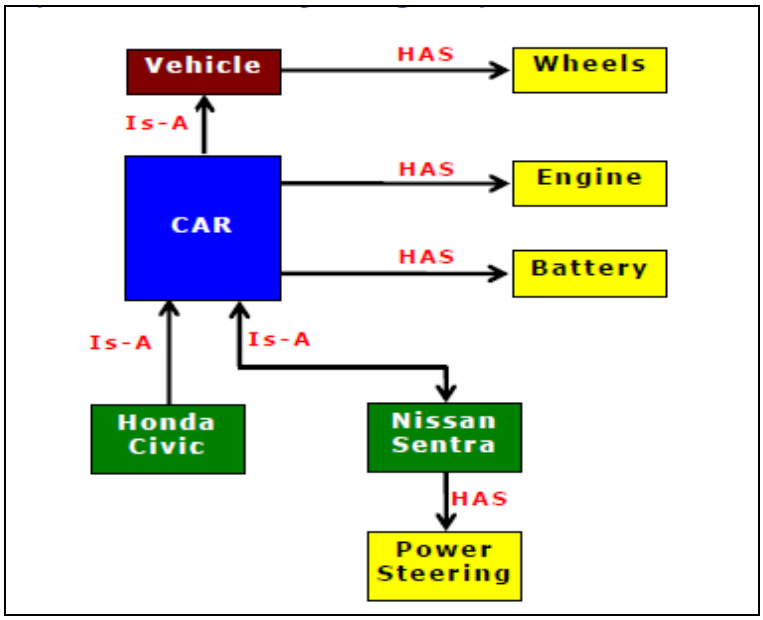
1. Rules "If-then" are predominant form of encoding knowledge in expert systems. These are of the form
   If $a_1, a_2, \ldots\ldots a_n$
   Then $b_1, b_2, \ldots\ldots b_n$ where
   each a_1 is a condition or situation, and
   each a_1 is an action or a conclusion

Semantic Networks

In this scheme, knowledge is represented in terms of objects and relationships between objects. The objects are denoted as nodes of a graph. The relationship between two objects are denoted as a link between the corresponding two nodes.

The most common form of semantic networks uses the links between nodes to represent IS-A and HAS relationship between objects. This kind of relationship establishes an inheritance hierarchy in the network, with the objects lower down in the network inheriting properties from the object higher up

Example of Semantic Networks
The example below shows a car IS-A vehicle; a vehicle HAS wheels



Semantic Network

Frames

In this technique, knowledge is decomposed into highly modular pieces called frames, which are generalized recode structures. Knowledge consists of concepts, situations, attributes of concepts, relationships between concepts, and procedures to handle relationships as well as attribute values.

- Each concept may be represented as a separate frame.
- The attributes, relationships between concepts, and the procedures are allotted to slots in the a frame.
- The contents of a slot may be of any data type- numbers, strings, functions, or procedures.
- The frames ay be linked to other frames, providing the same kind of inheritance as that provided by a semantic network.

A frame- based representation is suitable for object-oriented programming techniques

Example: Frame-based Representation of knowledge.
Two frames, their slots and the slots filled with data type are shown

Frame	Car
Inheritance Slot	Is-A
Value	Vehicle
Attribute Slot	Engine
Value	Vehicle
Value	1
Value	
Attribute Slot	Cylinders
Value	4
Value	6
Value	8
Attribute Slot	Doors
Value	2
Value	5
Value	4

Frame	Car
Inheritance Slot	Is-A
Value	Car
Attribute Slot	Make
Value	Honda
Value	
Value	
Attribute Slot	Year
Value	1989
Value	
Value	
Attribute Slot	
Value	
Value	
Value	

Roles of Individuals who interact with the system.

- Domain Expert: The individuals who currently are experts in solving the problems, here the system is intended to solve.
- Knowledge engineer: The individual who encodes the expert's knowledge in a declarative form that can be used by the expert system.
- User: The individual who will be consulting the system to get advice which would have been provided by the expert.

9.3.2. Working Memory.

Working memory refers to task-specific data for a problem. The content of the working memory, changes with each problem situation. It is the most dynamic component of an expert system provided it is kept current.

- Every problem in a domain has some unique data associated with it.
- Data may consist of the set of condition leading to the problem, its parameter and soon.
- Data specific to the problem needs to the input by the user at the time of using, means consulting the expert system. The working memory is related to user interface.

9.3.3. Inference Engine

The inference engine is a generic control mechanism for navigating through and manipulating knowledge and deduce results in an organized manner.

The generic control mechanism applies the axiomatic knowledge present in the knowledge base to the task-specific data to arrive at some conclusion.

- Inference engine the other key component of all expert systems
- Just a knowledge base alone is not of much use if there are no facilities for navigating through and manipulating the knowledge to deduce something from knowledge base.

The forward chaining, backward chaining and the tree searches are the most used techniques for drawing inference from the knowledge base.

9.4. Expert System Characteristics

Expert system operates as an interactive system that responds to questions, asks for clarifications, makes recommendations and generally aid the decision making process. Expert systems have many characteristics.

1. Operates an Interactive systems: This means that an expert system:
   - Responds to questions.
   - Asks for clarification
   - Makes recommendations
   - Aids the decision making process.
2. Tools have ability to sift (filter) knowledge
   - Storage and retrieval of knowledge.
   - Mechanism to expand and update knowledge base on a continuing basis
3. Make logical inferences based on knowledge stored.
   - Simple reasoning mechanism is used.
   - Knowledge base must have a means of exploiting the knowledge stored, else it is useless; e.g., learning all the worlds in the language without knowing how to combine those words to form a meaningful sentence.
3. Ability to Explain Reasoning
   - Remember logical chain reasoning; therefore user may ask
     - for explanation of a recommendation
     - For factors considered in a recommendation
   - Enhance user confidence in recommendation and acceptance of the expert system
4. Domain-Specific
   - A particular system caters a narrow area of specialization.
   - Quality of advice offered by the expert system is dependent on the amount of knowledge stored.
5. Capability of assign Confidence Values.
   - Can deliver quantitative information
   - Can interpret qualitatively derived values.
   - Can address imprecise and incomplete data through assignment of confidence values.
6. Application
   - Best suit for those dealing with expert heuristics for solving problems
   - Not a suitable choice for those problems that can be solved using purely numerical techniques
7. Cost-Effective alternative to Human Expert
   - They have become increasingly popular because of their specialization in a narrow field.
   - Encoding and storing the domain-specific knowledge is economic process due to small size.
   - Specialists in many areas are rare and the cost of consulting them is high; an expert system of those areas can be useful and cost effective alternative in the long run.

9.5. Expert System Features

The Features of an Expert system includes;

1. Goal Driven Reasoning or Backward Chaining: An inference technique which uses IF-THEN rules to repetitively break a goal into their smaller sub-goals which are easier to prove
   The algorithm proceeds from the desired goal, adding new assertions found

2. Coping with Uncertainty: The ability of the system to reason with rules and data which is incomplete.

Often knowledge is imperfect which causes uncertainty.
To work in the real world, Expert systems must be able to deal with uncertainty

- One simple way is to associate a numeric value with each piece of information in the system.
- The numeric value represents the certainty with which the information is known.

3. Data Driven Reasoning or Forward Chaining: An inference technique which uses IF-THEN rules to deduce a problem solution from initial data. The system keeps track of the current state of the problem solution and looks for rules which will move that state closer to a final solution. The algorithm proceeds from a given situation to a desired goal, adding new assentations found.

4. Data Representation: The way in which the problem specific data in the system is stored and accessed. Expert systems are built around knowledge base modules
    - Knowledge acquisition is transferring knowledge from human expert to computer
    - Knowledge representation is faithful representation of what the expert knows

   The success of the expert system depends on choosing knowledge encoding scheme best for the kind of knowledge the system is based on.
   The IF-Then rules, semantic networks and frames are the most commonly used representation schemes.
5. User Interface: That portion of the code which creates an easy to use system. The acceptability of an expert system depends largely on the quality of the user interface. Examples;
    - Scrolling dialog interface: It is easiest to implement and communicate with user.
    - Pop-up menus, windows, mice are more advanced interfaces and powerful tools for communicating with the user; they require graphics support
6. Explanations: The ability of the system to explain the reasoning process that is used to reach a recommendation
    - An important features of expert systems is their ability to explain themselves. Given that the system knows which rules were used during the inference process, the system provides those rules to the user as means of explaining the results.
    - By looking at the explanations, the knowledge engineer can see how the system is behaving and how the rules and data are interacting. This is a very valuable diagnostic tool during development.

9.6. Knowledge Acquisition

Knowledge acquisition includes the elicitation, collection, modelling and validation of knowledge.
Issues in Knowledge Acquisition: The important issues are:

1. Knowledge is in the head of experts
2. Experts have vast amounts of knowledge
3. Experts have a lot of implicit knowledge
    - They do not know all that they know and use.
    7. Implicit knowledge is hard (impossible) to describe
4. Experts are very busy and valuable people.
5. One expert does not know everything
6. Knowledge has a "shelf life"

9.6.1 Knowledge Acquisition techniques

The techniques for acquiring, analyzing and modelling knowledge are:

1. Protocol-generation techniques: Include many types of interviews (unstructured, semi-structured, and structured), reporting and observational techniques
2. Protocol analysis techniques: Used with transcripts of interviews or text-based information to identify basic knowledge objects within a protocol, such as goals, decisions, relationships and attributes. These act as a bridge between the use of protocol-based techniques and knowledge modelling techniques

3. Hierarchy-generation techniques: Involves creation, reviewing and modification of hierarchical knowledge. Hierarchy-generation techniques like Laddering are used to build taxonomies or other hierarchical structures like goal trees and decision trees
4. Matrix-based techniques: Involve the construction and filling in a 2-D matrix (grid, table) indicating such things as may be, e.g. between concepts and properties (attributes and values) or between problems and solutions, or between tasks and resources. The elements within the matrix can contain symbols (ticks, crosses, question marks), colours, numbers, text e.t.c
5. Sorting Techniques: Used to capture the way people compare and order concepts; it may reveal knowledge about classes, properties and priorities.
6. Limited-information and Constrained-processing tasks: These either limit the time and/or information available to the expert when performing tasks. e.g a twenty-questions technique provides an efficient way of accessing the key information in a domain in a prioritized order.
7. Diagram-based technique: Involves generation and use of concept maps, state transition networks, event diagrams and process maps. These are particularly important in capturing the "What, how, when, who, and why" of the tasks and events

9.7. Expert System Explanation

Most expert systems have explanation facilities that allow the user to ask question-
"why and how it reached some conclusion?".

The questions are answered by referring to the system goals, the rules being used, and the existing problem solving schemes. The rules typically reflect empirical or 'compiled' knowledge

Example: Dialog with an expert system designed to give advice on car problem

System	Does the car start?
User	No
System	Does the engine turn over?
User	Yes
System	Do you smell gas?
User	Yes
System	Recommendation- wait 5 minutes and try again
User	Why?
System	I used the rule: if not start, and engine-turn-over and smell-gas Then recommendation is wait 5 minutes and try again

9.8. Application of Expert Systems

- Diagnosis and Troubleshooting of Devices and Systems: Medical diagnosis was one of the first knowledge base areas to which expert system technology was applied

- Planning and scheduling: They have been used in airlines scheduling their flights, personnel, and gates, the manufacturing process planning and job scheduling.

- Configuring of Manufactured Objects from sub-assemblies: Used in modular home building and manufacturing involving complex engineering designs.

- Financial Decision Making: used in banking sectors in managing loans, Insurance campaniles to assess the risk presented by the customer and to determine a price for the insurance.

  Knowledge Publishing: The expert system deliver knowledge that is relevant to the user domain.

- Process Monitoring and Control: The expert system performs real-time analysis of data from physical devices, looking for anomalies, predicting trends, controlling optimality, and failure correction.

- Design and Manufacturing: Expert systems assist in the design of physical devices and processes, ranging from high-level conceptual design to abstract entities all the way to factory floor configuration of manufacturing processes.

CHAPTER TEN

Natural Language Processing

Chapter Objectives

By the end of this chapter the learner should be able to;

- Describe Natural Language
- Explain Syntactic Processing
- Explain Knowledge Representation using Predicate Logic and Rules
- Explain how knowledge may be represented as symbol structures that characterize bits of knowledge about objects, concepts.

10.1. Introduction

Natural language Processing (NLP) is a subfield of Artificial Intelligence and linguistic, devoted to make computers understand statements written in human language.

10.1.1. Natural Language:

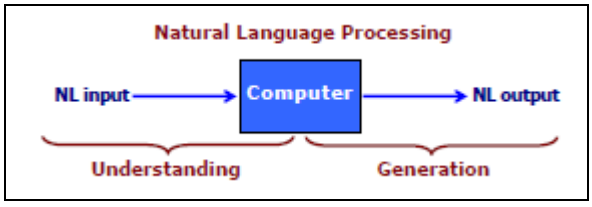
A natural language (or ordinary language) is a language that is spoken, written by human beings for general- purpose communication e.g. English, Swahili, Runyakitara e.t.c

A language is a system with a set of symbols and a set of rules (or grammar)

- The symbols are combined to convey new information
- The rules govern the manipulation of symbols

NLP encompasses anything a computer needs to understand languages (typed or spoken) and also generate the natural language

- Natural Language Understanding (NLU): The task involves understanding and reasoning while the input is a natural language.
- Natural Language Generation(NLG): is a sub-field of natural language processing some time referred to as text generation.



structure of NLP system

10.1.2. Formal Language

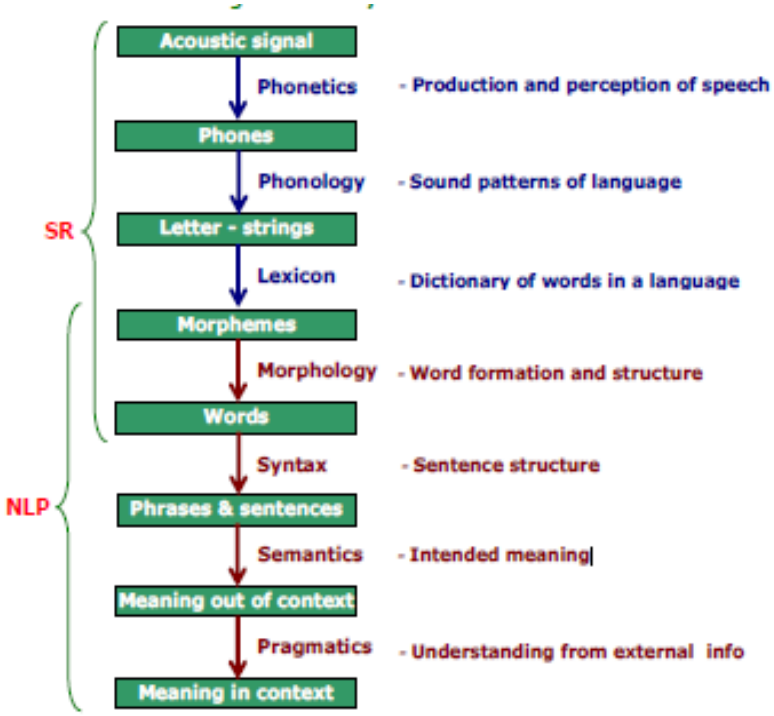
Definition of terms used in by Formal Language

- Symbol: is a character, an abstract entity that has no meaning by itself. e.g. letters, digits and special characters.
- Alphabet: is a finite set of symbols. An alphabet is often denoted by ∑(sigma) e.g; B =f0; 1g says that B is an alphabet of two symbols 0 and 1.

- String or a word is a finite sequence of symbols from an alphabet. e.g. 01110 and 11 are strings from the alphabet B above.
- Language is a set of strings from an alphabet
- Formal Language (simply language) is a set L of strings over some finite alphabet ∑
  Formal language is described by using formal grammar

10.2. Linguistics and Language Processing

Linguistics is the science of languages. It s study includes, sounds (phonology), word formation (morphology), sentence structure (syntax), meaning (semantics) and understanding (pragmatics).
The higher level in linguistics corresponds to speech recognition (SP) and the lower level



Levels of Linguistics

10.2.1 Steps in Natural language processing

1. Morphological and Lexical Analysis: The lexicon of a language is its vocabulary, that include its words and expressions. Morphology is the identification, analysis and description of structures of words. The word is the smallest unit syntax. The syntax refers to the rules and principles that govern the sentence structure of any individual language
   - Lexical analysis: Aims to divide the text into paragraphs, sentience and words. Lexical analysis is done hand in hand with morphological and syntactic analysis

2. Syntactic Analysis: The analysis of words in a sentence to know the grammatical structure of the sentence. The words are transformed into structures that show how the words relate to each other. In the process some words may be rejected if they violate the rules of the language for how words may be combined.
   Example: In English syntax analyzer would reject the sentence say " Boy the go the to store"

3. Semantic Analysis: Derive an absolute meaning from context. It determines the possible meaning of a sentence in a context. It makes a mapping between syntactic structures and objects in the task domain and structures for which no such mapping is possible is rejected

4. Discourse Integration: The meaning of an individual sentence may depend on the sentences that precede it and may influence the meaning of the sentences that follow it. Not a suitable choice for those problems that can be solved using purely numerical techniques. E.g. the word "It" in the sentence "You wanted It" depends on the prior discourse context

5. Pragmatic analysis: deriving knowledge from external; commonsense information. It means understanding the purposeful use of language in situations, particularly those aspects of language which require world knowledge. The idea is getting the correct interpretation for a given sentence in a particular situation

10.3. Linguistics Analysis Terms

1. Phones: are acoustic patterns that are significant and distinguishable in some human language. E.g. In English the L- sound at the beginning and end of the word "Loyal" are termed as light L and dark L by linguistics
2. Phonetics: Tell how acoustic signals are classified into phones.
3. Phonology: Tells how phones are grouped together to form phonemes in a particular human language.
4. Strings: An alphabet is finite set of symbols: a string is a sequence of symbols taken from an alphabet.
5. Lexicon: is a collection of information about words of a language. The information is about the lexical categories to which words belong. Example: The word "pig" usually refers to a noun (N) but also occurs as a verb (V) and an adjective (ADj).
6. Lexicon structure: is collection of lexical entries e.g. ("pig" N, V, ADJ).
7. Words: is a unit of language that carries a meaning. Example Words like bear, car , house called nouns are very different from words like run, sleep, think called verbs and are different from words like in, under, about called prepositions.
   These and other categories of words have names such as nouns, verbs, prepositions and so on.
   Words build phrases, which in turn build sentences.
8. Determiner: occur before nouns and indicate the kind of reference which the noun has. The example below shows determiners marked in bold letters the boy, a bus, our car, these children, both hospitals
9. Morphology: is the analysis of words into morphemes and conversely the synthesis of words from morphemes.
10. Morphemes: the smallest meaningful unit in the grammar of a language, A smallest linguistic unit that has a meaning. A unit of language immediately below the word level. or a smallest part of a word that can carry a discrete meaning.
    Example: the word unbreakable has 3 morphemes
    a) un- a bound morpheme
    b) break a free morpheme
    c) able a bound morpheme.
11. Also "un-" is also a prefix, "-able" is a suffix; Both are affixes
12. Syntax: is the structure of a language. It is the grammatical arrangement of words in a sentence to show its relationship to one another in a sentence. Syntax is a finite set of rules that specifies a language. The syntax rules govern proper sentence structure.
13. Semantics: meaning of word/phrases/sentence/whole text. semantics is restricted to "meaning out of context" -that is meaning as it can be determined without taking context into account
14. Pragmatics: tell how language is used, that is meaning in context e.g if someone say "the door is open" that it is necessary to know which door "the door" refers to, The need to know the intention of the

speaker. could be a pure statement of fact could be an explanation of how the cat got in or could be a request to the person addressed to close the door

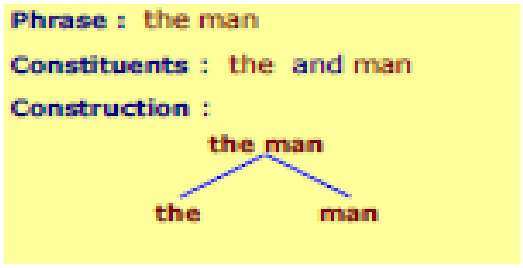
10.4. Types of Morphemes

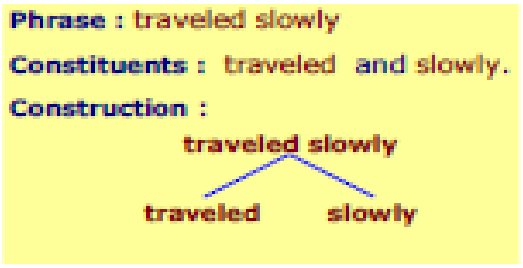
- Free Morphemes: can appear stand alone, or free e.g. town, dog or with other lexemes like town hall, dog house
- Bound Morphemes: appear only together with other morphemes to form a lexeme e.g. un in general it tend to fine prefix and suffix.
- Inflectional Morpheme: Modify a words tense, number, aspect etc. e.g. dog morpheme with a plural marker morpheme s becomes dogs
- Derivational Morphemes: can be added to a word to derive another word e.g. addition of "-ness" to "happy" gives "happiness"
- Root Morphemes: The primary lexical unit of a word. roots can be either free or bounded morphemes. Sometimes root is used to describe word minus its inflectional endings but with its lexical ending. e.g. word chatters has the inflectional root or lemma chatter but the lexical root chat
  Inflectional roots are often called stems and a root in the stricter sense may be though as of as a mono-morphemic stem.
- Null morphemes; invisible a fix also called morphemes represented as either the figure zero (0), the empty set symbol _ or its variant _.adding a null morphemes is called null affixation, null derivation or zero derivation; a null morpheme that contract singular morphemes with plural morphemes.
  e.g. cat = cat + 0 =ROOT("cat") + singular
  cats = cat + s =ROOT("cat") + plural

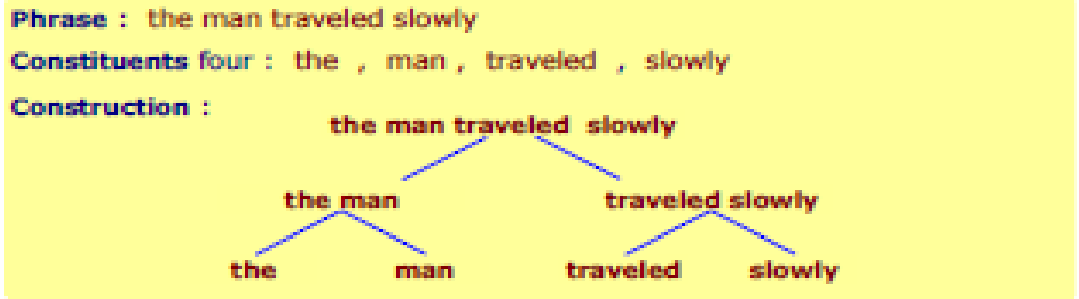
10.5. Grammatical Structure of Utterance.

Sentence is a string of words satisfying grammatical rules of a language. Sentences are classified as simple, compound and complex. A sentence is always abbreviated to "S"

Constituents: Assume that a phrase is a construction of some kind. A construction means a syntactic arrangement that consists of parts usually two called constituent. e.g . The phrase the man is a construction consists of two constituents the and man.







Constituents

Phrase

A phrase is a group of words that function as a single unit in the syntax sentence.e.g.1 "the house at the end of the street" is a phrase, acts like noun.
e.g.2 "end of the street" is a phrase, acts like adjective
Phrase formation is governed by a phase structure rule. Most phrases have a head or central word, which defines the type of phrase. Head is often the first word of the phrase
Some phrase are headless. eg,3 "the rich" is a noun phrase composed of a determiner and an adjective but noun. Phrase are classified based on type of head they take.

Classification of Phrases: Names(abbreviations)

- Sentence (S) often abbreviated to "S"
- Noun Phrase (NP): noun or pronoun as head, or optionally accompanied by a set of modifiers. The possible modifiers include determiners : articles (the, a) or adjectives (the red ball). e.g the
- black cat or a cat on the mat
- Verb Phrase (VP): verb head e.g eat cheese, jump up and down Adjective phrase (AD): adjective as a head e.g full of toys.
- Adverbial phrase (AdvP): adverb as head. e.g very carefully
- Prepositional Phrase (PP): preposition as head; e.g in love; over the rainbow
- Determiner phrase (DP): determiner as head; e.g a little dog; the little dogs.
  English, determiners are usually placed before nouns as noun modifiers that include: article(the, a), demonstrative (this, that); numerical(two, five etc), possessive(my, their etc.) and quantifiers(some, many etc).

Phrase Structure Rules

- Phrase structure rules provides a way to describe language syntax.
- Rules determine what goes into phrase and how its constituents are ordered.
- Rules are used to break a sentence down to its constituent parts namely phrase categories and lexical categories.
    - Phrase category include: noun phrase, verb phrase, prepositional phrase e.t.c
    - Lexical categories include: noun, verb, adverb, others etc
- Phrase structures are of the form A →BC. Meaning that A is separated into two sub-constituents B and C or simply A consists of B followed by C
    - S → NP VP Reads: s consists of an NP followed by a VP which means that a sentence consists of a noun phrase followed by a verb phrase
    - NP → Det N1 Reads: NP consists of an Det followed by N1 which means that a noun phrase consists of a determiner followed by a noun

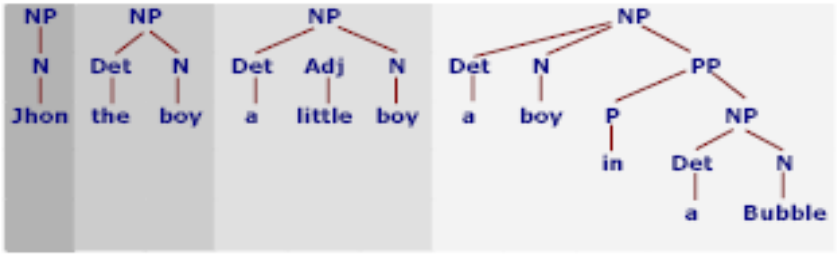
Phrase Structure Rules and Trees for Noun phrase

Noun Phrase (NP)	
John	N
the boy	Det N
A little boy	Det Adj N
A boy in a bubble	Det N PP

Phrase Structure rules for NPs

NP → (Det) (Adj) N (PP)

Phrase Structure trees for NPs



10.6. Syntactic Processing

Syntactic Processing converts a at input sentence into a hierarchical structure that corresponds to the unit of meaning in the sentence. Components of a syntactic processing

1. Grammar
2. Parser

Grammar: Is a declarative representation of syntactic facts about a language. It is the specification of the legal structures of a language.

It has three different components that is

1. terminal symbol,
2. non-terminal symbols and
3. rule (productions)

Context Free Grammar (CFG)

In formal languages theory a context free grammar is where every production rule is of the form: A ! where A is a single symbol called a non-terminal and is a string that is a sequence of symbols of terminals and/ or non-terminals(possibly empty.

Terminal, Non-terminal and Start Symbols: Terminal and Non-terminal symbols are those symbols that are used to construct production rules in a formal grammar.

- Terminal symbols: Any symbol used in the grammar which does not appear on the left-hand side of some rule (ie has no definition). Terminal symbols cannot be broken down into smaller units without losing their literal meaning.
- A Non- terminal is any symbols defined by a production rule like A ! meaning that A can be replaced by therefore A is called a non-terminal symbol. A non-terminal symbol may have more than on definition in that case a symbol j is used as the union operator e.g. 1 A → α|βwhich states that whenever we see A , we can replace it with α or with β, If the rule is NP → DetN|Prop then the vertical slash on the right side is a convention to represent that the NP can be replaced either by Det N or by Prop. E.g 2 S →NPVP states that the symbol S is replaced by the symbols NP and VP
- One spacial non-terminal is called start symbol, usually written as S. The production rules for this symbol are usually written first

How Grammar works?

a) The grammar start with the start symbol. Then successively applies the production rules until it reaches to a word which contains no non-terminals which is known as the derivation.

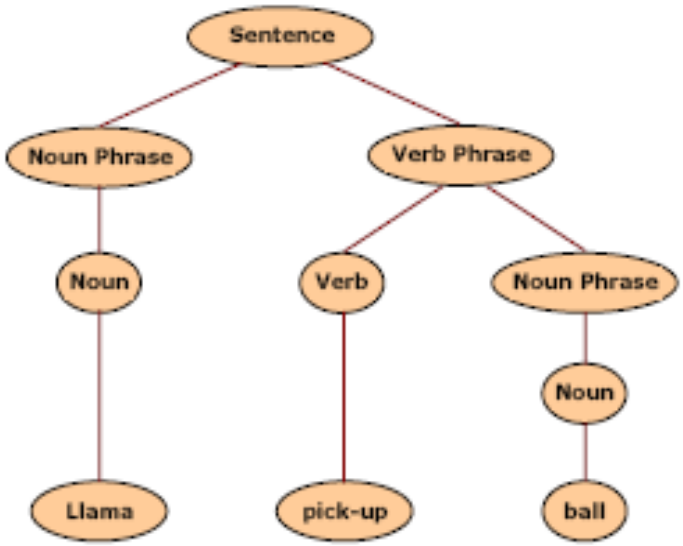
b) Anything derived from the start symbol by applying the production rules is called a sentential form.
c) A grammar may have an infinite number of sentences.
d) Example: The grammar S →Xc , X → YX, Y →a|b shows that it can derive all words which start arbitrarily and have many a's or b's and finish with a c. The language is defined by a regular expression (a|b)*C
e) Regular Expression is an expression which can be converted into a grammar.
f) A Regular grammar is a grammar which can be converted into a regular expression and the language is called a regular language.

Parser

A parser is a program that accepts as input a sequence of words in a natural language and breaks them up into parts (nouns, verbs, and their attributes) to be managed by other programming.

- Parsing is defined as the act of analyzing the grammaticality of a sentences according to some specific grammar
- It is the process of checking that a particular sequence of words in a sentence correspond to a language defined its grammar.
- Parse tree is a way of representing the out-put of a parser

Example 1: sentence "Liama pickup ball". The parse tree structure is as indicated below



Modeling a Sentence using Phase Structure

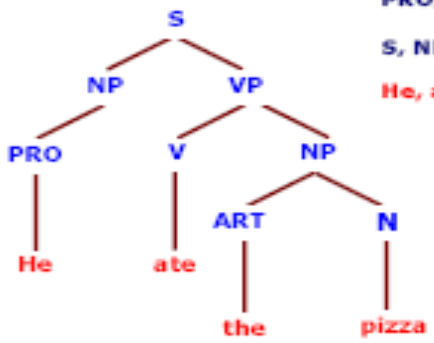
Every sentence consists of an internal structure which could be modeled with the phrase structure.

Algorithm : Steps

- Apply rules on an proposition
- The base proposition would be :
  S (the root, ie the sentence).
- The first production rule would be :
  (NP = noun phrase, VP = verb phrase)
  S -> (NP, VP)
- Apply rules for the 'branches'
  NP -> noun VP -> verb, NP
- The verb and noun have terminal nodes which could be any word in the lexicon for the appropriate category.
- The end is a tree with the words as terminal nodes, which is referred as the sentence.

Example : Parse tree

- sentence "He ate the pizza",
- apply the grammar with rules
  S -> NP VP, NP -> PRO, NP -> ART N, VP -> V NP,
- the lexicon structure is
  ("ate" V) ("he" PRO) ("pizza" N) ("the" ART)
- The parse tree is



PRO, V, ART, N - lexical non-terminals

S, NP, VP - phrasal non-terminal

He, ate, the, pizza - words or terminal

The semantics and pragmatics, are the two stages of analysis concerned with getting at the meaning of a sentence.

- In the first stage (semantics) a partial representation of the meaning is obtained based on the possible syntactic structure(s) of the sentence and the meanings of the words in that sentence.
- In the second stage (pragmatic), the meaning is elaborated based on : the contextual and the world knowledge.

For the difference between these stages, consider the sentence:

"He asked for the boss".

From knowledge of the meaning of the words and the structure of the sentence we can work out that :

- Someone (who is male) asked for someone who is a boss.
- We can't say who these people are and why the first guy wanted the second.
- If we know something about the context (including the last few sentences spoken/written) we may be able to work these things out.
- Maybe the last sentence was **"Fred had just been sacked."**
- From our general knowledge that bosses generally sack people : if people want to speak to people who sack them it is generally to complain about it.
- We could then really start to get at the meaning of the sentence : **"Fred wants to complain to his boss about getting sacked".**

References

1. Algorithmic Efficiency http://www.cprogramming.com/tutorial/computersciencetheory/algorithmicefficiency3.html
2. Artificial Intelligence Solving problems by searching, Luigi Ceccaroni, http://www.lsi.upc.edu/~luigi/MTI/AI-2006-fall/2-solving-problems-by-searching-(us).ppt.
3. Artificial Intelligence Chapter 9 Heuristic Search. Bio-intelligence Lab. http://bi.snu.ac.kr/Courses/4ai05s/Chap9.ppt
4. Backtracking, http://www.academic.marist.edu/~jzbv/algorithms/Backtracking.htm
5. Constraint Satisfaction Problems - Fahiem Bacchus http://www.cs.toronto.edu/~fbacchus/Presentations/CSP-BasicIntro.pdf
6. Elaine Rich and Kevin Knight, Carnegie Mellon University, "Artificial Intelligence", 2006.
7. Frans Coenen, University of Liverpool, Artificial Intelligence, 2CS24, http://www.csc.liv.ac.uk/~frans/OldLectures/2CS24/ai.html#definition
8. Heuristic Search http://sequoia.ict.pwr.wroc.pl/~witold/aiuwr/rpi_heuristic_search4.ps.
9. Heuristic Search http://www.cs.ubc.ca/~conati/322-2005/Module4-2005part3.pdf.
10. Heuristic Search –Jonathan Schaeffer http://www.cs.ualberta.ca/~jonathan/Courses/657/
11. Heuristic Search- Michael L. Littman http://www.cs.duke.edu/~mlittman/courses/cps271/lect-05/lect-05.html

12. Introduction to complexity, http://users.aber.ac.uk/smg/Modules/CO21120-April-2003/NOTES/15-Complexity.ppt#274,1,Introduction to complexity
13. John McCarthy, Stanford University, what is artificial intelligence? http://www-formal.stanford.edu/jmc/whatisai/whatisai.html
14. MIT, (2001), Applications of AI Massachusetts Institute of Technologys [WWW] Available from: http://web.mit.edu/STS001/www/Team7/application.html [Accessed 29 December 2011]
15. MCCARTHY J., (2000) John McCarthys' Webbpage *Stanford University* [www] Available from: http://www-formal.stanford.edu/jmc/whatisai/node3.html [Accessed 29 December 2011]
16. Stuart Russell and Peter Norvig, University of California, Artificial Intelligence: A Modern Approach, http://aima.cs.berkeley.edu/ , http://www.cs.berkeley.edu/~russell/intro.html
17. Tree Search http://www.cis.upenn.edu/~matuszek/cit594-2007/Lectures/26-tree-searching.ppt
18. Tree Search http://www.cis.upenn.edu/~matuszek/cit594-2002/Slides/tree-searching.ppt

Mt Kenya University

UNIVERSITY EXAMINATION 2010/2011

SCHOOL OF PURE AND APPLIED SCIENCES

DEPARTMENT OF INFORMATION TECHNOLOGY

EXAMINATION FOR BACHELOR OF BUSINESS INFORMATION TECHNOLOGY

BIT 3202: ARTIFICIAL INTELLIGENCE

Instructions:

*Answer question **ONE** and any other **TWO** questions* **Time: 2Hours**

QUESTION ONE (30 MARKS)

a). i). In problem solving, we have the two main search techniques. State and explain the two main category of the search technique. (4 marks)

ii). Explain the following search conditions in relation to the searching as a problem solving technique (6 marks)

1) Current state
2) Goal state
3) The solution

b) (i) Define the term knowledge representation. (4 marks)

(ii) A knowledge representation language is defined by two aspects. State and explain the two aspects (4 marks)

iii) Translate the following into first-order logic (FOL) (12 marks)

1) Some dogs bark
2) All dogs have four legs
3) Everybody likes ice cream
4) All barking dogs are irritating

QUESTION TWO (20 MARKS)

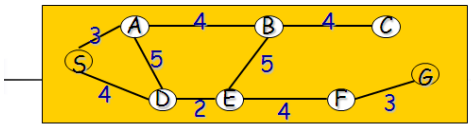
a) According to your knowledge of programming languages, what other programming language would you consider suitable for writing AI programs? Give supporting reasons for your answer. (5marks)

b) State any four social implications of AI (5marks)

c) What advances do you think need to be made in order for the Turing Test to be passed? (5mark)

QUESTION THREE (20 MARKS)

(a) Based on the algorithm of Depth-first search, trace the search of the following path (12 marks)



(b) Answer the following questions by indicating whether the statements are True (T) or False (F) (8 marks)

(a) Depth-first search is often slower than Breadth-first search.
(b) Draughts (checkers) and Scrabble are both deterministic games.
(c) When playing games, the horizon effect can be solved by limiting the search depth.
(d) A rational intelligent agent acts in such a way as to minimize its expected value of performance measure given the percept sequence to date.
(e) "Two primary school children, Dennis and Sarut are playing the Tic-tac-toe game. Dennis makes the first move (starts the game)." The minimum number of moves Sarut could make is 2 for Dennis to win the game.
(f) Is the following First-Order Logic interpretation of the statement "All men except butchers like vegetarians"
$\forall x \forall y \ male(x) \land \neg butcher(x) \land vegetarian(y) \Rightarrow likes(y,x)$
(g) Is the following First-Order Logic interpretation of the statement " Everyone likes someone "
$\forall x \ \exists y \ likes(x,y)$
(h) Is the following First-Order Logic interpretation of the statement " Someone is liked by everyone"
$\exists y \ \forall x \ likes(x,y)$

QUESTION FOUR (20 MARKS)

(a) (i) What is an Expert System Shell and how is it used? (3mrks)

(ii)What is the function of the Inference Engine in an Expert Systems? (3mrks)

(iii) Why is the prototyping approach used in Expert System development? (2mrks)

(iv) Why is knowledge sometimes difficult to extract from experts? (3mrks)

(v) Describe briefly the four primary phases of building an Expert System (3mrks)

(b) What makes Lisp and Prolog the most widely used languages for developing AI programs (3mrk)

c). What is the widely used criterion for determining the success of an AI system? Explain the working of this criterion (3mrks)

QUESTION FIVE (20 MARKS)

a) Discuss why agents in Artificial Intelligence need not only be software entities. (2 marks)

b) Describe any four characteristics of intelligent agents. (2 marks)

c) Describe an architecture of a simple reflex agent. (4 marks)

d) Describe statistical classification technique. (3 marks)

e) i) Let X = (x_1,x_2,x_3) and Y = (y_1,y_2,y_3). Define the square-block distance $d(X,Y)_{sqb}$ between X and Y. (2 marks)

(ii) Use k-Nearest Neighbour technique and square-block distance measure to classify an object P(10,30,5) given the data below:-

X	Y	Z	Category
30	30	10	A
20	10	20	A
10	20	20	B
15	40	60	A
6	5	10	B
20	10	15	B
13	15	7	B
40	70	70	A

(7 marks)

Mt Kenya University

UNIVERSITY EXAMINATION 2010/2011
SCHOOL OF APPLIED AND SOCIAL SCIENCES
DEPARTMENT OF EDUCATION

EXAMINATION FOR BACHELOR OF EDUCATION SCHOOL BASED PROGRAMME

BIT 3202: ARTIFICIAL INTELLIGENCE

Instructions:

*Answer question **ONE** and any other **TWO** questions* **Time: 2Hours**

QUESTION ONE (30 MARKS)

(a) Discuss any two reasons why psychology may be regarded as a foundation of Artificial Intelligence. State two other foundations of Artificial Intelligence. (4 marks)

(b) (i) Discuss one advantage and one disadvantage of informed search as a problem solving technique. (1 marks)

(ii) Show how a search problem may be specified. (5 marks)

(iii) Describe the best-first search heuristic using an example. Explain why you would recommend such a search method. (8 marks)

(iv) State any two real life applications of the search technique. (2 marks)

(c). Describe predicate calculus and frames as knowledge representation formalisms. State one advantage and one limitation of each of these knowledge representation formalisms. (10 marks)

QUESTION TWO (20 MARKS)

a). (i) In problem solving, we have the two main search techniques. State and explain the two main category of the search technique. (4 marks)

i. Uninformed (Blind) search
ii. Informed (heuristic) search

(ii) Explain the following search conditions in relation to the searching as a problem solving technique (6 marks)

i. Current state
ii. Goal state
iii. The solution
iv. Where one is

b) (i) Define the term knowledge representation. (4 marks)

Framework for knowledge and manipulating knowledge of set of syntactic and semantic conventions that makes it possible to describe things.

(ii). A knowledge representation language is defined by two aspects. State and explain the two aspects (4 marks)

1) Syntax
2) Semantics

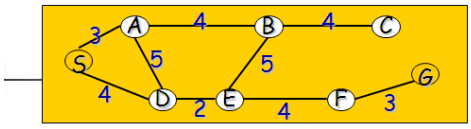
iii) Translate the following into first-order logic (FOL) (12 marks)

1) Some dogs bark
2) All dogs have four legs
3) Everybody likes ice cream
4) All barking dogs are irritating

x.Dogx ^ Bark x
V x .~~Dog~~x }yzwu. Leg y ^legz^leg w ^ leg u
^ has xy ^ has xz ^ has xw ^ has xu ^y
Vx likes (x, icecream)
Vx. Dog x ^ Bark~~ing~~x irritating x

QUESTION THREE (20 MARKS)

(a) Based on the algorithm of Depth-first search, trace the search of the following path (12 marks)



(a) Discuss why agents in Artificial Intelligence need not only be software entities. (2 marks)

(b) Describe any four characteristics of intelligent agents. (2 marks)

(c) Describe architecture of a simple reflex agent. (4 marks)

QUESTION FOUR (20 MARKS)

a). Answer the following questions by indicating whether the statements are True (T) or False (F) (8 marks)

(a) Depth-first search is often slower than Breadth-first search.
(b) Draughts (checkers) and Scrabble are both deterministic games.
(c) When playing games, the horizon effect can be solved by limiting the search depth.
(d) A rational intelligent agent acts in such a way as to minimize its expected value of performance measure given the percept sequence to date.
(e) "Two primary school children, Dennis and Sarut are playing the Tic-tac-toe game. Dennis makes the first move (starts the game)." The minimum number of moves Sarut could make is 2 for Dennis to win the game.
(f) Is the following First-Order Logic interpretation of the statement "All men except butchers like vegetarians"
$\forall x \forall y \ male(x) \land \neg butcher(x) \land vegetarian(y) \Rightarrow likes(y,x)$
(g) Is the following First-Order Logic interpretation of the statement " Everyone likes someone "
$\forall x \ \exists y \ likes(x,y)$
(h) Is the following First-Order Logic interpretation of the statement " Someone is liked by everyone"
$\exists y \ \forall x \ likes(x,y)$

b). Use k-Nearest Neighbour technique and square-block distance measure to classify an object P(10,30,5) given the data below:-

X	Y	Z	Category
30	30	10	A
20	10	20	A
10	20	20	B
15	40	60	A
6	5	10	B
20	10	15	B
13	15	7	B
40	70	70	A

(12 marks)

QUESTION FIVE (20 MARKS)

(a) Give a more recent definition of a robot. (2 marks)

(b) Discuss any three grounds showing why we should have robots. (3 marks)

(c) Describe locomotion and sensors in robots. (6 marks)

d). Explain the meaning of the terms supervised learning, hypothesis and example with respect to Machine Learning in Artificial Intelligence. (3 marks)

(e) Use a diagram to describe a structure of a learning system. (4 marks)

(f) State any two techniques used in Machine Learning. (2 marks)