Artificial Intelligence and Robotics

Code: EC 704A

Semester: 7th

Credits: 3

Preparted by: Dr. Soumik Podder

Course objectives:

Artificial Intelligence and Robotics lead to an exciting, evolving career that is predicted to grow sharply by upcoming years. The course will impact all segments of daily life, with applications in a wide range of industries such as healthcare, transportation, insurance, transport and logistics and even customer service. This course emphasis the applications of AI and robotics that exists in every field as companies seek to give machine the ability to think, learn and adapt.

Course outcomes:

CO-1:
Able to understand the notion of searching algorithms to explore a dataset.
CO-2:
Able to represent information about the world in a form that a computer system can utilize to solve complex tasks.
CO-3:
Able to describe how a machine automatically learn and improve from experience without being explicitly programmed.
CO-4:
Able to understand the position and rotation of a robot by changing different joints using sensors and motors.
CO-5:
Able to identify direct and inverse kinematics problem of a robot.
CO-6:
Able to apply trajectory planning in joint and cartesian space using PID controller.

Prerequisites:

- Linear algebra and probability theory
- Knowledge of control systems
- Concept of basic programming

Module 1 Introduction

Introduction to Artificial Intelligence

McCarthy coined the Term AI in 1956. Now AI is the buzzword ringing your mind in every field of study. Researchers are doing extreme in every aspect to accommodate the concept of AI is all recent application. Now a day, industry is demanding young professional with the basic knowledge of AI. The basic of AI is human brain: how it acquires the data and how the data are processed to produce information as well as to take decision. This philosophy you have to analyze mathematically for mimicking the same concept into a machine. The machine then acquires the machine intelligence or Artificial Intelligence to serve as like a human being. Thus AI is the guide of future Information Technology.

Definition of AI:
Artificial Intelligence (AI) is the technique to create machine, which acts intelligently as like a smart human being.

The term ‘artificial intelligent’ refers to the ‘behavior of a machine’ if it acts rationally. The intelligent behavior acquired by intelligent programming.

The science of AI could be described as "synthetic psychology," experimental philosophy," or "computational epistemology"; epistemology is the study of knowledge.

Finally, AI is concerned with the study of building machines/computers that simulate human behavior. Traditional computing (numeric information is processed by algorithm), cannot tackle symbolic information and heuristic processing but AI is capable of both.

Goals of AI

a) To Create Expert Systems − the systems which exhibit intelligent behavior, learn, demonstrate, explain, and advice its users. The first expert system was MYCIN developed by Stanford University using LISP.
b) To Implement Human Intelligence in Machines − Creating systems that understand, think, learn, and behave like humans.

AI technique

AI technique is a method that exploits knowledge that should be represented in such a way that:

a) The knowledge captures generalizations.
b) It can be understood by people who must provide it.
c) It can easily be modified to correct errors
d) It can be used in a great many situations even if it is not entirely accurate.

Different AI techniques are:

e) Natural Language Processing: Techniques for Speech Synthesis
f) Machine Learning in Medicine: Multilayer perceptrons , Bayesian networks
g) Cognitive Modelling: Artificial Neural Networks
h) Knowledge-Based Systems: Intelligent information visualization, Integration of information technology with telecommunication

Foundations and History of Artificial Intelligence

About 400 years ago people started to write about the nature of thought and reason. Hobbes (1588-1679), who has been described by Haugeland (1985), p. 85 as the "Grandfather of AI," espoused the position that thinking was symbolic reasoning like talking out loud or working out an answer with pen and paper. The idea of symbolic reasoning was further developed by Descartes (1596-1650), Pascal (1623-1662), Spinoza (1632-1677), Leibniz (1646-1716), and others who were pioneers in the philosophy of mind.

The idea of symbolic operations became more concrete with the development of computers. The first general-purpose computer designed (but not built until 1991, at the Science Museum of London) was the Analytical Engine by Babbage (1792-1871). In the early part of the 20th century, there was much work done on understanding computation. Several models of computation were proposed, including the Turing machine by Alan Turing (1912-1954), a theoretical machine that writes symbols on an infinitely long tape, and the lambda calculus of Church (1903-1995), which is a mathematical formalism for rewriting formulas. It can be shown that these very different formalisms are equivalent in that any function

computable by one is computable by the others. This leads to the Church-Turing thesis:
Any effectively computable function can be carried out on a Turing machine (and so also in the lambda calculus or any of the other equivalent formalisms).

Turing test:

The Turing test, developed by Alan Turing in 1950, is a test of a machine's ability to exhibit intelligent behavior equivalent to, or indistinguishable from, that of a human.

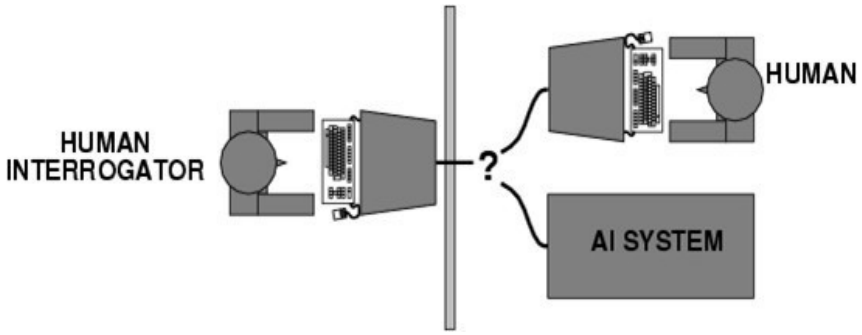


Fig.1.1 Turing Test

A ‘human interrogator’ as judge engages in a natural language conversation with two other parties, one a ‘human’ and the other an ‘AI System(machine)’; if the judge cannot reliably tell which is which, then the machine is said to pass the test. It is assumed that both the human and the machine try to appear human. The ‘machine can think’, is the conclusion of this test.
So, AI is a system which can think and act humanly as well as rationally.

Intelligent Agents and Environment

An agent is defined as anything which perceiving its environment through sensors and acting upon that environment through actuators or effectors. Intelligent programming controls the behavior of the agent.

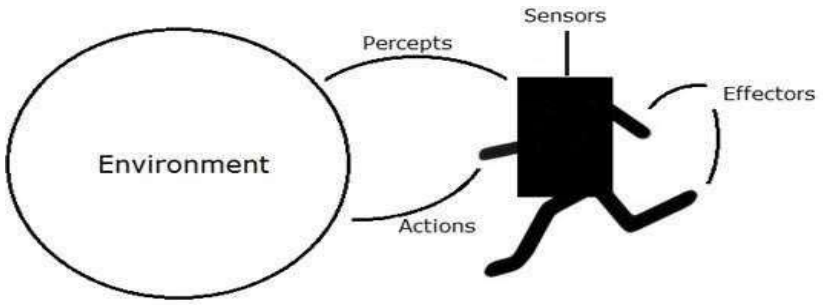


Fig.1.2 Agent system/ architecture

Intelligent Agent (IA) is the agent, which acts rationally that means, does the right thing at right moment with having good sense of judgment.

The Structure of Intelligent Agents

Agent's structure can be viewed as agent's architecture consists of machinery that an agent executes on and agent program is the implementation of an agent function. Agent's structure can be viewed as:

a) Agent = Architecture + Agent Program
b) Architecture = the machinery that an agent executes on.
c) Agent Program = an implementation of an agent function.

The *agent system* architecture is the hardware upon which the program runs. This system may be physical, or may be virtual as in the case of software agents (soft-bots) which live in a world of software.

Types of Agent

Human *agent*- perceives the environment through eyes, ears, & other organs for sensors and acting upon that environment through hands, legs, mouth, & other body parts for actuators

Robotic agent- perceives the environment through cameras and infrared range finders for sensors and acting upon that environment through various motors for actuators

Software agent- perceives the environment through input data for sensor and acting upon that environment through output signal via actuator.

1.7 Basic types of agent in order of increasing generalization

.1 Simple Reflex Agents

They choose actions only based on the current percept. They are rational only if a correct decision is made only based on current precept.

Their environment is observable. It responds immediately to the single percepts. Condition-Action Rule − It is a rule that maps a state (condition) to an action.

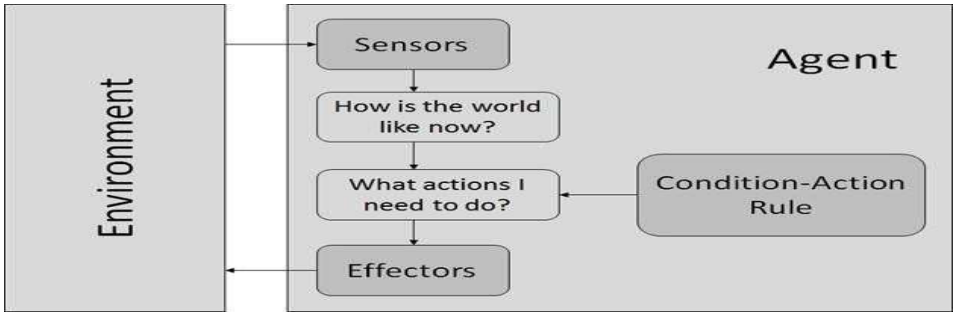


Fig.1.3 Simple Riflex Agent environment

Model Based Reflex Agents

They use a model of the world to choose their actions. They maintain an internal state. The knowledge about how the things happen in the world.

Internal State is a representation of unobserved aspects of current state depending on percept history.

Updating the state requires the information about −How the world evolves. How the agent's actions affect the world.

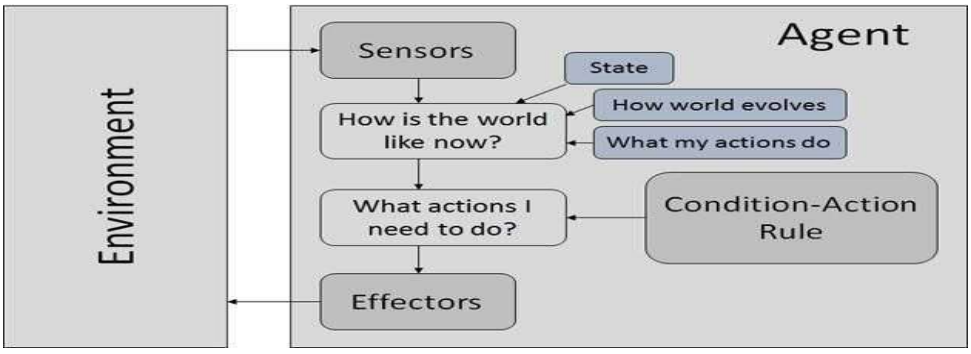


Fig.1.4 Model Based Riflex Agent environment

Goal Based Agents

They choose their actions in order to achieve goals. Goal-based approach is more flexible than reflex agent since the knowledge supporting a decision is explicitly

modeled, thereby allowing for modifications. It acts to achieve goals. Goal is the description of desirable situations.

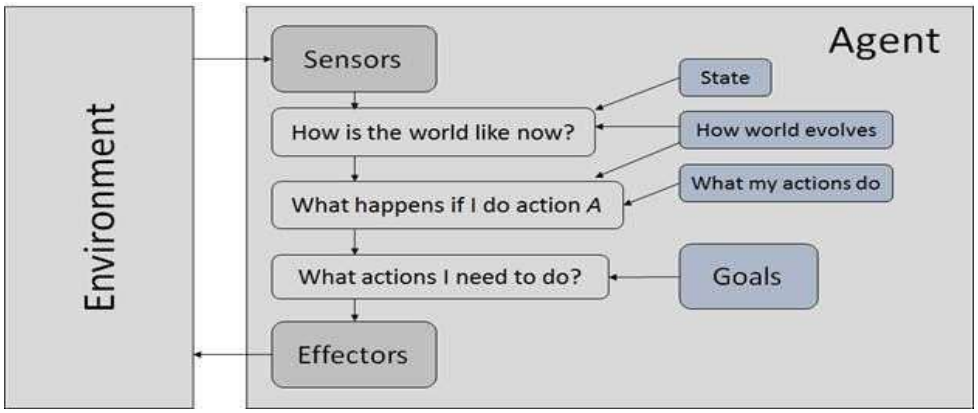


Fig.1.5 Goal Based Agent environment

Utility Based Agents

They choose actions based on a preference (utility) for each state. Goals are inadequate when −There are conflicting goals, out of which only few can be achieved.

Goals have some uncertainty of being achieved and you need to weigh likelihood of success against the importance of a goal. It acts to maximize their "happiness".

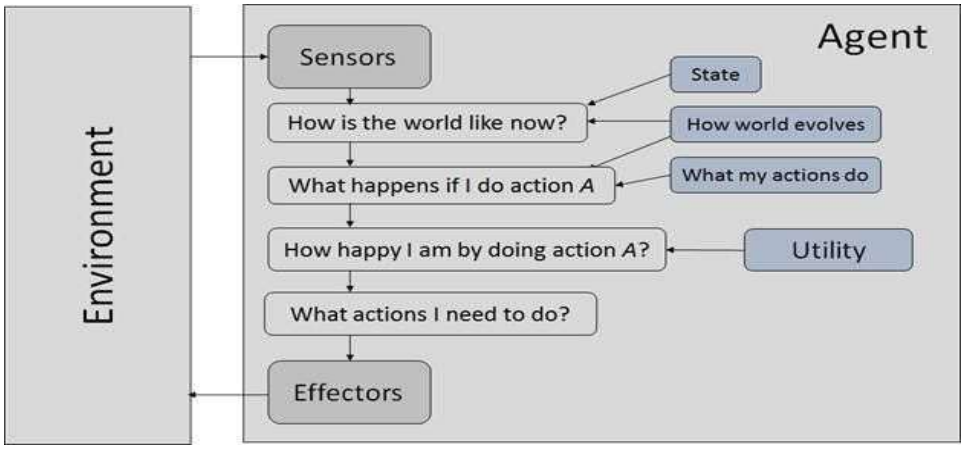


Fig.1.6 Utility Based Agent environment

1.7.4 Table Driven Agent

Table-driven agents, which respond to the percepts sequence seen so far. The advantage table driven agent is to do what we want; it implements the desired agent function.

Rational Agent

A rational agent always performs right action, where the right action means the action that causes the agent to be most successful in the given percept sequence.

Rationality

Rationality is nothing but status of being reasonable, sensible, and having good sense of judgment. Rationality is concerned with expected actions and results depending upon what the agent has perceived. Performing actions with the aim of obtaining useful information is an important part of rationality.

Ideal Rational Agent

An ideal rational agent is the one, which is capable of doing expected actions to maximize its performance measure, on the basis of (a) its percept sequence, and (b) its built-in knowledge base. Rationality of an agent depends on the following four factors:

(a) The performance measures, which determine the degree of success.
(b) Agent's Percept Sequence till now.
(c) The agent's prior knowledge about the environment. (d) The actions that the agent can carry out.

A rational agent always performs right action, where the right action means the action that causes the agent to be most successful in the given percept sequence.
The problem, the agent solves is characterized by
Performance measure, Environment, Actuators, and Sensors (PEAS).

Production System (or production rule system)

A production system is defined as a set of rules, which is coded in computer program used to provide some form of artificial intelligence. The set of rules is about human behavior.

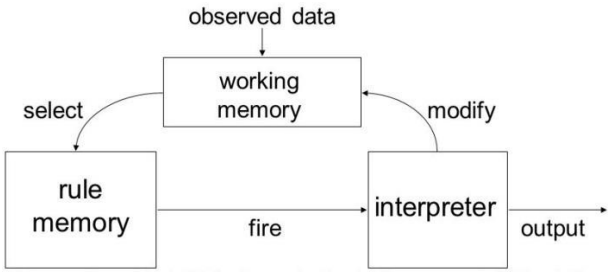


Fig.1.7 Architecture of a production system

Components of production system

A production system consists of the following components:

a) A set of rule,
b) Working memory with database,
c) A control strategy with rule memory and
d) A rule applier /interpreter.

Types of production system

There are four types of production system:

1. Non-monotonic production system- is a system when there are many rules applicable at time, and then you can select any one of them at current time. If the applied rule does not lead to a solution then this system does not allow applying the rest of the rules.

2. Monotonic production system- is a system when there are many rules applicable at time, and then you can select any one of them at current time. If the applied rule does not lead to a solution then this system allow applying the rest of the rules.

3. Partially commutative production system- is a system in which the application of a particular sequence of rules transforms state X into state Y, then any permutation of those rules that is allowable also transforms state X into state Y.
4. Commutative production system- is a system with the property of both monotonic and partially commutative.

Conflict resolution in Production System

This strategy used for choosing which production rule to fire. The need for such a strategy arises when the conditions of two or more rules are satisfied by the currently known facts. Conflict resolution strategies fall into several categories:

1. Refractoriness: Do not allow a rule to fire twice on same data.
2. Recency: Take the data, which arrived in working memory most recently, and find a rule that uses this data.
3. Specificity: Use the most specific rule (the one with the most conditions attached).
4. Order: Pick the first applicable rule in order of presentation.
5. Arbitrary choice: Pick a rule at random.

Benefits of Production System

a) Production systems provide an excellent tool for structuring AI programs.
b) Production Systems are highly modular because the individual rules can be added, removed or modified independently.
c) The production rules are expressed in a natural form, so the statements contained in the knowledge base acquire a recording of an expert thinking.
d) It is language independent.

Short questions and Answer

1. What is Agent? How many types of agents are there?

A: An agent is anything that can be viewed as perceiving and acting upon the environment through the sensors and actuators.

The four types of agents are Simple reflex, Model based, Goal based and Utility based agents.

2. What is percept sequence? [WBUT 2016 CSE] ***How does the agent work?*** A: An agent's percept sequence is the complete history of everything that the agent has ever perceived.

An agent program will implement function-mapping percepts to actions. When the environment becomes more tricky means, the agent needs plan and search action sequence to achieve the goal.

3. How does the Simple reflex agent work? How can agent improve its performance?

A: Simple reflex agent is based on the present condition and so it is condition action rule.
An agent can improve its performance by storing its pervious actions. This process is also known as learning.

11. What is Problem generator?

A: Problem generator will give the suggestion to improve the output for learning agent.

11. What is utility function?

A: A utility function maps a state onto a real number, which describes the associated degree of happiness

11. What is AI?

A: AI is to make things work automatically through machine without using human effort. Machine will give the result with just giving input from human.
That means the system or machine will act intelligently as per the requirement.

11. What do you mean by agent behavior?

A: An agent's behavior is described by the agent function that maps any given percept sequence to an action, which can be implemented by agent program. The agent function is an abstract mathematical description; the agent program is a concrete implementation, running on the agent architecture.

8. What is rational agent?

A: Rational agent is the one who always does the right thing, Right in a sense that it makes the agent the most successful.
For each possible percept sequence, a rational agent should select an action that is expected to maximize its performance measure, given the evidence provided by the percept sequence and whatever built-in knowledge the agent has.

9. What is an omniscient agent? Is it possible in real world?

A: An omniscient agent knows the actual outcome of its actions and can act accordingly; but omniscience is impossible in reality. Therefore, in reality, Rational Agent is possible which always does the right thing.

10. What is PEAS?

A: The task in environment of an agent is described by four parts: performance measures, environment, actuators and sensors, which is generally known as the PEAS descriptions.

11. State and Explain the PEAS attributes of a Medical Diagnostic System. A: Performance measures, Environment, Actuators and Sensors are known as the PEAS descriptions.

Performance measures attributes of a Medical Diagnostic System: Healthy patient, minimize costs, lawsuits, etc.

Environment attributes of a Medical Diagnostic System: Patient, hospital, staff, etc.

Actuators attributes of a Medical Diagnostic System: Screen display such as questions, tests, diagnoses, treatments, referrals, etc.

Sensors attributes of a Medical Diagnostic System: Keyboard such as entry of symptoms, findings, patient's answers, etc. Medical diagnosis is a forward reasoning strategy.

Multiple Choice Questions (MCQ)

1. Which instruments are used for perceiving and acting upon the environment?
a) Sensors and Actuators
b) Sensors
c) Perceiver
d) None of the mentioned **3.**

2. What is meant by agent's percept sequence?
a) Used to perceive the environment How many types of agents are there in artificial intelligence?
a) **Error! Bookmark not defined.**
b) 11
c)d) 4 **Error! Bookmark not defined.**

b) Complete history of actuator
c) Complete history of perceived things
d) Both a & b

4. What is the rule of simple reflex agent?
a) Simple-action rule
b) Condition-action rule
c) Both a & b
d) None of the mentioned
View Answer

5. What are the composition for agents in artificial

intelligence? a) Program
b) Architecture
c) Both a & b
d) None of the mentioned View Answer

6. In which agent does, the problem generator is present? a) Learning agent
b) Observing agent
c) Reflex agent
d) None of the mentioned

7. Which is used to improve the agents performance? a) Perceiving
b) Learning
c) Observing
d) None of the mentioned

8. Which agent deals with happy and unhappy states? a) Simple reflex agent
b) Model based agent
c) Learning agent
d) Utility based agent

9. Which action sequences are used to acheive the agent's goal?
a) Search
b) Plan
c) Reterive
d) Both a & b

10. What is Artificial intelligence?
a) Putting your intelligence into Computer
b) Programming with your own intelligence
c) Making a Machine intelligent
d) Playing a Game
e) Putting more memory into Computer

11. Which is not the commonly used programming language for AI?
a) PROLOG
b) Java
c) LISP
d) Perl

12. Artificial Intelligence has its expansion in the following application.
a) Planning and Scheduling
b) Game Playing
c) Diagnosis and Robotics
d) All of the above

13. What is perception sequence of an agent?
a) A periodic inputs sets

b) a complete history of everything the agent has ever perceived
c) Both a) and b)
d) None of the mentioned

14. Agents behavior can be best described by
a) Perception sequence
b) Agent function
c) Sensors and Actuators
d) Environment in which agent is performing

15. Satellite Image Analysis System is
a) Episodic
b) Semi-Static
c) Single agent
d) Partially Observable

16. An agent is composed of,
a) Architecture

17. In which of the following agent does the problem generator is present?
a) Learning agent
b) Observing agent
c) Reflex agent
d) None of the mentioned

18. The first expert system was **[WBUT 2012 IT]** a) MYCIN
b) DENDRAL
c) EMYCIN
d) PROSPECTOR

19. AI is described as
a) Synthetic psychology
b) Experimental philosophy
c) Computational epistemology
d) All of these

MCQ Answer keys

b) Agent Function
c) Perception Sequence
d) Architecture and Program

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
a	c	d	b	c	a	b	d	d	c	d	d	c	b	d	d	d	a	d

Test Your Skills

1. What is Artificial Intelligence?
2. What are AI techniques? [WBUT 2013 IT]
3. What is epistemology?
4. Briefly explain Turing Test with example. [WBUT 2009 IT]
5. What is an agent? [WBUT 2016 CSE] What are the advantages of table driven agent? [WBUT 2011 CSE]
6. Describe various types of agent. [WBUT 2016 CSE]
7. What is Rational Agent? How it differ from Omniscient Agent?
8. What is PEAS?
9. State and explain the PEAS attributes of a taxi driver.
10. State and explain different types of environment.
11. What is production system? [WBUT 2016 CSE] Explain conflict resolution strategies. [WBUT 2010,11 IT]
12. State the components of Production System. [WBUT 2013 IT]
13. Discuss the benefit of production system. [WBUT 2010 CSE]
14. What is agent system? [WBUT 2016 CSE]

Module 2

Searching and Problem Solving

State Space Search

State space search characterize problem solving as the process of finding a solution path from the start state to a goal state.

Strategies for State Space Search

Data driven search (forward chaining)

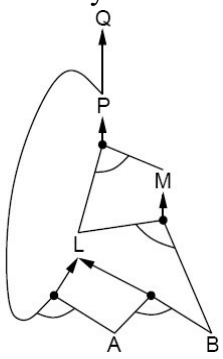
It begins with the given facts and a set of legal moves or rules for changing the state. Searches proceed by applying rules to facts to produce new facts, which are in turn used by rules to generate more new facts. This process continues until it generates a path that satisfies the goal condition.

Example: AND-OR Graph

It is an instance of forward chaining. Multiple links joined by an arc indicate conjunction – every link must be proved.

Multiple links without an arc indicate disjunction – any link can be proved

$$P \Rightarrow Q$$
$$L \wedge M \Rightarrow P$$
$$B \wedge L \Rightarrow M$$
$$A \wedge P \Rightarrow L$$
$$A \wedge B \Rightarrow L$$
$$A$$
$$B$$

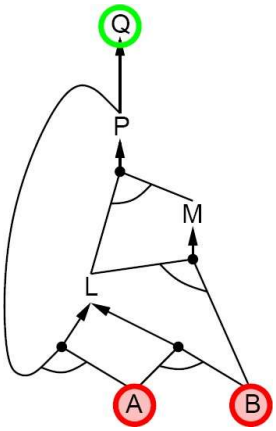


For each new piece of data, generate all new facts, until the desired fact is generated (Data-directed reasoning).

Goal driven search (Backward chaining)

Begin with goal to see what rules or legal moves could used to generate this goal, and determined what conditions must be true to use them. These conditions become the new goals, sub-goals, for the search. This process continues, working backward through successive sub-goals, until a path is generated that leads back to the facts of the problem.

Example: To prove the goal, find a clause that contains the goal as its head, and prove the body recursively (Goal-directed reasoning)



Uninformed search strategies

A search strategy is defined by picking the order of node expansion. Uninformed search strategies use only the information available in the problem definition.

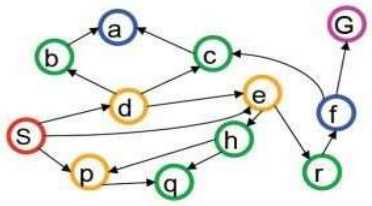
Uninformed search strategies are Breadth-first search, Depth-first search, Iterative deepening search and Uniform-cost search.

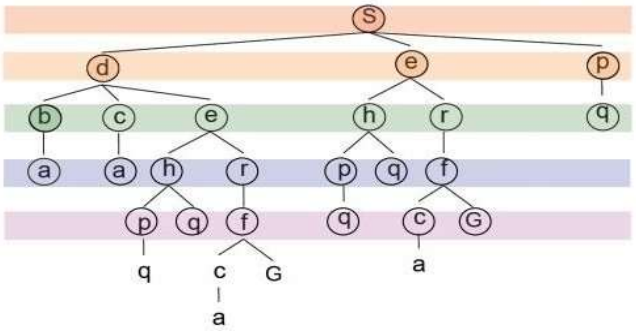
Breadth First Search (BFS)

There are many ways to traverse graphs. BFS is the most commonly used approach. BFS is a traversing algorithm where you should start traversing from a selected node (source or starting node) and traverse the graph layer wise thus exploring the neighbor nodes (nodes which are directly connected to source node). You must then move towards the next-level neighbor nodes. As the name BFS suggests, you are required to traverse the graph breadth wise as follows:

✓ First move horizontally and visit all the nodes of the current layer
✓ Move to the next layer. ✓ Repeat this process until the queue is empty.

- Expansion order:
  (S,d,e,p,b,c,e,h,r,q,a,a,
  h,r,p,q,f,p,q,f,q,c,G)





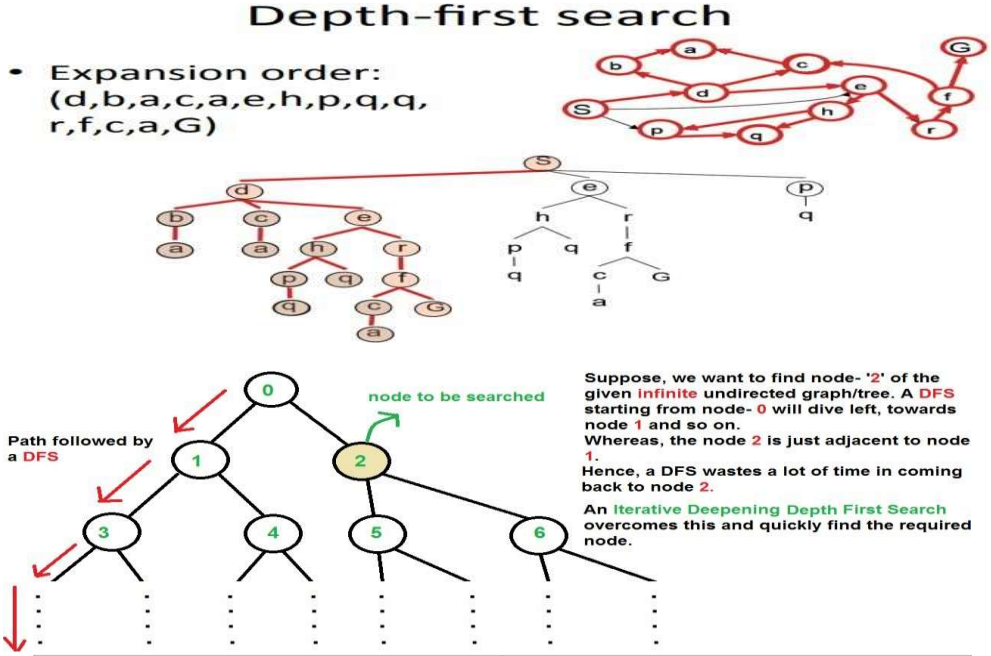
Depth First Search (DFS)

The DFS algorithm is a recursive algorithm that uses the idea of backtracking. It involves exhaustive searches of all the nodes by going ahead, if possible, else by backtracking.

Here, the word backtrack means that when you are moving forward and there are no more nodes along the current path, you move backwards on the same path to find nodes to traverse. All the nodes will be visited on the current path till all the unvisited nodes have been traversed after which the next path will be selected.

This recursive nature of DFS can be implemented using stacks. The basic idea is as follows:

- ✓ Pick a starting node and push all its adjacent nodes into a stack.
- ✓ Pop a node from stack to select the next node to visit and push all its adjacent nodes into a stack.
- ✓ Repeat this process until the stack is empty. However, ensure that the nodes that are visited are marked. This will prevent you from visiting the same node more than once. If you do not mark the nodes that are visited and you visit the same node more than once, you may end up in an infinite loop.



DFS	BFS
Tests all possibilities in one path until it needs to backtrack	Visit all nearest neighbors and then their neighbours
DFS goes to bottom of a subtree, then backtracks	BFS finds shortest path to destination
Both run till no unvisited location left	
STACK Data Structure to keep track of next location to visit	QUEUE Data Structure to keep track of next location to visit

Iterative deepening search/ Iterative deepening DFS

IDDFS combines depth-first search's space-efficiency and breadth-first search's fast search (for nodes closer to root).

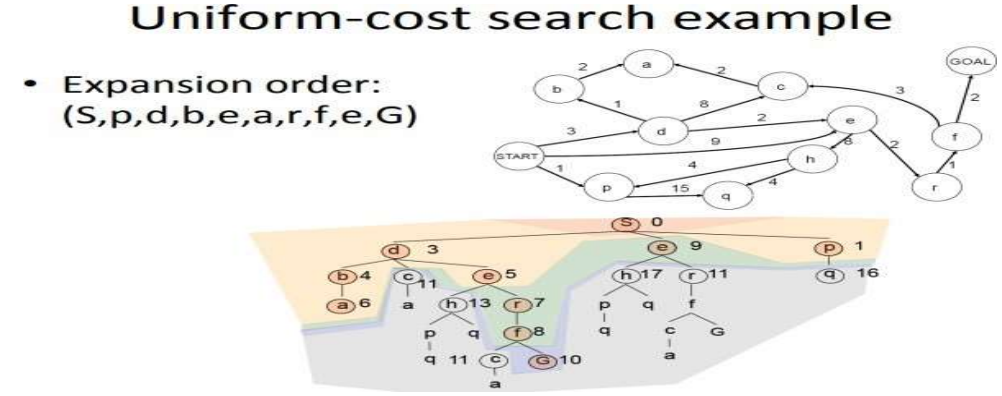
How does IDDFS work?

IDDFS calls DFS for different depths starting from an initial value. In every call, DFS is restricted from going beyond given depth. So basically we do DFS in a BFS fashion.

To avoid the infinite depth problem of DFS, we can decide to only search until depth L, i.e. we don't expand beyond depth L. Hence the search is called *Depth-Limited Search.*

Uniform-cost search

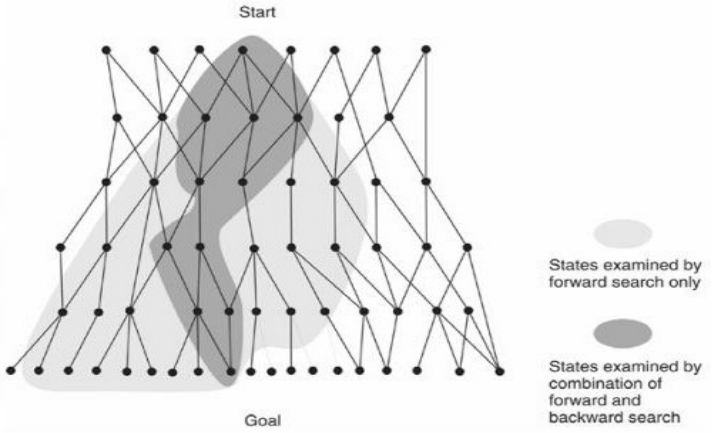
In this search, expand the frontier node with the lowest path cost.
Implementation: frontier is a priority queue ordered by path cost.
This search is equivalent to breadth-first if step costs are all equal.



Bidirectional search Algorithm

Bidirectional search involves alternate searching from the start state toward the goal and from the goal state toward the start. The algorithm stops when the frontiers intersect. A search algorithm has to be selected for each half. How does the algorithm know when the frontiers of the search tree intersect?

For bidirectional search to work well, there must be an efficient way to check whether a given node belongs to the other search tree. Bidirectional search can sometimes lead to finding a solution more quickly. The reason can be seen from inspecting the following figure.



Also note that the algorithm works well only when there are unique start and goal states.

4.4 Analysis of search strategies

Analysis of search strategies

- Strategies are evaluated along the following criteria:
  - **Completeness:** does it always find a solution if one exists?
  - **Optimality:** does it always find a least-cost solution?
  - **Time complexity:** number of nodes generated
  - **Space complexity:** maximum number of nodes in memory
- Time and space complexity are measured in terms of
  - ***b:*** maximum branching factor of the search tree
  - ***d:*** depth of the optimal solution
  - ***m***: maximum length of any path in the state space (may be infinite)

Review: Uninformed search strategies

Algorithm	Complete?	Optimal?	Time complexity	Space complexity
BFS	Yes	If all step costs are equal	$O(b^d)$	$O(b^d)$
DFS	No	No	$O(b^m)$	O(bm)
IDS	Yes	If all step costs are equal	$O(b^d)$	O(bd)
UCS	Yes	Yes	Number of nodes with $g(n) \leq C^*$	

b: maximum branching factor of the search tree
d: depth of the optimal solution
m: maximum length of any path in the state space
C*: cost of optimal solution
g(n): cost of path from start state to node n

Limitation of Uninformed search

- ✓ Depth-first search and breadth-first search are examples of blind (or uninformed) search strategies. ✓ Breadth-first search produces an optimal solution (eventually, and if one exists), but it still searches blindly through the states pace.
- ✓ Neither uses any knowledge about the specific domain in question to search through the state-space in a more directed manner.
- ✓ If the search space is big, blind search can simply take too long to be practical, or can significantly limit how deep we're able to look into the space.

Informed search strategies are introduced here to eliminate the above limitation.

Informed search strategies

A search strategy, which searches the most promising branches of the state-space first can: – find a solution more quickly,

– find solutions even when there is limited time available,

– find a better solution, since more profitable parts of the state-space can be examined, while ignoring the unprofitable parts.

- An evaluation function f(n) determines how promising a node n in the search tree appears to be for the task of reaching the goal.
- Best-first search chooses to expand the node that appears by evaluation function to be most promising the among the candidates.

- Traditionally, one aims at minimizing the value of function f.
- A key component of an evaluation function is a heuristic function h(n), which estimates the cost of the cheapest path from node n to a goal node.
- In connection of a search problem "heuristics" refers to a certain (but loose) upper or lower bound for the cost of the best solution.
- Goal states are nevertheless identified: in a corresponding node n it is required that h(n) = 0.
- Heuristic functions are the most common form in which additional knowledge is imported to the search algorithm.

A search strategy which is better than another at identifying the most promising branches of a search-space is said to be more informed.

Now, follow some inform search strategies:

Greedy best-first search

- Greedy best-first 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.
- Thus, the evaluation function is f(n) = h(n).
- E.g. in minimizing road distances a heuristic lower bound for distances of cities is their straight-line distance.
- Greedy search ignores the cost of the path that has already been traversed to reach n.
- Therefore, the solution given is not necessarily optimal
- If repeating states are not detected, greedy best-first search may oscillate forever between two promising states.

- Because greedy best-first search can start down an infinite path and never return to try other possibilities, it is incomplete.
- Because of its greediness the search makes choices that can lead to a dead end; then one backs up in the search tree to the deepest unexpanded node.
- Greedy best-first search resembles depth-first search in the way it prefers to follow a single path all the way to the goal, but will back up when it hits a dead end.
- The worst-case time and space complexity is O(b*m), where m=depth of search.
- The quality of the heuristic function determines the practical usability of greedy search.

Example:

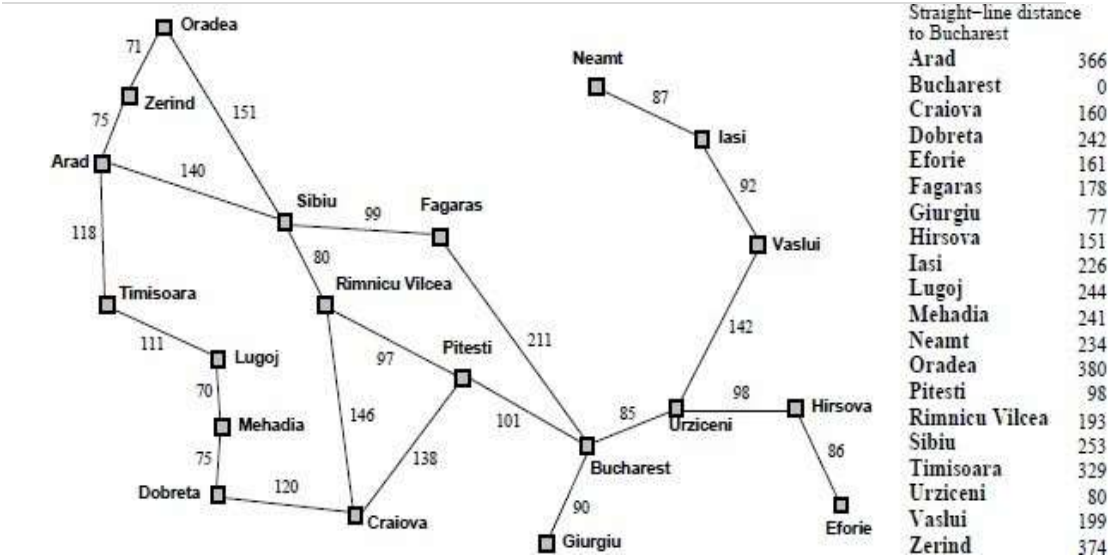


Fig1: Romania with Step Costs in km

Solution:

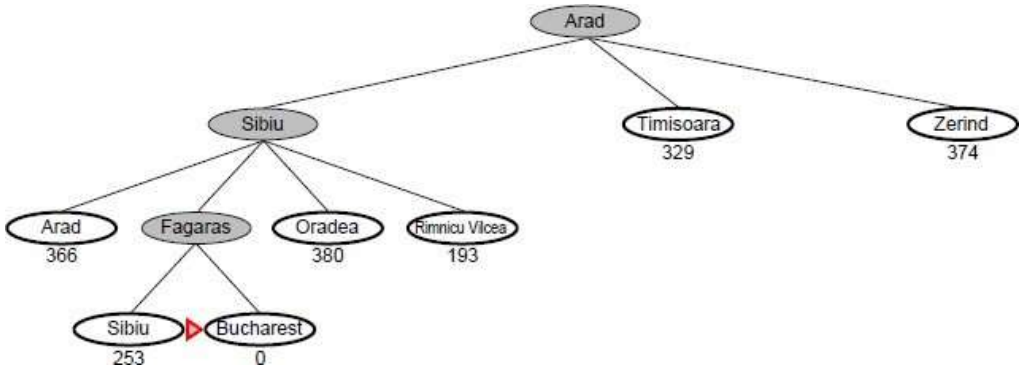


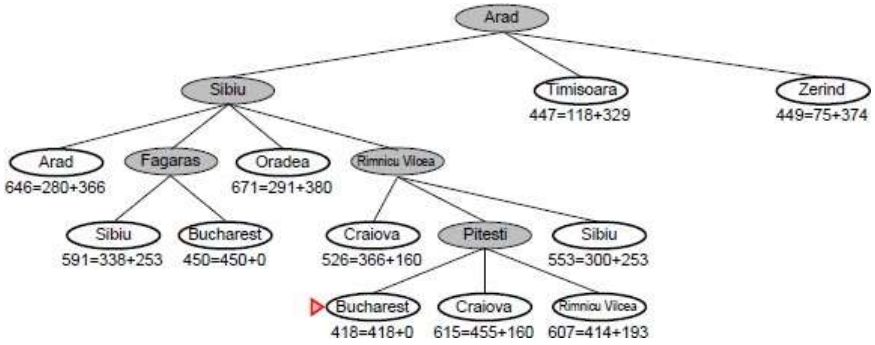
Fig: Greedy search

A* search

- A* combines the value of the heuristic function h(n) and the cost to reach the node n, g(n)
- Evaluation function f(n) = g(n) + h(n) thus estimates the cost of the cheapest solution through n
- A* tries the node with the lowest f(n) value first
- This leads to both complete and optimal search algorithm, provided that h(n) satisfies certain conditions

Example: Fig1

Solution:



Optimality of A*

- Provided that h(n) never overestimates the cost to reach the goal, then in tree search A* gives the optimal solution
- Suppose G2 is a suboptimal goal node generated to the tree
- Let C* be the cost of the optimal solution
- Because G2 is a goal node, it holds that h(G2) = 0,and we know that f(G2) = g(G2) > C*
- On the other hand, if a solution exists, there must exist a node n that is on the optimal solution path in the tree
- Because h(n) does not overestimate the cost of completing the solution path, f(n) = g(n) + h(n) C*
- We have shown that f(n) C* < f(G2),so G2 will not be expanded and A* must return an optimal solution

Memory-bounded heuristic search

- Once again the main drawback of search is not computation time, but rather space consumption.
- Therefore, one has had to develop several memory-bounded variants of A*.
- IDA* (Iterative Deepening A*) adapts the idea of iterative deepening.
- The cutoff used in this context is the f-cost (g + h) rather than the depth.
- At each iteration the cutoff value is the smallest f-cost of any node that exceeded the cutoff on the previous iteration.
- Subsequent more modern algorithms carry out more complex pruning.

Local searches: This search is used in many problems, where, the path to the goal is irrelevant.

- Local search algorithms operate using a single current state and generally move to neighbors of that state.
- Typically, the paths followed by the search are not retained.
- They use very little memory, usually a constant amount.
- Local search algorithms lead to reasonable results in large or infinite

(continuous) state spaces for which systematic search methods are unsuitable.

Local Search Algorithms and Optimization Problems

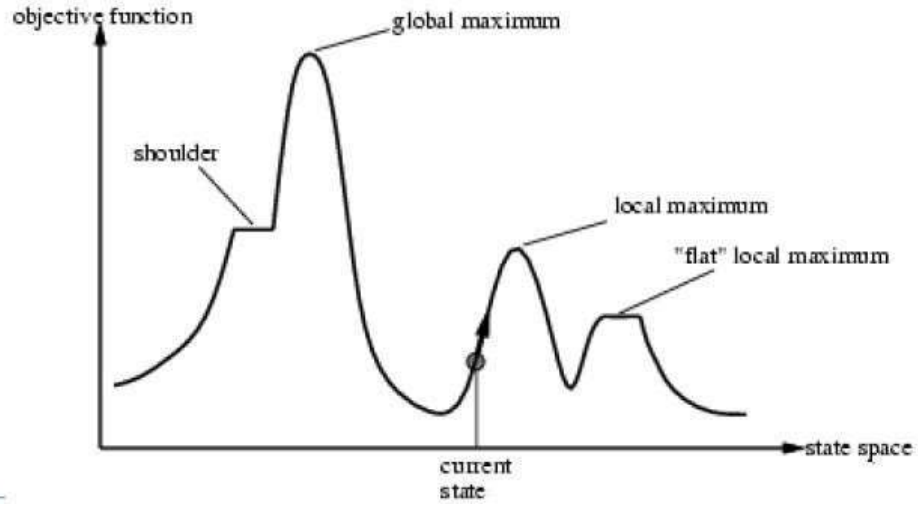
- Local search algorithms are also useful for solving pure optimization problems. • In optimization the aim is to find the best state according to an objective function.
- Optimization problems are not always search problems in the same sense as they were considered above.
- For instance, Darwinian evolution tries to optimize reproductive fitness.
- There does not exist any final goal state (goal test).
- Neither does the cost of the path matter in this task.
- Optimization of the value of objective function can be visualized as a state space landscape, where the height of peaks and depth of valleys corresponds to the value of the function
- A search algorithm giving the optimal solution to a maximization problem comes up with the global maximum
- Local maxima are higher peaks than any of their neighbors, but lower than the global maxima.

Hill climbing search

- In hill-climbing one always chooses a successor of the current state s that has the highest value for the objective function f: maxs f(s')
- Search terminates when all neighbors of the state have a lower value for the objective function than the current state has
- Most often search terminates in a local maximum, sometimes by chance, in a global maximum
- Also plateaux cause problems to this greedy local search
- On the other hand, improvement starting from the initial state is often very fast • Sideways moves can be allowed when search may proceed to states that are as good as the current one.
- Stochastic hill-climbing chooses at random one of the neighbors that improve the situation.

- Neighbors can, for example, be examined in random order and choose the first one that is better than the current state.
- Also these versions of hill-climbing are incomplete because they can still get stuck in a local maximum.
- By using random restarts one can guarantee the completeness of the method.
- Hill-climbing starts from a random initial state until a solution is found.

Example:



Constraint satisfaction problems

Formally speaking, a constraint satisfaction problem (or CSP) is defined by a set of variables, X1, X2, . . . , Xn, and a set of constraints, C1, C2, . . . , Cm. Each variable Xi has a nonempty domain Di of possible values. Each constraint Ci involves some subset of the variables and specifies the allowable combinations of values for that subset. A state of the problem is defined by an assignment of values to some or all of the variables, {Xi = vi , Xj = vj , . . .}. An assignment that does not violate any constraints is called a consistent or legal assignment. A complete assignment is one in which every variable is mentioned, and a solution to a CSP is a complete assignment that satisfies all the constraints. Some CSPs also require a solution that maximizes an objective function.

So what does all this mean? Suppose that, having tired of Romania, we are looking at a map of Australia showing each of its states and territories, as in Figure 5.1(a), and that we are given the task of coloring each region either red, green, or blue in such a way that no neighboring regions have the same color. To formulate this as a CSP, we define the variables to be the regions: WA, NT, Q, NSW, V, SA, and T.

The domain of each variable is the set {red, green, blue}.

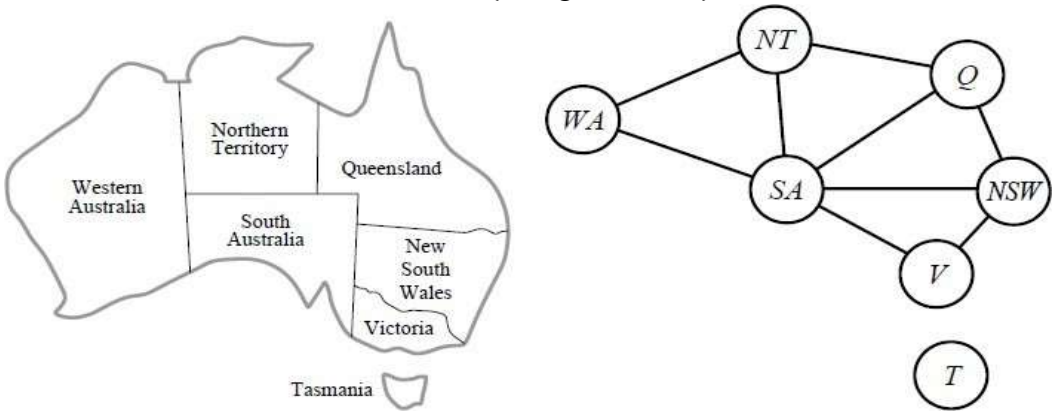


Fig: (a) The principal states and territories of Australia. Coloring this map can be viewed as a constraint satisfaction problem. The goal is to assign colors to each region so that no neighboring regions have the same color. (b) The map-coloring problem represented as a constraint graph.

The constraints require neighboring regions to have distinct colors; for example, the allowable combinations for WA and NT are the pairs {(red, green),(red, blue),(green, red),(green, blue),(blue, red),(blue, green)}. (The constraint can also be represented more succinctly as the inequality WA 6= NT, provided the constraint satisfaction algorithm has some way to evaluate such expressions.) There are many possible solutions, such as {WA = red, NT = green, Q = red, NSW = green, V = red, SA = blue, T = red }.

It is helpful to visualize a CSP as a constraint graph, as shown in Figure 5.1(b). The nodes of the graph correspond to variables of the problem and the arcs correspond to constraints. Treating a problem as a CSP confers several important benefits. Because the representation of states in a CSP conforms to a standard pattern—that is, a set of variables with assigned values—the successor function and goal test can written in a generic way that applies to all CSPs. Furthermore, we can develop effective, generic heuristics that require no additional, domain- specific expertise. Finally, the structure of the constraint graph can be used to simplify the solution process, in some cases giving an exponential reduction in complexity.

It is easy to see that a CSP can be given an incremental formulation as a standard search problem as follows:

I. Initial state: the empty assignment {}, in which all variables are unassigned.

II. Successor function: a value can be assigned to any unassigned variable, provided that it does not conflict with previously assigned variables.

III. Goal test: the current assignment is complete.

IV. Path cost: a constant cost for every step

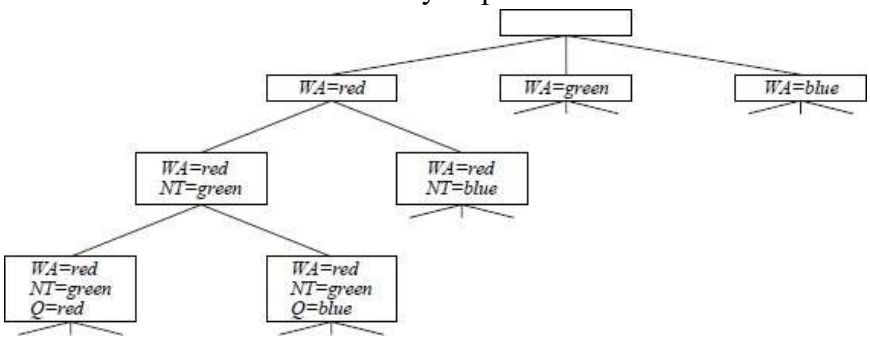


Fig: Part of the search tree generated by simple backtracking for the map-coloring problem

Summary

- Blind search: Depth-First, Breadth-First, IDS – Do not use knowledge of problem space to find solution.
- Informed search
- Best-first search: Order agenda based on some measure of how 'good' each state is.
- Uniform-cost: Cost of getting to current state from initial state = g(n)
- Greedy search: Estimated cost of reaching goal from current state – Heuristic evaluation functions, h(n) • A* search: f(n) = g(n) + h(n)
- Admissibility: h(n)never overestimates the actual cost of getting to the goal state.
- Informedness: A search strategy which searches less of the states pace in order to find a goal state is more informed.
- Constraint satisfaction problems (or CSPs) consist of variables with constraints on them. Many important real-world problems can be described as CSPs. The structure of a CSP can be represented by its constraint graph.
- Backtracking search, a form of depth-firsts earch, is commonly used for solving CSPs.

- The minimum remaining values and degree heuristics are domain-independent methods for deciding which variable to choose next in a backtracking search. The least constraining-value heuristic helps in ordering the variable values.
- By propagating the consequences of the partial assignments that it constructs, the backtracking algorithm can reduce greatly the branching factor of the problem. Forward checking is the simplest method for doing this. Arc consistency enforcement is a more powerful technique, but can be more expensive to run.
- Backtracking occurs when no legal assignment can be found for a variable. Conflict directed back jumping backtracks directly to the source of the problem.
- Local search using the min-conflicts heuristic has been applied to constraint satisfaction problems with great success.
- The complexity of solving a CSP is strongly related to the structure of its constraint graph. Tree-structured problems can be solved in linear time. Cut set conditioning can reduce a general CSP to a tree-structured one and is very efficient if a small cut set can be found. Tree decomposition techniques transform the CSP into a tree of sub problems and are efficient if the tree width of the constraint graph is small.

Stochastic search

Stochastic search is a general approach for solving combinatorial problems with significant research and practical potential.

Stochastic Local Search:

- randomize initialization step.
- random initial solutions.
- randomized construction heuristics.

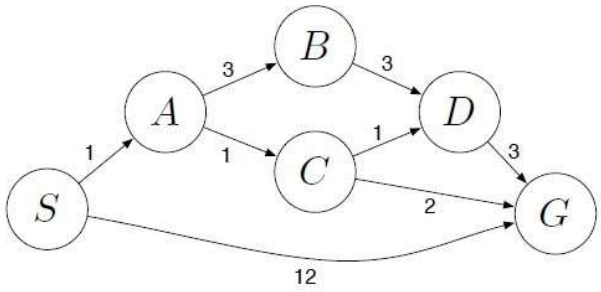
- Simulated annealing is a popular stochastic algorithm designed in analogy with the physical process of cooling a molten substance where condensing of matter into a crystalline solid takes place. In this context, searching for an optimal solution is like finding a configuration of the cooled system with minimum free energy. Because of its ability of escaping from local optima, simulated annealing is a powerful algorithm for numerical and combinatorial optimization. And another one is Genetic and evolutionary methods.

This search is easy to implement but sometimes it may be incomplete.

For instance, sophisticated search techniques form the backbone of modern machine learning and data analysis. Computer systems that are able to extract

information from huge data sets (data mining), to recognize patterns, to do classification, or to suggest diagnoses, in short, systems that are adaptive and — to some extent — able to learn, fundamentally rely on effective and efficient search techniques.

Workout Problem:



Answer the following questions about the search problem shown above. Break any ties alphabetically. For th questions that ask for a path, please give your answers in the form '$S - A - D - G$.'

(a) **(2 pt)** What path would breadth-first graph search return for this search problem?

$S - G$

(b) **(2 pt)** What path would uniform cost graph search return for this search problem?

$S - A - C - G$

(c) **(2 pt)** What path would depth-first graph search return for this search problem?

$S - A - B - D - G$

(d) **(2 pt)** What path would A* graph search, using a consistent heuristic, return for this search problem?

$S - A - C - G$

Module 3 Knowledge Representation and Reasoning

Adversarial search

Examine the problems that arise when we try to plan in a world where other agents are planning against us. A good example is in board games. Adversarial games, while much studied in AI, are a small part of game theory in economics.

Search versus Games

Search is not adversary

Solution is (heuristic) method for finding goal Heuristic techniques can find optimal solution

Evaluation function: estimate of cost from start to goal through given node Examples: path planning, scheduling activities

Games is adversary

Solution is strategy (strategy specifies move for every possible opponent reply). Optimality depends on opponent. Why? Time limits force an approximate solution Evaluation function: evaluate "goodness" of game position Examples: chess, checkers, Othello, backgammon

Mini-Max Algorithm

Mini-max is a type of backtracking algorithm. It is used in decision-making and game theory to find the optimal move for a player. It is widely used in two player turn based games such as Tic-Tac-Toe, Chess, etc

In Mini-Max algorithm, the two players are called Max and Min. The Max tries to get the highest score possible while the Min tries to get the lowest score possible.

Example:

Consider a game which has 4 final states and paths to reach final state are from root to 4 leaves of a perfect binary tree as shown in fig. Assume you are the max player and you get the first chance to move, i.e., you are at root, and your opponent at next level. Which

move you would make as a max player considering that your opponent also plays optimally?

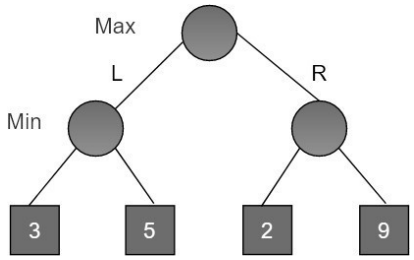


Fig. 1

Since this is a backtracking based algorithm, it tries all possible moves, then backtracks and makes a decision.

Solution:

Step1. Max goes left:

It is now the Min turn. The Min now has a choice between 3 and 5. Being the Min it will definitely choose the least among both, that is 3 Step2. Max goes right:

It is now the Min turn. The Min now has a choice between 2 and 9. He will choose 2 as it is the least among the two values.

Step3. Being the Max you would choose the larger value that is 3. Therefore, the optimal move for the Max is to go left and the optimal value is 3. Step4. If there are more levels in the game tree, the process is repeat again. The following fig shows the solution tree.

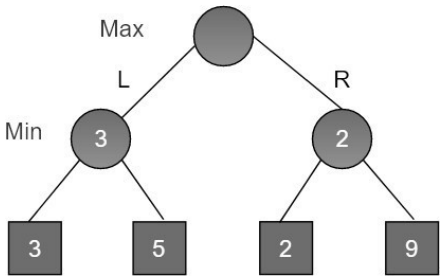


Fig.2

The disadvantage of the mini-max algorithm

The disadvantage of the mini-max algorithm is one time to find its children and a second time to evaluate the all-heuristic values, which increase the complexity.

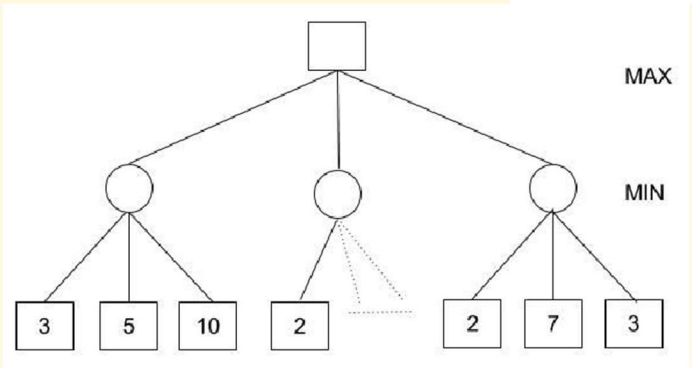
These disadvantages can be overcome by eliminating nodes from the tree without analyzing, and this process is called pruning.

Alpha-Beta pruning Algorithm

Alpha-Beta pruning algorithm reduces the computation time by a huge factor than Mini-Max. It cuts off (prune) branches in the game tree, which need not be searched because there already exists a better move available. It uses two parameters in the mini-max algorithm, namely alpha and beta that is why it is called Alpha-Beta pruning.

Alpha: It is the best choice so far for the player MAX. We want to get the highest possible value here.
Beta: It is the best choice so far for MIN, and it has to be the lowest possible value.

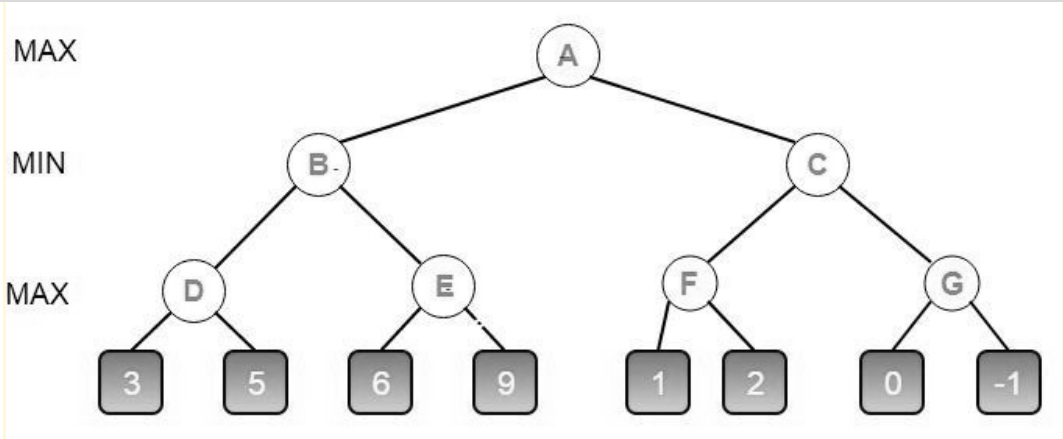
Let us understand the intuition behind this first and then we will formalize the algorithm. Suppose, we have the following game tree:



In this case,
Minimax Decision = MAX{MIN{3,5,10}, MIN{2,a,b}, MIN{2,7,3}}
= MAX{3,c,2}
= 3

You would be surprised! How could we calculate the maximum with a missing value? Here is the trick. MIN{2,a,b} would certainly be less than or equal to 2, i.e., c<=2 and hence MAX{3,c,2} has to be 3. The question now is do we really need to calculate c? Of course not. We could have reached to the conclusion without looking at those nodes. And this is where alpha-beta pruning comes into the picture.

Example

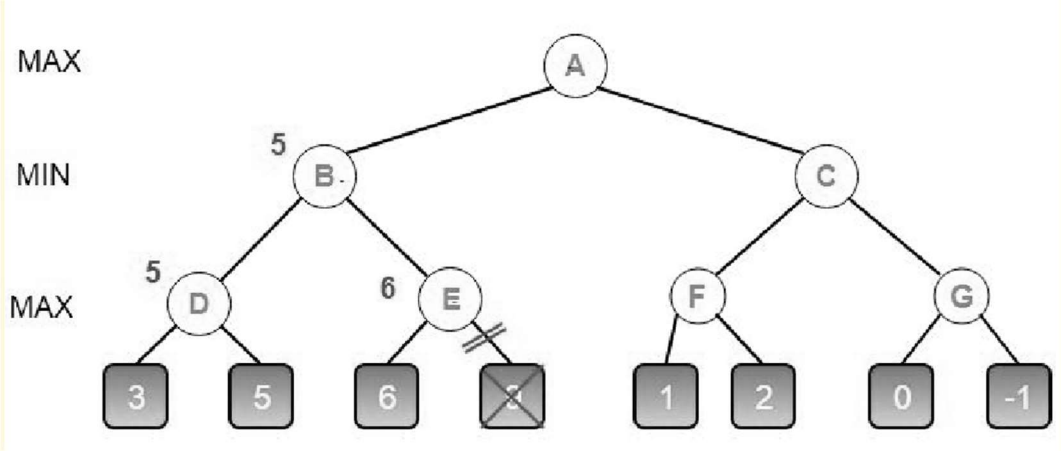


Solution

- The initial call starts from **A**. The value of alpha here is **-INFINITY** and the value of beta is **+INFINITY**. These values are passed down to subsequent nodes in the tree. At **A** the maximizer must choose max of **B** and **C**, so **A** calls **B** first ○ At **B** it the minimizer must choose min of **D** and **E** and hence calls **D** first.
- At **D**, it looks at its left child which is a leaf node. This node returns a value of 3. Now the value of alpha at **D** is max(-INF, 3) which is 3.
- To decide whether its worth looking at its right node or not, it checks the condition beta<=alpha. This is false since beta = +INF and alpha = 3. So it continues the search.
- **D** now looks at its right child which returns a value of 5.At **D**, alpha = max(3, 5) which is 5. Now the value of node **D** is 5 ○ **D** returns a value of 5 to **B**. At **B**, beta = min(+INF, 5) which is 5. The minimizer is now guaranteed a value of 5 or lesser. **B** now calls **E** to see if he can get a lower value than 5.
- At **E** the values of alpha and beta is not -INF and +INF but instead -INF and 5 respectively, because the value of beta was changed at **B** and that is what **B** passed down to **E**
- Now **E** looks at its left child which is 6. At **E**, alpha = max(-INF, 6) which is 6. Here the condition becomes true. beta is 5 and alpha is 6. So beta<=alpha is true. Hence it breaks and **E** returns 6 to **B**
- Note how it did not matter what the value of **E**'s right child is. It could have been +INF or -INF, it still wouldn't matter, We never even had to look at it because the minimizer was guaranteed a value of 5 or lesser. So as soon as the maximizer saw the 6 he knew the minimizer would never come this way because he can get a 5 on the left side of **B**. This way we dint have to look at that 9 and hence saved computation time.

- **E** returns a value of 6 to **B**. At **B**, beta = min(5, 6) which is 5.The value of node **B** is also 5

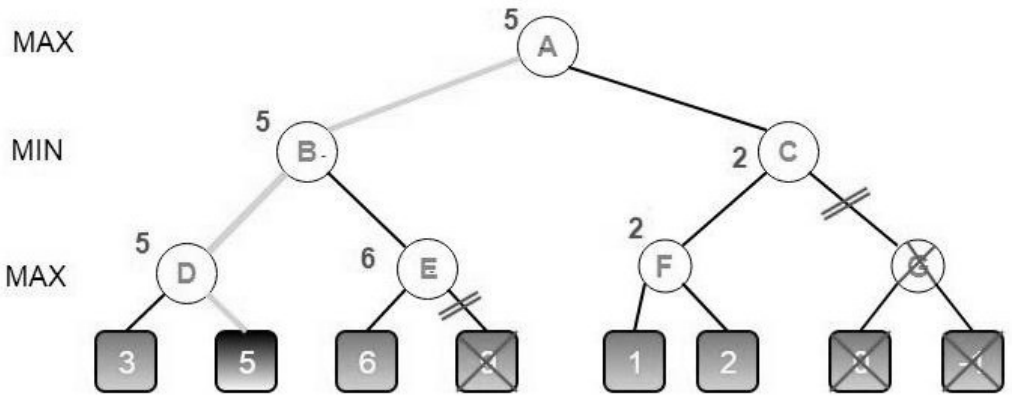
So far this is how our game tree looks. The 9 is crossed out because it was never computed.



- ✓ **B** returns 5 to **A**. At **A**, alpha = max(-INF, 5) which is 5. Now the maximizer is guaranteed a value of 5 or greater. **A** now calls **C** to see if it can get a higher value than 5.
- ✓ At **C**, alpha = 5 and beta = +INF. **C** calls **F**
- ✓ At **F**, alpha = 5 and beta = +INF. **F** looks at its left child which is a 1. alpha = max(5, 1) which is still 5. ✓ **F** looks at its right child which is a 2. Hence the best value of this node is 2.

  Alpha still remains 5 ✓ **F** returns a value of 2 to **C**. At **C**, beta = min(+INF, 2). The condition beta <= alpha becomes false as beta = 2 and alpha = 5. So it breaks and it dose not even have to compute the entire sub-tree of **G**.
- ✓ The intuition behind this break off is that, at **C** the minimizer was guaranteed a value of 2 or lesser. But the maximizer was already guaranteed a value of 5 if he choose **B**. So why would the maximizer ever choose **C** and get a value less than 2 ? Again you can see that it did not matter what those last 2 values were. We also saved a lot of computation by skipping a whole sub tree.
- ✓ **C** now returns a value of 2 to **A**. Therefore the best value at **A** is max(5, 2) which is a 5.
- ✓ Hence the optimal value that the maximizer can get is 5

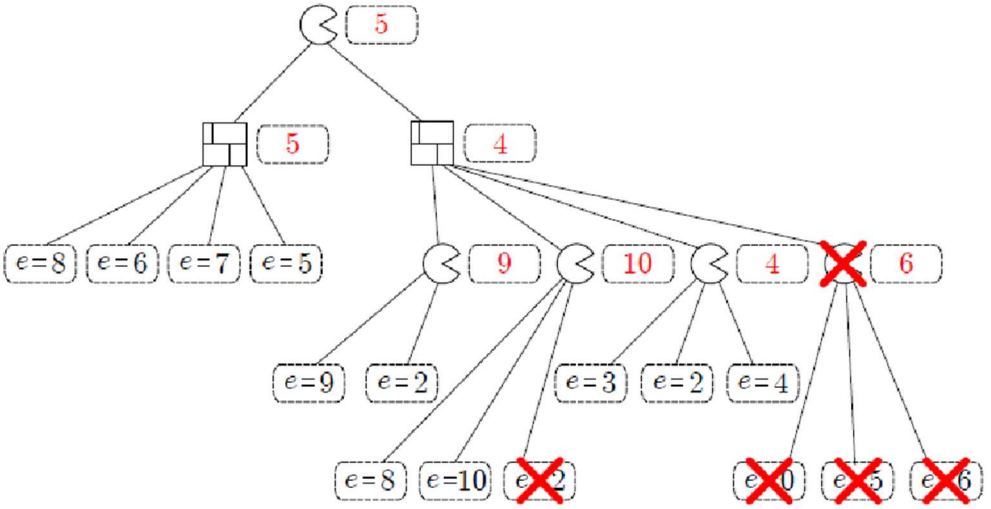
This is how our final game tree looks like. As you can see **G** has been crossed out as it was never computed.



Workout Problem

(3 pt) In the dashed boxes, fill in the values of all internal nodes using the minimax algorithm.

(3 pt) Cross off any nodes that are not evaluated when using alpha-beta pruning (assuming the standard left-to-right traversal of the tree).



(Filling values returned by alpha-beta or not crossing off children of a crossed off node were not penalized.)

Module 4 Learning

Knowledge

Knowledge is all kinds of facts about the world and it is necessary for intelligent behavior.

Knowledge Representation (KR)

Knowledge Representation is the area of Artificial Intelligence (AI) concerned with how knowledge can be represented symbolically and manipulated in an automated way by reasoning programs. KR is the field of study concerned with representations of propositions.

Reasoning

Reasoning is the process of deriving new proposition from KR.

Knowledge Acquisition (KA)

Knowledge Acquisition (KA) is the transfer of propositions from source to sink.

Knowledge base (KB)

A knowledge base (KB) is used to store complex structured and unstructured (symbolic and heuristic) information used by a computer system.
Symbolic: It incorporates knowledge that is symbolic as well as numeric.
Heuristic: It reasons with judgmental, imprecise, and qualitative knowledge as well as with formal knowledge of established theories.

Propositional Logic

Propositional logic is a formal language for representing knowledge and for making logical inferences. A proposition is a statement that is either true or false.

An atomic proposition is a statement that must be true or false. Examples of atomic propositions are "7is a prime" and "program p terminates".

A compound proposition can be created from other propositions using logical connectives. Truth-values of elementary propositions and the meaning of connectives define the truth of a compound proposition. The truth table for a compound proposition is a table with entries (rows) for all possible combinations of truth-values of elementary propositions.

Applications of propositional logic

- **Translation of English sentences**
- **Inference and reasoning:**
  - new true propositions are inferred from existing ones
  - Used in Artificial Intelligence:
    - Rule based (expert) systems
    - Automatic theorem provers
- **Design of logic circuit**

Inference

It is the process of deriving new proposition from the existing old ones.

Application: inference

The field of Artificial Intelligence:

- Builds programs that act intelligently
- Programs often rely on symbolic manipulations

Expert systems:

- Encode knowledge about the world in logic
- Support inferences where new facts are inferred from existing facts following the semantics of logic

Theorem provers:

- Encode existing knowledge (e.g. about math) using logic
- Show that some hypothesis is true

Logical connectives

Propositional logics are constructed from atomic propositions by using logical connectives:

Connectives	
0	false
1	true
$\neg$	not
$\wedge$	and
$\vee$	or
$\rightarrow$	conditional (implies)
$\Leftrightarrow$	biconditional (equivalent)

Well-formed formula (wff)

The well-formed formulas of propositional logic are obtained by using the construction rules below:

- An atomic proposition ϕ is a well-formed formula.
- If ϕ is a well-formed formula, then so is $\neg\phi$.
- If ϕ and ψ are well-formed formulas, then so are $\phi \wedge \psi$, $\phi \vee \psi$, $\phi \rightarrow \psi$, and $\phi \Leftrightarrow \psi$.
- If ϕ is a well-formed formula, then so is (ϕ).

Truth table

A truth table shows whether a propositional formula is true or false for each possible truth assignment. Truth functions are sometimes called Boolean functions.

If we know how the five basic logical connectives work, it is easy (in principle) to construct a truth table.

X	$\neg X$
0	1
1	0

X	Y	$X \wedge Y$	$X \vee Y$	$X \rightarrow Y$	$X \Leftrightarrow Y$
0	0	0	0	1	1
0	1	0	1	1	0
1	0	0	1	0	0
1	1	1	1	1	1

Tautology and Contradiction

A propositional formula S is

- a *tautology* if $S[\alpha] = 1$ for all α.
- a *contradiction* if $S[\alpha] = 0$ for all α.
- *satisfiable* if $S[\alpha] = 1$ for some α.

It is easy to see that $A \vee \neg A$ is a tautology and that $A \wedge \neg A$ is a contradiction

Implication

⋆ **Modus Ponens:** Given A and $A \Rightarrow B$, conclude B.

⋆ **Modus Tollens:** Given $A \Rightarrow B$ and $\neg B$, conclude $\neg A$.

Two formulae S and T are *equivalent* iff for any truth assignment α we have $S[\alpha] = T[\alpha]$.

Logical Equivalence

The propositions p and q are called logically equivalent if they have the same truth table.

Important logical equivalence

- **Identity**
  - $p \wedge T \iff p$
  - $p \vee F \iff p$
- **Domination**
  - $p \vee T \iff T$
  - $p \wedge F \iff F$
- **Idempotent**
  - $p \vee p \iff p$
  - $p \wedge p \iff p$
- **Double negation**
  - $\neg(\neg p) \iff p$
- **Commutative**
  - $p \vee q \iff q \vee p$
  - $p \wedge q \iff q \wedge p$
- **Associative**
  - $(p \vee q) \vee r \iff p \vee (q \vee r)$
  - $(p \wedge q) \wedge r \iff p \wedge (q \wedge r)$
- **Distributive**
  - $p \vee (q \wedge r) \iff (p \vee q) \wedge (p \vee r)$
  - $p \wedge (q \vee r) \iff (p \wedge q) \vee (p \wedge r)$
- **De Morgan**
  - $\neg(p \vee q) \iff \neg p \wedge \neg q$
  - $(p \wedge q) \iff \neg p \vee \neg q$
- **Other useful equivalences**
  - $p \vee \neg p \iff T$
  - $p \wedge \neg p \iff F$
  - **$p \rightarrow q \iff (\neg p \vee q)$**
- **Example:** **Show $(p \wedge q) \rightarrow p$ is a tautology.**
- **Proof:** (we must show $(p \wedge q) \rightarrow p \iff T$)

$(p \wedge q) \rightarrow p$	$\iff \neg(p \wedge q) \vee p$	Useful
	$\iff [\neg p \vee \neg q] \vee p$	DeMorgan
	$\iff [\neg q \vee \neg p] \vee p$	Commutative
	$\iff \neg q \vee [\neg p \vee p]$	Associative
	$\iff \neg q \vee [T]$	Useful
	$\iff T$	Domination

- **Example 2:** Show $(p \rightarrow q) <=> (\neg q \rightarrow \neg p)$

  Proof:

- $(p \rightarrow q) <=> (\neg q \rightarrow \neg p)$
- $<=> \neg(\neg q) \vee (\neg p)$ Useful
- $<=> q \vee (\neg p)$ Double negation
- $<=> \neg p \vee q$ Commutative
- $<=> p \rightarrow q$ Useful

Example 1

Assume a sentence:

If you are older than 13 or you are with your parents then you can attend a PG-13 movie.

Parse:

- **If** (you are older than 13 **or** you are with your parents) **then** (you can attend a PG-13 movie)

Atomic (elementary) propositions:

- A= you are older than 13
- B= you are with your parents
- C=you can attend a PG-13 movie

- **Translation:** $\mathbf{A \vee B \rightarrow C}$

Example 2

You can have free coffee if you are senior citizen and it is a Tuesday

a **b** **c**

Step 3 rewrite the sentence in propositional logic

$\mathbf{b \wedge c \rightarrow a}$

Example 3

- Assume two elementary statements:
  - **p: you drive over 65 mph ; q: you get a speeding ticket**
- **Translate each of these sentences to logic**
  - you do not drive over 65 mph. $(\neg p)$
  - you drive over 65 mph, but you don't get a speeding ticket. $(p \wedge \neg q)$
  - you will get a speeding ticket if you drive over 65 mph. $(p \rightarrow q)$
  - if you do not drive over 65 mph then you will not get a speeding ticket.$(\neg p \rightarrow \neg q)$
  - driving over 65 mph is sufficient for getting a speeding ticket. $(p \rightarrow q)$
  - you get a speeding ticket, but you do not drive over 65 mph. **$(q \wedge \neg p)$**

First Order Predicate Logic (FOPL)

In propositional logic, each possible atomic fact requires a separate unique propositional symbol.

Predicate logic includes a richer ontology such as objects (terms), properties (unary predicates on terms), relations (n-ary predicates on terms), and functions (mappings from terms to other terms).

FOPL allows more flexible and compact representation of knowledge than propositional logic using universal and existential quantifiers.

Syntax for First-Order Logic

Sentence → AtomicSentence
| Sentence Connective Sentence
| Quantifier Variable Sentence
| ¬Sentence
| (Sentence)

AtomicSentence → Predicate(Term, Term, ...)
| Term=Term

Term → Function(Term,Term,...)
| Constant
| Variable

Connective → ∨ | ∧ | ⇒ | ⇔

Quanitfier → ∃ | ∀

Constant → A | John | Car1

Variable → x | y | z |...

Predicate → Brother | Owns | ...

Function → father-of | plus | ...

- Objects are represented by **terms**:
  - **Constants**: Block1, John
  - **Function symbols**: father-of, successor, plus
    An *n*-ary function maps a tuple of *n* terms to another term: father-of(John), succesor(0), plus(plus(1,1),2)

- Terms are simply names for objects. Logical functions are not procedural as in programming languages. They do not need to be defined, and do not really return a value. Allows for the representation of an infinite number of terms.

- Propositions are represented by a **predicate** applied to a tuple of terms. A predicate represents a property of or relation between terms that can be true or false:
  Brother(John, Fred), Left-of(Square1, Square2)
  GreaterThan(plus(1,1), plus(0,1))

- In a given interpretation, an *n*-ary predicate can defined as a function from tuples of *n* terms to {True, False} or equivalently, a set tuples that satisfy the predicate:

  {<John, Fred>, <John, Tom>, <Bill, Roger>, ...}

Quantifiers

- Allows statements about entire collections of objects rather than having to enumerate the objects by name.

- Universal quantifier: $\forall x$
  Asserts that a sentence is true for all values of variable x

  $\forall x$ Loves(x, FOPC)
  $\forall x$ Whale(x) $\Rightarrow$ Mammal(x)
  $\forall x$ Grackles(x) $\Rightarrow$ Black(x)
  $\forall x$ ($\forall y$ Dog(y) $\Rightarrow$ Loves(x,y)) $\Rightarrow$ ($\forall z$ Cat(z) $\Rightarrow$ Hates(x,z))

- Existential quantifier: $\exists$
  Asserts that a sentence is true for at least one value of a variable x

  $\exists x$ Loves(x, FOPC)
  $\exists x$(Cat(x) $\wedge$ Color(x,Black) $\wedge$ Owns(Mary,x))
  $\exists x$($\forall y$ Dog(y) $\Rightarrow$ Loves(x,y)) $\wedge$ ($\forall z$ Cat(z) $\Rightarrow$ Hates(x,z))

- General Identities
  - $\forall x \neg P \Leftrightarrow \neg\exists x\, P$
  - $\neg\forall x\, P \Leftrightarrow \exists x \neg P$
  - $\forall x\, P \Leftrightarrow \neg\exists x \neg P$
  - $\exists x\, P \Leftrightarrow \neg\forall x \neg P$
  - $\forall x\, P(x) \wedge Q(x) \Leftrightarrow \forall x P(x) \wedge \forall x Q(x)$
  - $\exists x\, P(x) \vee Q(x) \Leftrightarrow \exists x P(x) \vee \exists x Q(x)$

Modus Ponen Rule:

Lemma 1: If α : $(\forall x)$ (P(x) $\rightarrow$ Q(x)) and β : P(c) are two formulae, then Q(c) is a logical consequence of α and β, where c is a constant.

Modus Tollen Rule:

Lemma 2: If α : $(\forall x)$ (P(x) $\rightarrow$ Q(x)) and β : ~ Q(c) are two formulae, then ~ P(c) is a logical consequence of α and β, where c is a constant.

Representing Facts in First-Order Logic

1. Lucy* is a professor
2. All professors are people.
3. John is the dean.
4. Deans are professors.
5. All professors consider the dean a friend or don't know him.
6. Everyone is a friend of someone.
7. People only criticize people that are not their friends.
8. Lucy criticized John .

Knowledge base:

- is-prof(lucy)
- $\forall$ x (is-prof(x) $\rightarrow$ is-person(x))
- is-dean(John)
- $\forall$ x (is-dean(x) $\rightarrow$ is-prof(x))
- $\forall$ x ($\forall$ y (is-prof(x) $\wedge$ is-dean(y) $\rightarrow$ is-friend-of(y,x) $\vee \neg$knows(x, y)))
- $\forall$ x ($\exists$ y (is-friend-of (y, x)))
- $\forall$ x ($\forall$ y (is-person(x) $\wedge$ is-person(y) $\wedge$ criticize (x,y) $\rightarrow \neg$is-friend-of (y,x)))
- criticize(lucy, John)

Knowledge base:

- P1(A)
- $\forall$ x (P1(x) $\rightarrow$ P3(x))
- P4(B)
- $\forall$ x (P4(x) $\rightarrow$ P1(x))
- $\forall$ x ($\forall$ y (P1(x) $\wedge$ P4(y) $\rightarrow$ P2(y,x) $\vee$ $\neg$P5(x, y)))
- $\forall$ x ($\exists$ y (P2(y, x)))
- $\forall$ x ($\forall$ y (P3 (x) $\wedge$ P3(y) $\wedge$ P6(x,y) $\rightarrow$ $\neg$P2(y,x)))
- P6(A, B)

Question: $\neg$P2(B ,A)?

1. All professors are people.
2. Deans are professors.
3. All professors consider the dean a friend or don't know him.
4. Everyone is a friend of someone.
5. People only criticize people that are not their friends.

General Knowledge

6. Lucy* is a professor
7. John is the dean.
8. Lucy criticized John.

Specific problem

9. Is John a friend of Lucy's?

Query

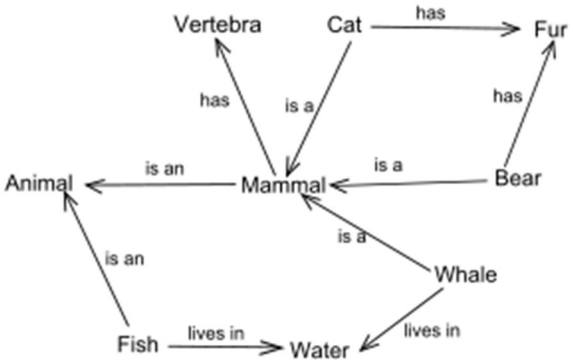
Semantic Net/Network

Semantic Network represents the knowledge in the form of graphical network. A semantic net is a knowledge representation technique used to represent propositional logic.

It represents knowledge about the world described in terms of graphs. Nodes correspond to the Concepts or objects in the domain and links to relations. Three kinds are subset links (isa, part-of links), member links (instance links) and function

links. It can be transformed to the first-order logic language. The graphical representation is often easier to work with better overall view on individual concepts and relations

Example



Frame

Frame is a collection of semantic network nodes or slots that together describe a stereotyped object, act or event. It representing knowledge about the objects and events typical to specific situations.

Example

Generic Restaurant Frame	
a-kind-of:	Business Establishment
Types:	Range: (Cafeteria, Seat-Yourself, Wait-to-be-seated, Fastfood) Default: IF plastic-orange-counter THEN fastfood IF stack-of-trays THEN cafeteria IF wait-for-waitress-sign OR reservation-made THEN waitto-be-seated OTHERWISE seat_yourself
Location:	Range: an ADDRESS if-needed: (Look at the menu)
Name:	if needed: (Look at the menu)

Food-style:	Range: (Burgers, Chinese, American, Seafood, French) Default: Chinese if-added: (Update Alternative of Restaurant)
Time-of-Operation:	Range: a time-of-day Default: open evenings except Mondays
Payment form:	Range: (Cash, CreditCard, Check, Washing-Dishes Script)
Event-Sequence:	Default: Eat-at-Restaurant Script
Alternatives::	Ranges: all restaurant with some foodstyle if-needed: (find all restaurants with the same foodstyle)

Script

A script is a structure that prescribes a set of circumstances, which could be expected to follow on from one another. It is similar to a thought sequence or a chain of situations, which could be anticipated. It could be considered to consist of a number of slots or frames but with more specialized roles.

The components of a script include:
Entry Conditions: These must be satisfied before events in the script can occur.
Results: Conditions that will be true after events in script occur.
Props: Slots representing objects involved in events.
Roles: Persons involved in the events.
Track: Variations on the script.
Scenes: The sequence of events that
occur. Example

- **Script Name: -Shopping Script**

Track:- Super Market
Roles: - Shopper, deli attendant , Check out clerk , sacking clerk , Cashier
Entry Condition: - Shopper needs groceries, food market open
Props: - Shopping Cart , Display aisles , market items ,checkout stands, money

SCENE1: - Enter Market
Shopper PTRANS Shopper into Market
Shopper PTRANS Shopping Cart to Shopper

SCENE2: - Shop For Item
Shopper MOVE Shopper through aisles
Shopper ATTEND eyes to display items
Shopper PTRANS item to Shopping Cart

SCENE3:- Check Out
Shopper MOVE Shopper to check out stand
Shopper ATTEND Eyes to charges
Shopper ATRANS money to cashier
Sacker ATRANS bag to shopper

Situation calculus

Situation calculus is Logic for reasoning about changes in the state of the world. The world is described by the sequences of situations of the current state. Changes from one situation to another are caused by actions. The situation calculus allows us to describe the initial state and a goal state. It builds the KB that describes the effect of actions (operators). It proves that the KB and the initial state lead to a goal state.

Basic Elements of Situation Calculus

a. ***Actions*-** that can be performed in the world The actions can be quantified.

Actions: move(x, y) implies robot is moving to a new location (x, y). pickup(o) implies robot picks up an object o. drop(o) implies robot drops the object o that holds.

b. ***Fluents*-** that describe the state of the world.

Relational fluents

Relational fluents are statements whose truth value may change and they take a situation as a final argument. is_carrying(o, s): robot is carrying object o in situation s.

Functional fluents

Functions that return a situation-dependent value and they take a situation as a final argument. location(s): returns the location (x, y) of the robot in situation s.

c. ***Situations-*** represent a history of action occurrences. A dynamic world is modeled as progressing through a series of situations as a result of various actions being performed within the world. It is a finite sequence of actions and a situation is not a state, but a history.

Situations:

Initial situation S0: no actions have yet occurred

A new situation, resulting from the performance of an action a in current situation s, is denoted using the function symbol do(a, s). do(move(2, 3), S0): denotes the new situation after the performance of action move(2, 3) in initial situation S0.

do(pickup(Ball), do(move(2, 3), S0)).

Automated reasoning systems

1) *Theorem proving:* Prove sentences in the first-order logic. Use inference rules, resolution rule and resolution refutation.
2) *Deductive retrieval systems*: Systems based on rules (KBs in Horn form) and prove theorems or infer new assertions (forward, backward chaining).
3) *Production systems:* Systems based on rules with actions in antecedents and it uses forward chaining mode of operation.
4) *Semantic networks:* Graphical representation of the world, objects are nodes in the graphs, relations are various links.

Theorem proving

Example 1. From the following facts, prove that *Ravi likes peanuts.*

a) Ravi likes all kind of food.
b) Apple and chicken are food.
c) Anything anyone eats and is not killed is food.
d) Ajay eats peanuts and still alive.

Solution:
Step1: Negate the statement to be proved.
Ravi likes peanuts. FOPL: likes(Ravi,peanuts) ¬likes(Ravi,peanuts)

Step2: Covert the facts into FOPL.

#	*Facts*	*FOPL*
a	Ravi likes all kind of food.	∀x: food(x) →likes(Ravi,x)
b	Apple and chicken are food.	food(Apple) food(Chicke n
c	Anything anyone eats and is not killed is food.	∀x∀ y: eats(x, y) ∧¬*killed(x)* →food(y)
d	Ajay eats peanuts and still alive	eats(Ajay, peanuts) ∧alive(Ajay)

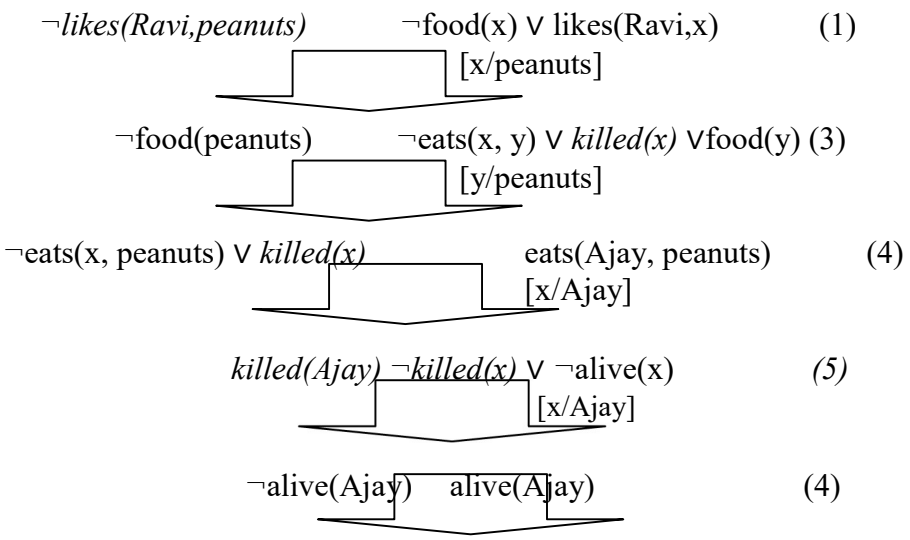
¬*killed(x)*
→alive(x) *killed(x)*
→¬alive(x)

Step3: Covert FOPL into CNF.

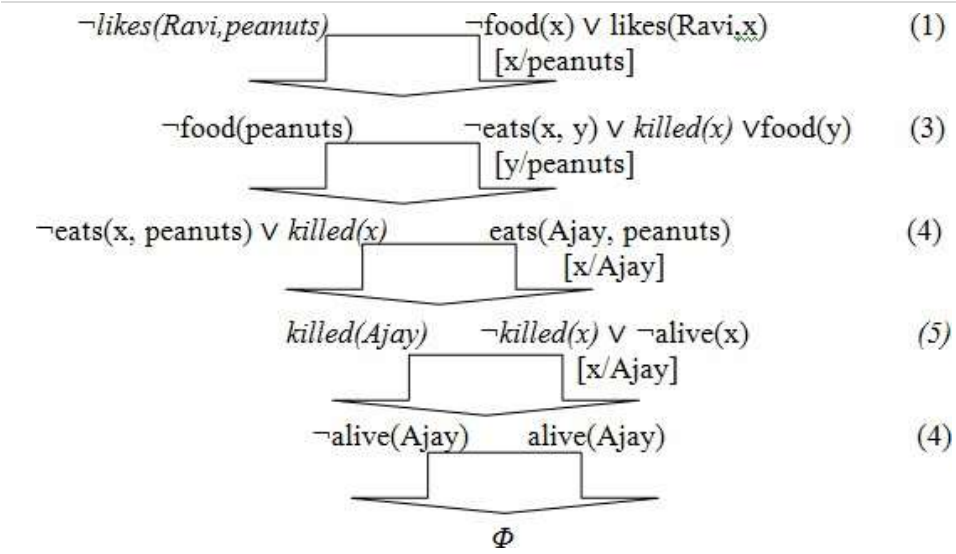
#	*FOPL*	*CNF*
1	∀x: food(x) →likes(Ravi,x)	¬food(x) ∨ likes(Ravi,x)

2	food(Apple) ∧food(Chicken	food(Apple) food(Chicke n
3	∀x∀y: eats(x, y) ∧¬*killed(x)* →food(y)	¬eats(x, y) ∨ *killed(x)* ∨food(y)
4	eats(Ajay, peanuts) ∧alive(Ajay)	eats(Ajay, peanuts) alive(Ajay)
5	*killed(x)* ∨ alive(x) ¬*killed(x)* ∨ ¬alive(x)	¬*killed(x)* →alive(x) *killed(x)* →¬alive(x)

Step4: Draw resolution graph.



Φ



That means the outcome of resolution graph is false. So our assumption ¬likes(Ravi,peanuts) is false implies likes(Ravi,peanuts) i.e, Ravi likes peanuts is proved.

Planning

Problem is difficult because: agent must consider action sequences starting from initial state - Before we can purchase anything, have to get to supermarket! but the agent (using a search technique) does not know this. *Planning requires explicit knowledge!* **Planning =** **find a sequence of actions that achieves a goal**	Planning Let us start by looking at an everyday problem that is too difficult to solve by search or inference techniques:

Different planners

- Once description available, planning can be done
- Progression vs. regression planners (state-space search)
- Partial order planning (incomplete planning)

Two approaches to search

- Forward state-space search - progression planning
  - Start with initial state, **choose actions (How?)**, goal test
- Backward state-space search - regression planning
  - Start with goal state, apply only **relevant** actions

Forward state space search - Progression Planners

- search forward from initial state
- determine which actions apply using preconditions
- use add/delete lists to compute new state
- main problem: often have huge search space because of large branching factor

Backward state-space search - Regression Planners

- search backwards from goal state to initial state
- note: we must deal with partial descriptions of the state since we do not yet know what actions will get us to goal
- in typical problems, goal state has a small number of conjuncts, each of which only has a few appropriate actions
- note: we have to be careful not to take actions that undo the desired literals
- searching backwards is complicated because we have to achieve a conjunction of goals (one alone is easy)

Partial-Order Planner: Definitions

Each plan has the following components:

- A plan is *complete* if every precondition of every step is achieved by some other step
  - A plan *achieves* a condition if the condition is one of the effects of the step
- A plan is *consistent* if there are no contradictions in the ordering or binding constraints

Formulation for POP

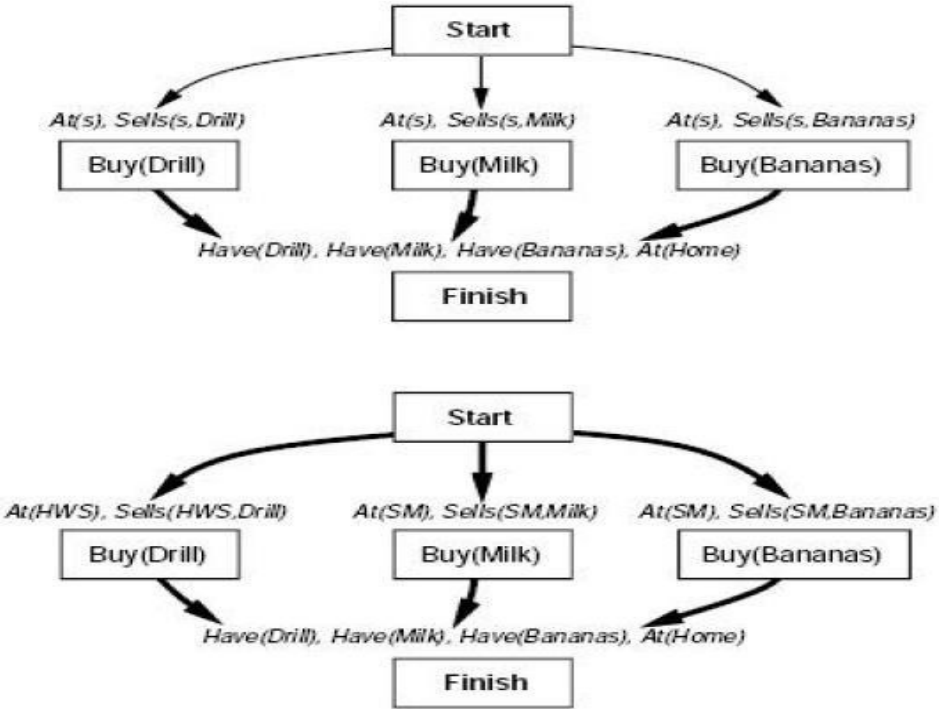
- The initial plan:
  - *Start* and *Finish*,
  - ordering constraint $Start \prec Finish$
  - no casual links
  - all preconditions in *Finish* are open preconditions
- Successor function arbitrarily picks one open precondition p on an action B and generates a successor plan for every possible consistent way of choosing an action A that achieves p

Partial-Order Planning

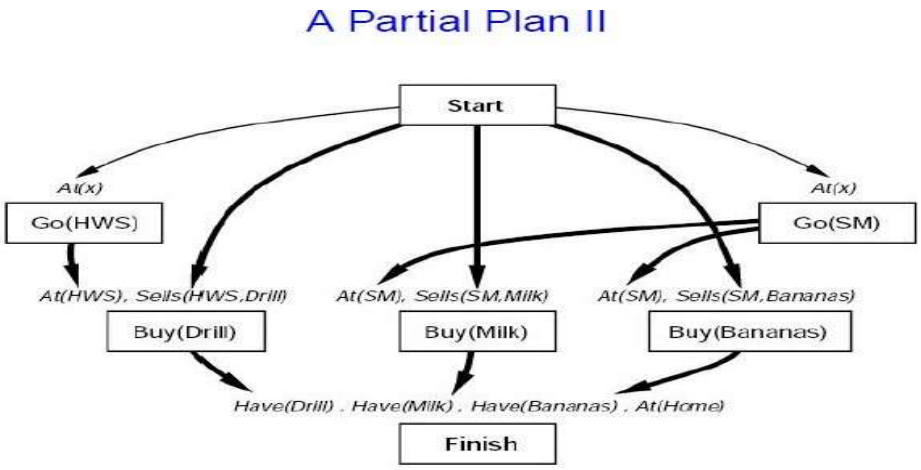
- search through plan space rather than situation space (state of the world)
- start with an initial plan consisting of the start and finish steps
- at each iteration, add one more step
- if this leads to an inconsistent plan, backtrack and try another branch of search space
- only add steps that achieve a precondition that has not yet been achieved

POP Example: Initial Plan

Example POS I:

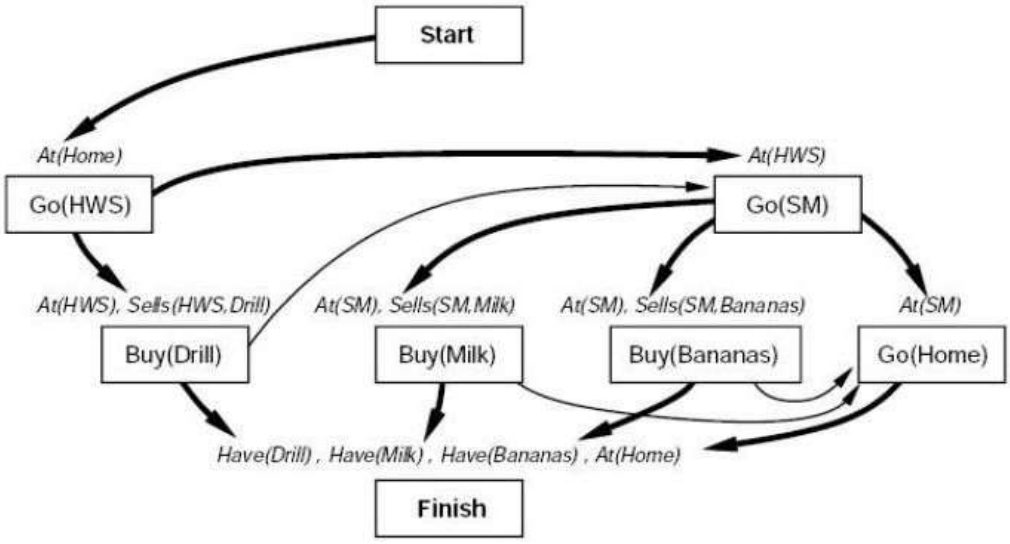


Example POS II:



Example POS III:

A Partial Plan III



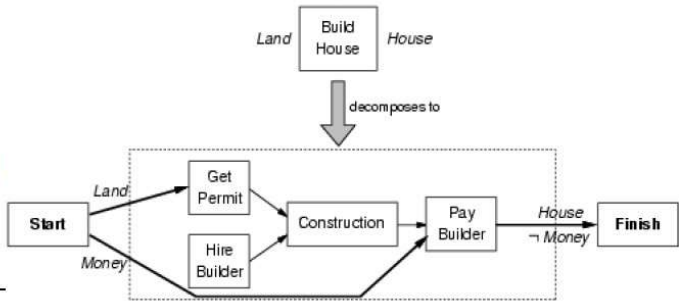
Planning and Scheduling

- **Planning**: given one or more goals, generate a sequence of actions to achieve the goal(s)
- **Scheduling**: given a set of actions and constraints, allocate resources and assign times to the actions so that no constraints are violated
- Traditionally, planning is done with specialized logical reasoning methods
- Traditionally, scheduling is done with constraint satisfaction, linear programming
- However, planning and scheduling are closely interrelated and can't always be separated

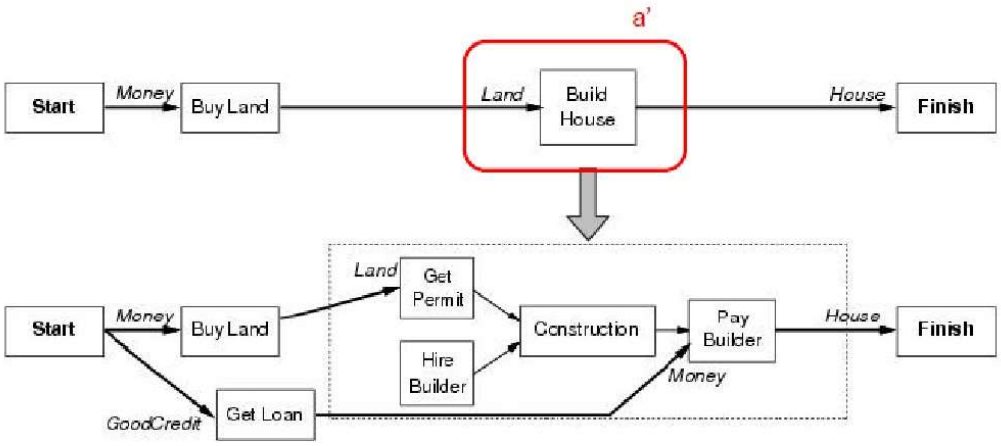
Hierarchical planning

- **Hierarchical decomposition**: express action at one level as a small set of actions at the next level
- **Hierarchical Task Network (HTN) Planning**: start with an abstract plan and continue expanding each activity until only **primitive action** exist

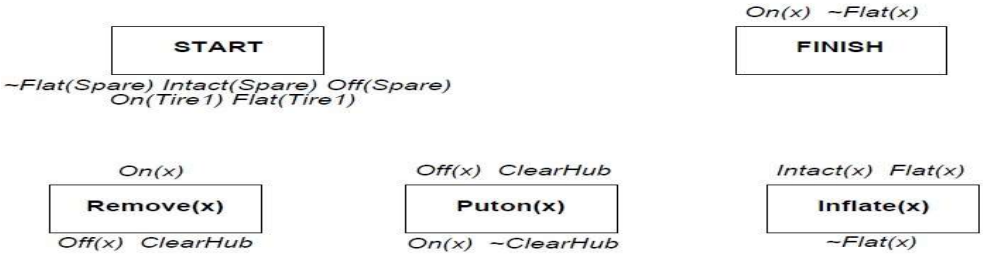
- HTN – Hierarchical Task Networks
  - √ Planner keeps library of subplans
  - √ Extend planning algorithm to use subplans
  - √ Can reduce time&space requirements considerably
- Most real-world planners use HTN variants



POP+HTN example



Planning and acting in nondeterministic domains



Incomplete information

Unknown preconditions, e.g., $Intact(Spare)$?
Disjunctive effects, e.g., $Inflate(x)$ causes
$Inflated(x) \vee SlowHiss(x) \vee Burst(x) \vee BrokenPump \vee \ldots$

Incorrect information

Current state incorrect, e.g., spare NOT intact
Missing/incorrect postconditions in operators

Qualification problem:
can never finish listing all the required preconditions and possible conditional outcomes of actions

Conformant or sensorless planning (env. with no observations)
Devise a plan that works regardless of state or outcome
Such plans may not exist

Conditional planning (partially observable and nondeterministic)
Plan to obtain information (observation actions)
Subplan for each contingency, e.g.,
$[Check(Tire1), \textbf{if}\ Intact(Tire1)\ \textbf{then}\ Inflate(Tire1)\ \textbf{else}\ CallAAA]$
Expensive because it plans for many unlikely cases

Monitoring/Replanning (unknown environments)
Check progress *during execution*, replan if necessary
Unanticipated outcomes may lead to failure (e.g., no AAA card)

(Really need a combination; plan for likely/serious eventualities,
deal with others when they arise, as they must eventually)

Probabilty

(Frequentist) An event's **probability** is the proportion of times that we expect the event to occur, if the experiment were repeated a large number of times.

(Subjectivist) A subjective **probability** is an individual's degree of belief in the occurrence of an event.

(Classical) An event's **probability** is the ratio of the number of favorable outcomes and possible outcomes in a (symmetric) experiment.

Term	*Description*	*Example*
Experiment	Phenomenon where outcomes are uncertain	Single throws of a six-sided die
Sample space	Set of all outcomes of the experiment	$S = \{1, 2, 3, 4, 5, 6\}$, $(1, 2, 3, 4, 5,$ *or* 6 dots show)
Event	A collection of outcomes; a subset of S	$A = \{3\}$ (3 dots show), $B = \{3, 4, 5,$ *or* $6\}$ (3, 4, 5, or 6 dots show) or 'at least three dots show'
Probability	A number between 0 and 1 assigned to an event.	$P(A) = \frac{1}{6}$. $P(B) = \frac{4}{6}$.

Property	*Notation*
If event S will *always* occur, its probability is 1.	$P(\mathcal{S}) = 1$
If event $\emptyset$ will *never* occur, its probability is 0.	$P(\emptyset) = 0$
Probabilities are always between 0 and 1, inclusive	$0 \leq P(A) \leq 1$
If $A, B, C, \ldots$ are all mutually exclusive then $P(A \cup B \cup C \ldots)$ can be found by addition.	$P(A \cup B \cup C \ldots) = P(A) + P(B) + P(C) + \ldots$
If A and B are mutually exclusive then $P(A \cup B)$ can be found by addition.	$P(A \cup B) = P(A) + P(B)$
Addition rule: The general *addition rule* for probabilities	$P(A \cup B) = P(A) + P(B) - P(A \cdot B)$
Since A and A^c are mutually exclusive and between them include all possible outcomes, $P(A \cup A^c)$ is 1.	$P(A \cup A^c) = P(A) + P(A^c) = P(\mathcal{S}) = 1$, and $P(A^c) = 1 - P(A)$

Rule	*Notation*
Definitions The *conditional probability* of A given B is the probability of event A, if event B occurred.	$P(A\|B)$
A is *independent* of B if the conditional probability of A given B is the same as the unconditional probability of A.	$P(A\|B) = P(A)$
Multiplication rule: The general *multiplication rule* for probabilities	$P(A \cdot B) = P(A)P(B\|A) = P(B)P(A\|B)$
For *independent events* only, the multiplication rule is simplified.	$P(A \cdot B) = P(A)P(B)$

The events $H_1, \ldots, H_n$ are usually called **hypotheses** and from their definition follows that $P(H_1) + \cdots + P(H_n) = 1\ (= P(\mathcal{S}))$.

Let the event of interest A happens under any of the hypotheses H_i with a known (conditional) probability $P(A|H_i)$. Assume, in addition, that the probabilities of hypotheses $H_1, \ldots, H_n$ are known. Then $P(A)$ can be calculated using the *total probability formula.*

Total Probability Formula.

$$P(A) = P(A|H_1)P(H_1) + \cdots + P(A|H_n)P(H_n).$$

Bayes Formula. Let the event of interest A happens under any of hypotheses H_i with a known (conditional) probability $P(A|H_i)$. Assume, in addition, that the probabilities of hypotheses $H_1, \ldots, H_n$ are known (*prior* probabilities). Then the conditional (*posterior*) probability of the hypothesis H_i, $i = 1, 2, \ldots, n$, given that event A happened, is

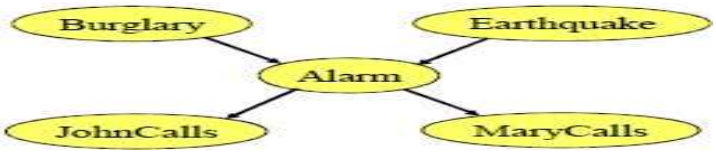
$$P(H_i|A) = \frac{P(A|H_i)P(H_i)}{P(A)},$$

where

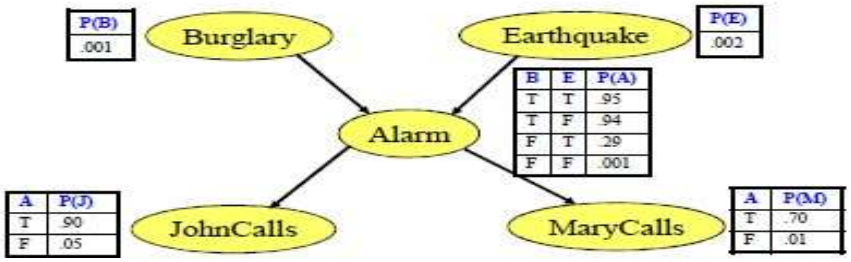
$$P(A) = P(A|H_1)P(H_1) + \cdots + P(A|H_n)P(H_n).$$

A **Bayesian network** is a graphical structure that allows us to represent and reason about an uncertain domain. The nodes in a Bayesian network represent a set of random variables, $\mathbf{X} = X_1, ..X_i, ...X_n$, from the domain. A set of directed **arcs** (or links) connects pairs of nodes, $X_i \rightarrow X_j$, representing the direct dependencies between variables. Assuming discrete variables, the strength of the relationship between variables is quantified by conditional probability distributions associated with each node. The only constraint on the arcs allowed in a BN is that there must not be any directed cycles: you cannot return to a node simply by following directed arcs. Such networks are called directed acyclic graphs, or simply **dags**.

- If no assumption of independence is made, then an exponential number of parameters must be estimated for sound probabilistic inference.
- No realistic amount of training data is sufficient to estimate so many parameters.
- If a blanket assumption of conditional independence is made, efficient training and inference is possible, but such a strong assumption is rarely warranted.
- **Graphical models** use directed or undirected graphs over a set of random variables to explicitly specify variable dependencies and allow for less restrictive independence assumptions while limiting the number of parameters that must be estimated.
  - **Bayesian Networks**: Directed acyclic graphs that indicate causal structure.
  - **Markov Networks**: Undirected graphs that capture general dependencies.
- Directed Acyclic Graph (DAG)
  - Nodes are random variables
  - Edges indicate causal influences



- Each node has a **conditional probability table** (**CPT**) that gives the probability of each of its values given every possible combination of values for its parents (conditioning case).
  - Roots (sources) of the DAG that have no parents are given prior probabilities.



CPT Comments

- Probability of false not given since rows must add to 1.
- Example requires 10 parameters rather than $2^5 - 1 = 31$ for specifying the full joint distribution.
- Number of parameters in the CPT for a node is exponential in the number of parents (fan-in).

Joint Distributions for Bayes Nets

- A Bayesian Network implicitly defines a joint distribution.

$$P(x_1, x_2, \ldots x_n) = \prod_{i=1}^{n} P(x_i \mid \text{Parents}(X_i))$$

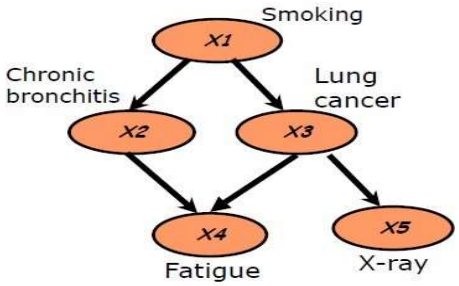
- Example

$$P(J \wedge M \wedge A \wedge \neg B \wedge \neg E)$$
$$= P(J \mid A)P(M \mid A)P(A \mid \neg B \wedge \neg E)P(\neg B)P(\neg E)$$
$$= 0.9 \times 0.7 \times 0.001 \times 0.999 \times 0.998 = 0.00062$$

- Therefore an inefficient approach to inference is:
  - 1) Compute the joint distribution using this equation.
  - 2) Compute any desired conditional probability using the joint distribution.

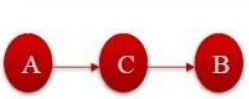
Bayesian Networks (BNs)

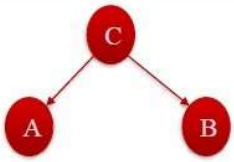
- Directed Acyclic Graphs (DAGs) for joint probability distributions
  - Nodes - random variables
  - Edges - direct dependence

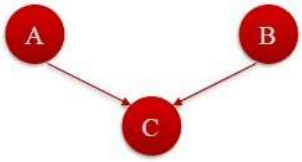


$$P(X1, X2, X3, X4, X5)$$
$$= P(X1)P(X2 \mid X1)P(X3 \mid X1)P(X4 \mid X2, X3)P(X5 \mid X3)$$

Independent Assumptions

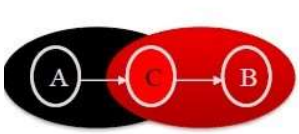


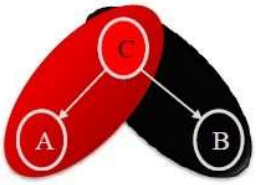




A and B are *marginally* dependent

A and B are *marginally* independent

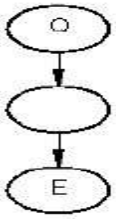




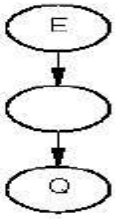
A and B are *conditionally* independent

A and B are *conditionally* dependent

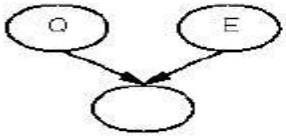
Types of Inference



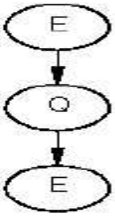
Diagnostic



Causal



(Explaining Away)
Intercausal



Mixed

Sample Inferences

- **Diagnostic (evidential, abductive)**: From effect to cause.
  - P(Burglary | JohnCalls) = 0.016
  - P(Burglary | JohnCalls ∧ MaryCalls) = 0.29
  - P(Alarm | JohnCalls ∧ MaryCalls) = 0.76
  - P(Earthquake | JohnCalls ∧ MaryCalls) = 0.18
- **Causal (predictive)**: From cause to effect
  - P(JohnCalls | Burglary) = 0.86
  - P(MaryCalls | Burglary) = 0.67
- **Intercausal (explaining away)**: Between causes of a common effect.
  - P(Burglary | Alarm) = 0.376
  - P(Burglary | Alarm ∧ Earthquake) = 0.003
- **Mixed**: Two or more of the above combined
  - (diagnostic and causal) P(Alarm | JohnCalls ∧ ¬Earthquake) = 0.03
  - (diagnostic and intercausal) P(Burglary | JohnCalls ∧ ¬Earthquake) = 0.017

Reasoning

Bayesian networks provide full representations of probability distributions over their variables. That implies that they can be conditioned upon any subset of their variables, supporting any direction of reasoning.

For example, one can perform **diagnostic reasoning**, i.e., reasoning from symptoms to cause, such as when a doctor observes *Dyspnoea* and then updates his belief about *Cancer* and whether the patient is a *Smoker*. Note that this reasoning occurs in the *opposite* direction to the network arcs.

Or again, one can perform **predictive reasoning**, reasoning from new information about causes to new beliefs about effects, following the directions of the network arcs. For example, the patient may tell his physician that he is a smoker; even before any symptoms have been assessed, the physician knows this will increase the chances of the patient having cancer. It will also change the physician's expectations that the patient will exhibit other symptoms, such as shortness of breath or having a positive X-ray result.

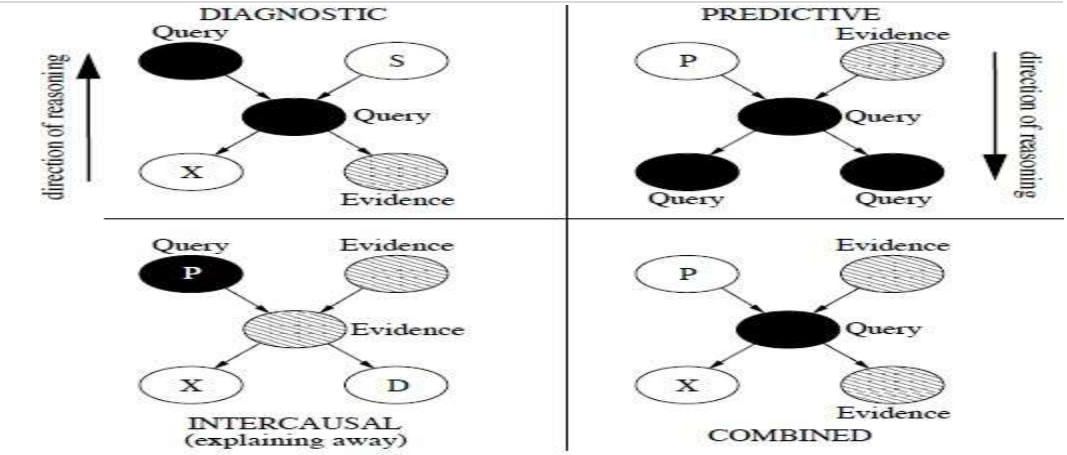


FIGURE 2.2: Types of reasoning.

explaining away is of some interest. Suppose that there are exactly two possible causes of a particular effect, represented by a **v-structure** in the BN. This situation occurs in our model of Figure 2.1 with the causes *Smoker* and *Pollution* which have a common effect, *Cancer* (of course, reality is more complex than our example!). Initially, according to the model, these two causes are independent of each other; that is, a patient smoking (or not) does not change the probability of the patient being subject to pollution. Suppose, however, that we learn that Mr. Smith has cancer. This will raise our probability for both possible causes of cancer, increasing the chances both that he is a smoker and that he has been exposed to pollution. Suppose then that we discover that he is a smoker. This new information explains the observed cancer, which in turn *lowers* the probability that he has been exposed to high levels of pollution. So, even though the two causes are initially independent, with knowledge of the effect the presence of one explanatory cause renders an alternative cause less likely. In other words, the alternative cause has been *explained away*.

Since any nodes may be query nodes and any may be evidence nodes, sometimes the reasoning does not fit neatly into one of the types described above. Indeed, we can combine the above types of reasoning in any way. Figure 2.2 shows the different varieties of reasoning using the Cancer BN. Note that the last combination shows the simultaneous use of diagnostic and predictive reasoning.

Example

Example statement: *You have a new burglar alarm installed. It reliably detects burglary, but also responds to minor earthquakes. Two neighbors, John and Mary, promise to call the police when they hear the alarm. John always calls when he hears the alarm, but sometimes confuses the alarm with the phone ringing and calls then also. On the other hand, Mary likes loud music and sometimes doesn't hear the alarm. Given evidence about who has and hasn't called, you'd like to estimate the probability of a burglary (from Pearl (1988)).*

A BN representation of this example is shown in Figure 2.6. All the nodes in this BN are Boolean, representing the true/false alternatives for the corresponding propositions. This BN models the assumptions that John and Mary do not perceive a burglary directly and they do not feel minor earthquakes. There is no explicit representation of loud music preventing Mary from hearing the alarm, nor of John's confusion of alarms and telephones; this information is summarized in the probabilities in the arcs from *Alarm* to *JohnCalls* and *MaryCalls*.

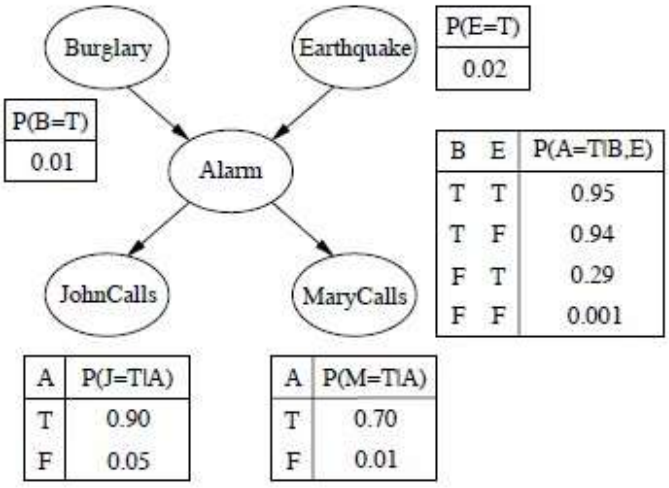


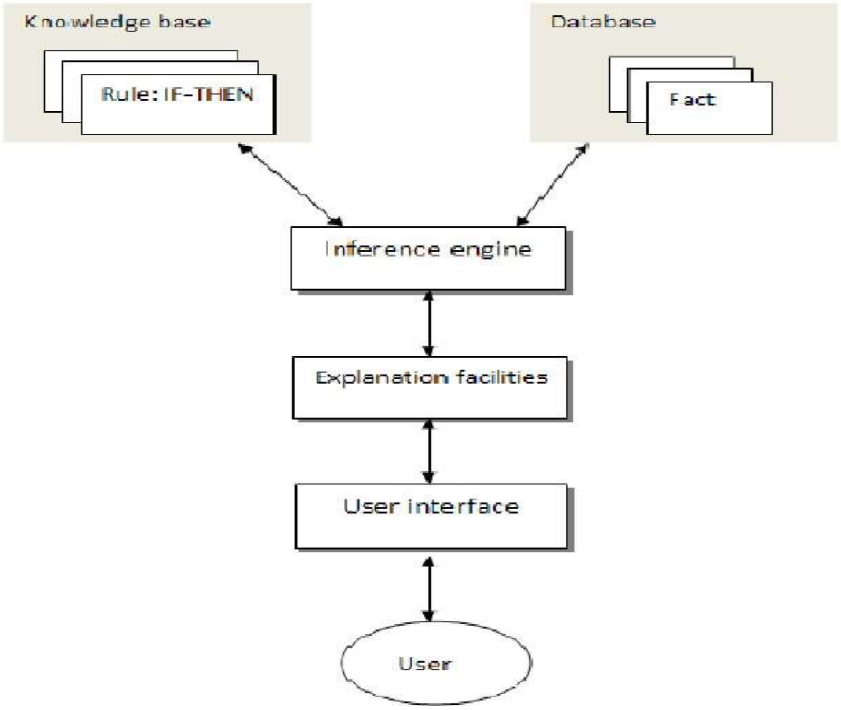
FIGURE 2.6: Pearl's Earthquake BN.

Architecture of a rule based system

A typical rule-based system consists of the following components:

- **knowledge base**: contains the rules embodying expert knowledge about the problem domain
- **database**: contains the set of known facts about the problem currently being solved
- **inference engine**: carries out the reasoning process by linking the rules with the known facts to find a solution
- **explanation facilities**: provides information to user about the reasoning steps that are being followed
- **user interface**: communication between the user and the system

Architecture of a rule-based system



Knowledge engineering

"....an engineering discipline that involves integrating knowledge into computer systems in order to solve complex problems normally requiring a high level of human expertise." - Feigenbaum and McCorduck

- Knowlege engineering is the process used to create a RBS:

  Assessing the problem and **selecting** an appropriate task for the RBS

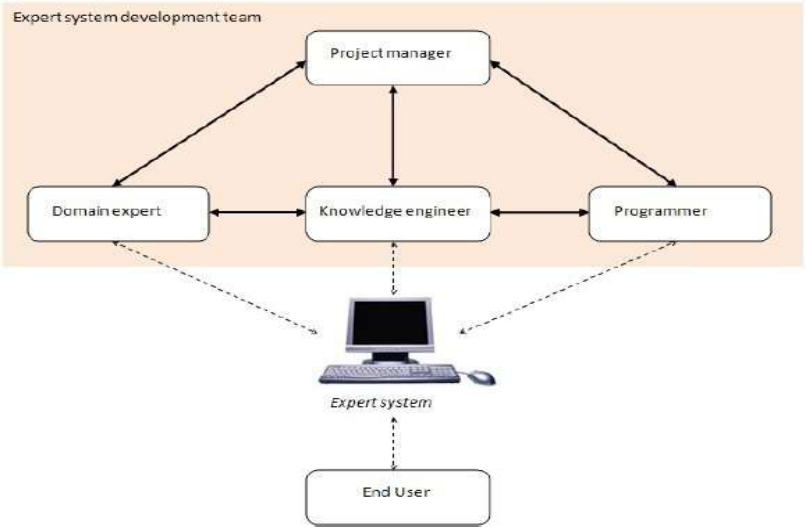
  Interviewing the domain expert to find out how the problem is solved

  Representing the domain knowledge as facts or rules

  Choosing appropriate development software such as a RBS "shell" and **encoding** the knowledge within that software

  Testing, **revising**, and **integrating** the system in the workplace

Knowlege engineering team structure



Dempster-Shafer Theory

- **Designed to Deal with the distinction between uncertainty and ignorance**
  - Rather than computing the probability of a proposition it computes the probability that evidence **supports** the proposition
- **Applicability of D-S**
  - assume lack of sufficient data to accurately estimate the prior and conditional probabilities to use Bayes rule
  - incomplete model ⇒ rather than estimating probabilities it uses belief intervals to estimate how close the evidence is to determining the truth of a hypothesis

Allows representation of *ignorance* about support provided by evidence

- allows reasoning system to be skeptical

For example, suppose we are informed that one of three terrorist groups, A, B or C has planted a bomb in a building.

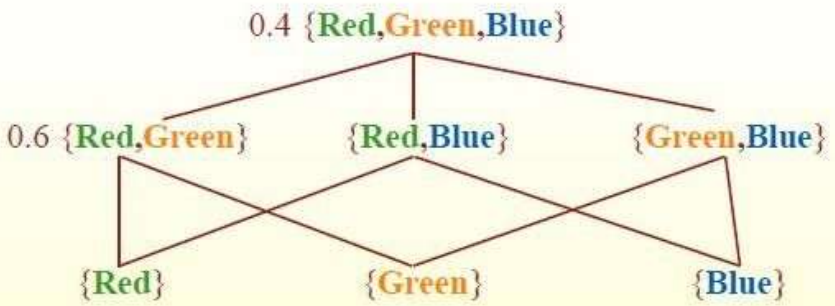
We may have some evidence the group C is guilty, P (C) =0.8

We would not want to say the probability of the other two groups being guilty is .1

In traditional theory, forced to regard belief and disbelief as functional opposites p(a) + p(not a) = 1 and to distribute an equal amount of the remaining probability to each group

D-S allows you to leave relative beliefs unspecified

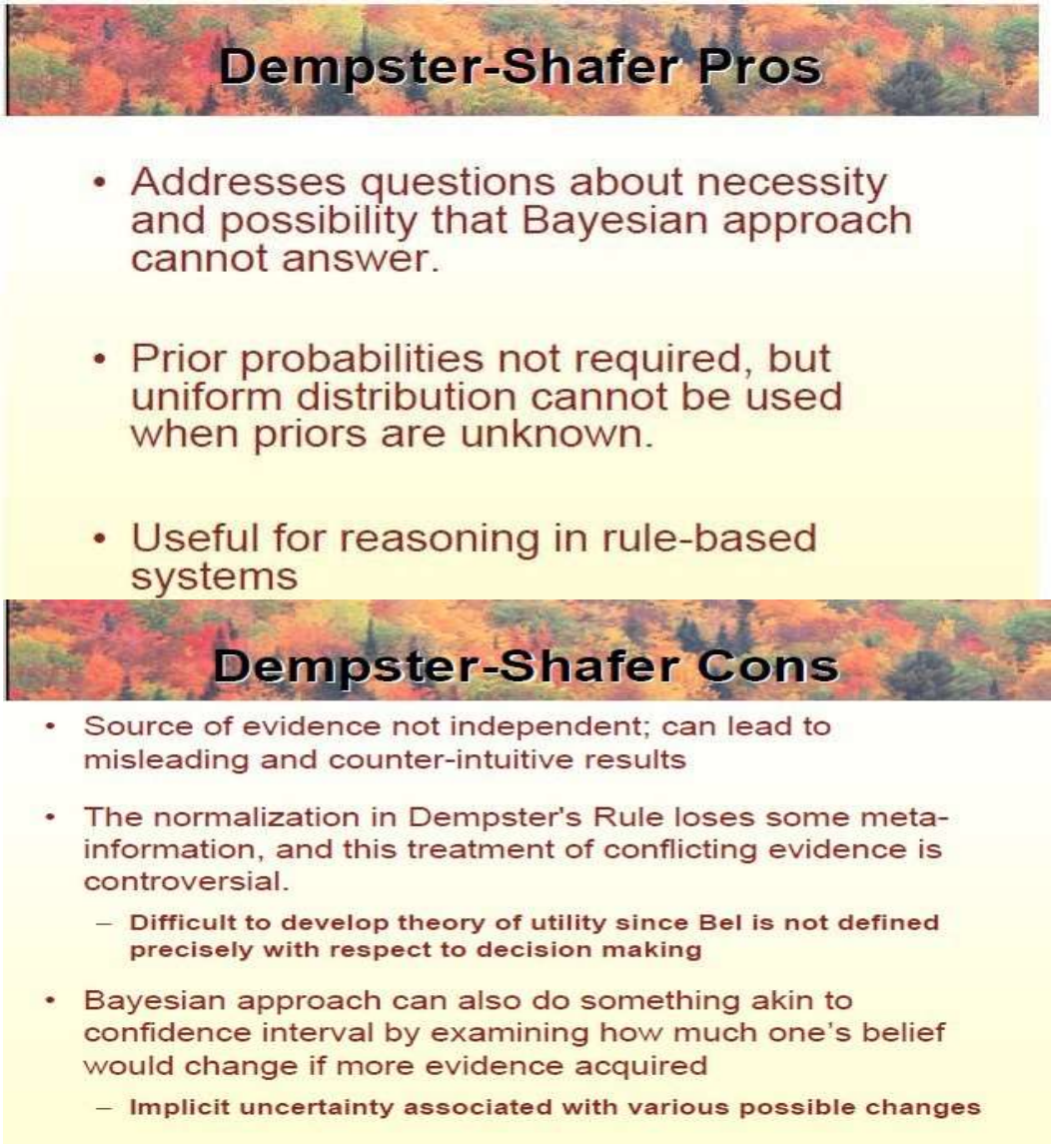
Belief Subsets



Suppose that the evidence supports {red,green} to the degree .6. The remaining support will be assigned to {red,green,blue} while a Bayesian model assumes that the remaining support is assigned to the negation of the hypothesis (or its complement) {blue}.

- **Given a population F =(blue,red, green) of mutually exclusive elements, exactly one of which (f) is true, a basic probability assignment (m) assigns a number in [0,1] to every subset of F such that the sum of the numbers is 1.**
  - Mass as a representation of evidence support
- **There are $2^{|F|}$ propositions, corresponding to "the true value of f is in subset A".**
  - (blue),(red),(green),(blue, red), (blue,green),(red,green),(red,blue,green),(empty set)
- **A belief in a subset entails belief in subsets containing that subset.**
  - Belief in (red) entails Belief in (red,green),(red,blue),(red,blue,green)

- **Belief (or possibility) is the probability that B is provable (supported) by the evidence.**
  - $Bel(A) = \Sigma_{\{B \text{ in } A\}} m(B)$ (Support committed to A)
  - *Bel ((red,blue)) = m((red))+m((blue))+m((red,blue))*
- **Plausibility is the probability that B is compatible with the available evidence (cannot be disproved).**
  - Upper belief limit on the proposition A
- **$Pl(A) = \Sigma_{\{B \cap A \neq \{\}\}} m(B)$ Support that can move into A**
  - *Pl((red,blue)) = m((red))+m((blue))+m((red,blue))+m((red,green))+ m((red,blue,green))+m((blue,green))*
- **$Pl(A) = 1\text{-}Bel(\neg A)$**



Learning

Learning is defined as the process of making change in the system that enables to do the task more efficiently. It is an important feature of intelligent.

Learning is the process of knowledge acquisition and skill refinement.
Knowledge acquisition (example: learning physics) implies learning new symbolic information coupled with the ability to apply that information in an effective manner

whereas skill refinement (example: riding a bicycle, playing the piano) occurs at a subconscious level by virtue of repeated practice.

Learning is a goal-directed process of a system that improves the knowledge base (KB) of the system by exploring experience and prior knowledge.

Learning method depends on (i) type of performance element,(ii) available feedback, (iii) type of component to be improved, and its representation.

Types learning of agents

Learning Agent	Sensors	Actuators
Human Agent	Eyes, Ears, etc	Hands, Legs, Mouth
Robot Agent	Camera, IR range finder	Motors
Software Agent	Keys strokes, Files contents	Display to screen, write files

Components of a learning system

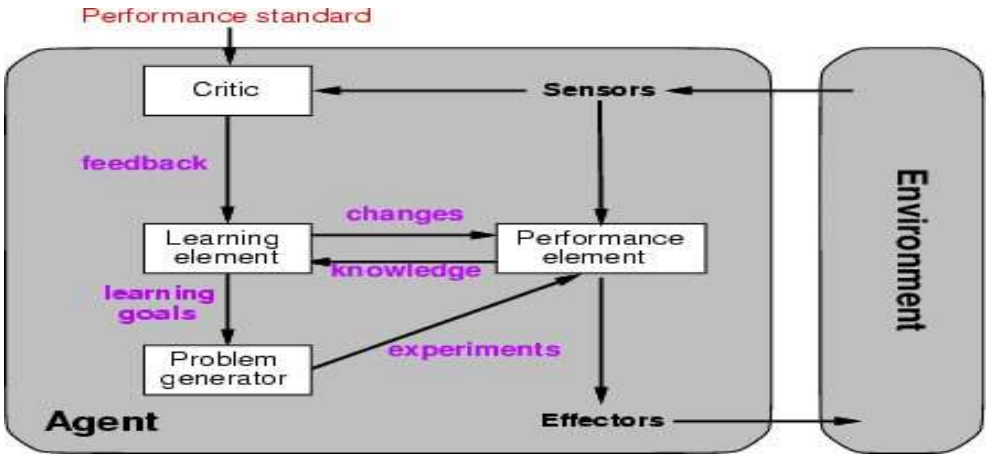
There are four components of a learning system:

(i) Learning element-It is responsible for performance improvement by taking knowledge and feedback.

(ii) Performance element- It is the agent itself, based on percept sequence it actuates through effectors.

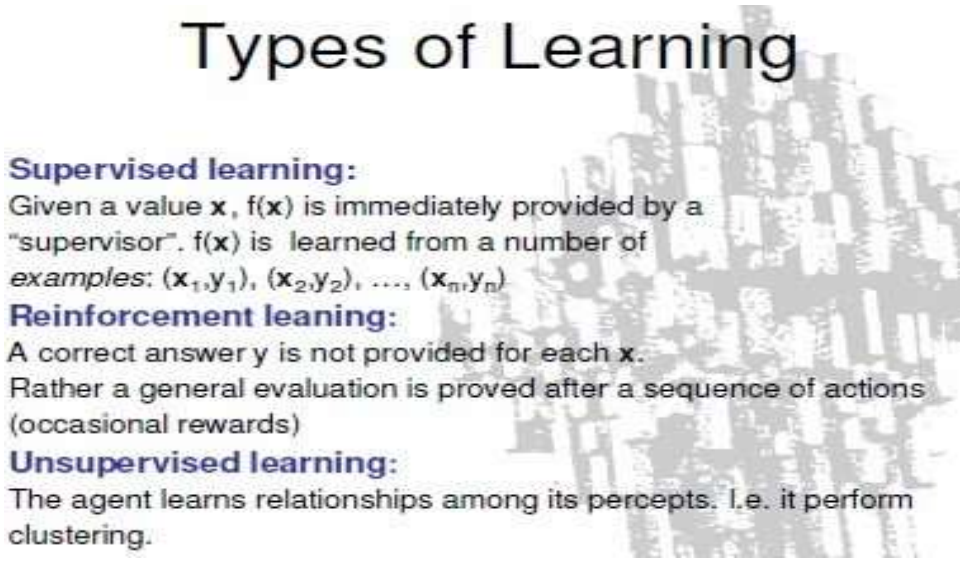
(iii) Critic-It gives feedback (success or failure) to the learning element for performance improvement.

(iv) Problem generator- It suggests actions that generate new experience for farther learning.



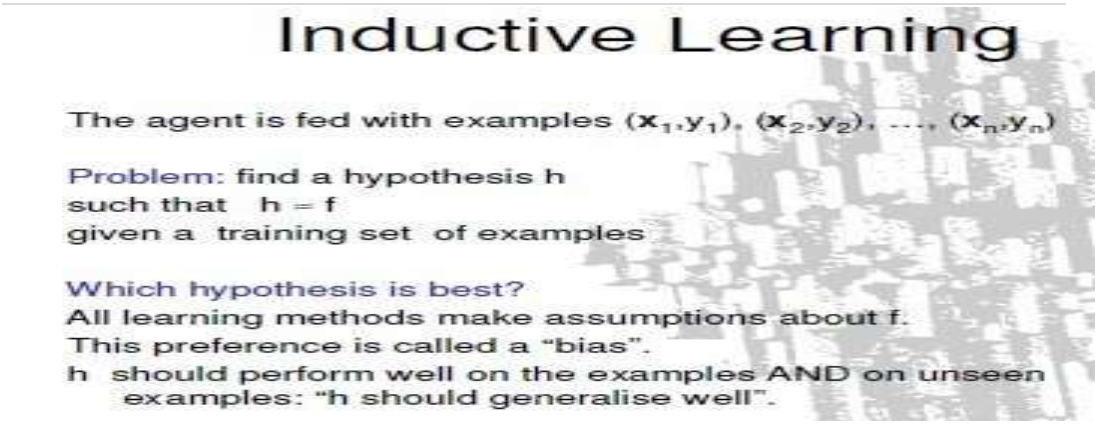
Learning from observation

- Supervised learning involves learning a function from examples of its inputs and outputs
- Unsupervised learning involves learning patterns in the input when no specific output values are supplied
- In reinforcement learning the agent must learn from reinforcement (reward, less exact feedback than in supervised learning)



Inductive learning

Learning a function from examples of its inputs and outputs is called inductive learning. It is measured by their learning curve, which shows the prediction accuracy as a function of the number of observed examples.



Learning Decision trees

Decision trees are one of the simplest, universal and most widely used prediction models.

- A *decision tree* takes as input an object or situation described by a set of attributes
- It returns a decision — the predicted output value for the input
- If the output values are discrete, then the decision tree *classifies* the inputs
- Learning a continuous function is called *regression*
- Each internal node in the tree corresponds to a to a test of the value of one of the properties, and the branches from the node are labeled with possible values of the test
- Each leaf node in the tree specifies the value to be returned if the leaf is reached
- To process an input, it is directed from the root of the tree through internal nodes to a leaf, which determines the output value

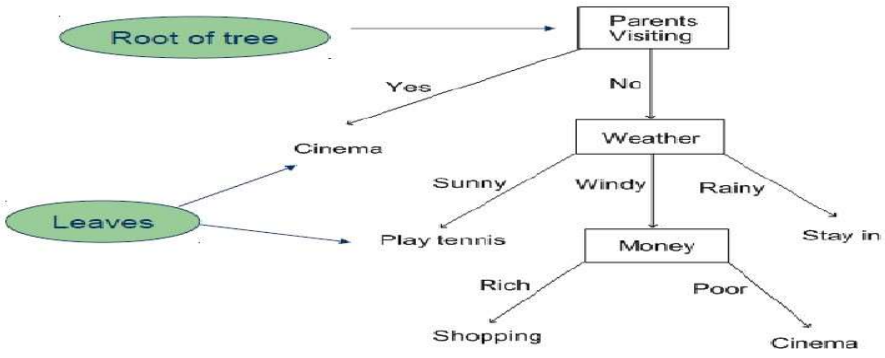
Decision Trees

Complex decisions can often be expressed in terms of a series of questions:

What to do this Weekend?

- If my parents are visiting
  - We'll go to the cinema
- If not
  - Then, if it's sunny I'll play tennis
  - But if it's windy and I'm rich, I'll go shopping
  - If it's windy and I'm poor, I'll go to the cinema
  - If it's rainy, I'll stay in

These decision can be expressed in terms of a tree



Decision trees can be written as

- Horn clauses in first order logic

Read from the root to every tip

- If this and this and this ... and this, then do this

In our example:

- If no_parents and sunny_day, then play_tennis
- no_parents $\wedge$ sunny_day $\rightarrow$ play_tennis

Learning Decision Trees

The tree can often represent a set of examples in a compact way by identifying patterns in the examples.

This is a "simpler" function with hopefully better generalisation.

Decision trees is a major tool for Machine Learning in real applications.

Explanation based learning

Learning general problem solving techniques by observing and analyzing human solutions to specific problems.

Learning by Generalizing Explanations

Given

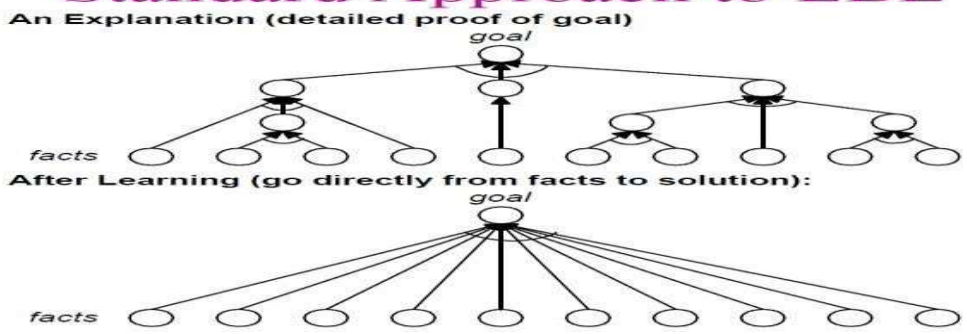
- Goal (e.g., some predicate calculus statement)
- Situation Description (facts)
- Domain Theory (inference rules)
- Operationality Criterion

Use problem solver to justify, using the *rules*, the *goal* in terms of the *facts*.

Generalize the justification as much as possible.

The *operationality criterion* states which other terms can appear in the generalized result.

Standard Approach to EBL



Unification-Based Generalization

- An explanation is an inter-connected collection of "pieces" of knowledge (inference rules, rewrite rules, etc.)
- These "rules" are connected using *unification*, as in Prolog
- The generalization task is to compute the *most general unifier* that allows the "knowledge pieces" to be connected together as generally as possible

Sample EBL Problem

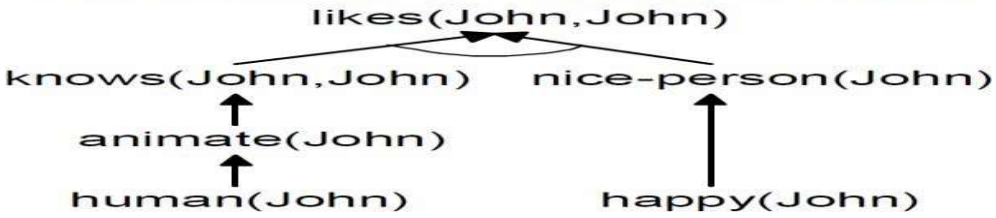
Initial Domain Theory

knows(?x,?y) AND nice-person(?y) -> likes(?x,?y)
animate(?z) -> knows(?z,?z)
human(?u) -> animate(?u)
friendly(?v) -> nice-person(?v)
happy(?w) -> nice-person(?w)

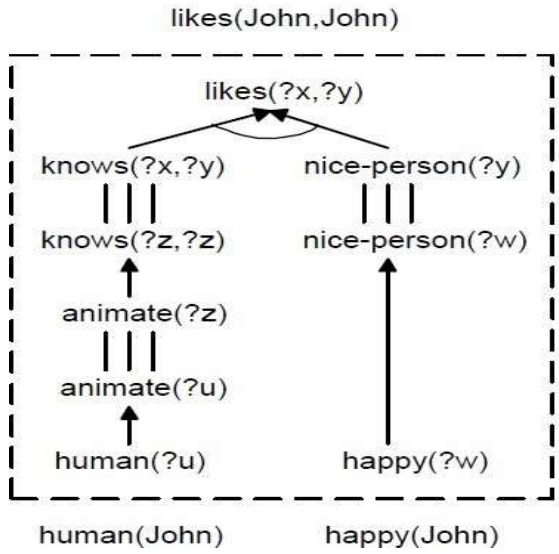
Specific Example

Given human(John) AND happy(John) AND male(John),
show that likes(John,John)

Explanation to Solve Problem



Explanation Structure



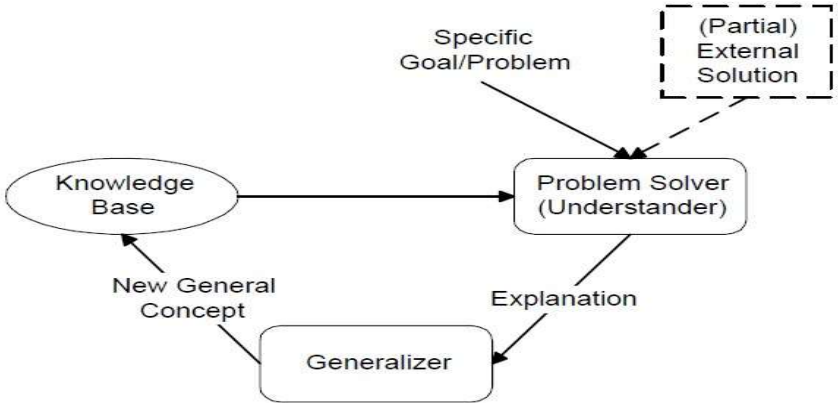
Necessary Unifications:

All variables must match ?z

Resulting Rule:

human(?z) AND happy(?z) -> likes(?z,?z)

Prototypical EBL Architecture



Neural Networks

Two major types:

Neural Networks
In our brains

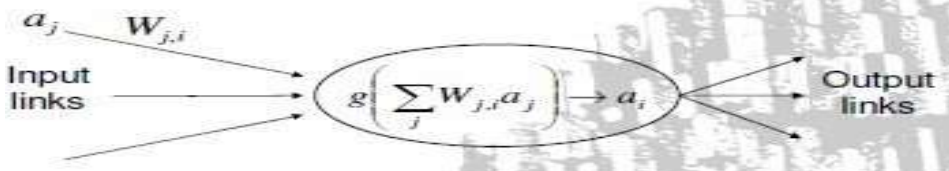
Artificial Neural Networks
Attempts to imitate our Biological Neural Networks with software and hardware

Computing Elements

Neurons
- The computational unit of the brain
- Connected through Synapses which can be Excitatory or Inhibitory
- Computes an output as a function of all inputs from surrounding neurons

Units (nodes)
- The computational unit of an Artificial Neural Network
- Connected through links with numeric weights
- Computes an output as a function of all inputs from surrounding nodes

Neuron



Terminology:

g : activation function
a_i : activation level of node i (output of the node).

The neuron is also called node, or unit.

Activation Functions

The output from a node:

$$a_i = g\left(\sum_{j=1}^{n} W_{j,i} a_j\right)$$

Common activation functions *g*:

Step function: $\text{step}_t(x) = \begin{cases} 1, & x \geq t \\ 0, & x < t \end{cases}$

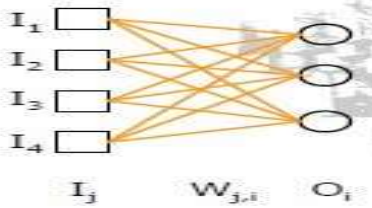
Sign function: $\text{sign}(x) = \begin{cases} +1, & x \geq 0 \\ -1, & x < 0 \end{cases}$

Sigmoid: $\text{sigmoid}(x) = \frac{1}{1+e^{-x}}$

tanh: $\tanh(x)$

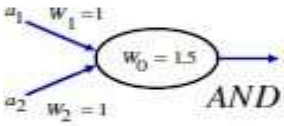
Perceptron

Input layer with 4 inputs
Output layer with 3 output nodes



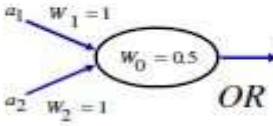
What Functions Can a Perceptron Represent?

$$a = \text{step}\left(\sum_{j=0}^{n} W_j a_j\right)$$



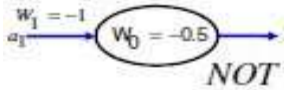
AND

$$a = \text{step}(-W_0 + W_1 a_1 + W_2 a_2) = \text{step}(-1.5 + a_1 + a_2)$$



OR

$$a = \text{step}(-W_0 + W_1 a_1 + W_2 a_2) = \text{step}(-0.5 + a_1 + a_2)$$



NOT

$$a = \text{step}(-W_0 + W_1 a_1) = \text{step}(0.5 - a_1)$$

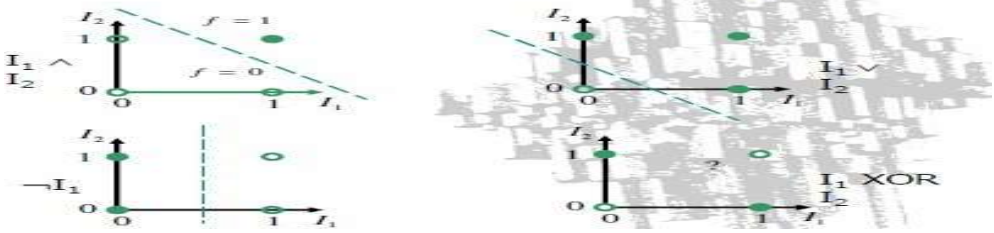
Linear Separability

Consider a perceptron with two inputs and a threshold (bias):

- The perceptron fires if $w_1 a_1 + w_2 a_2 - t \geq 0$
- Recall the weights for the "and" perceptron:
  - $a_1 + a_2 - 1.5 \geq 0$
- This is really the equation for a line!
  - $a_2 \geq - a_1 + 1.5$

The activation threshold for a perceptron is actually a linearly separable "hyperplane" in the space of inputs

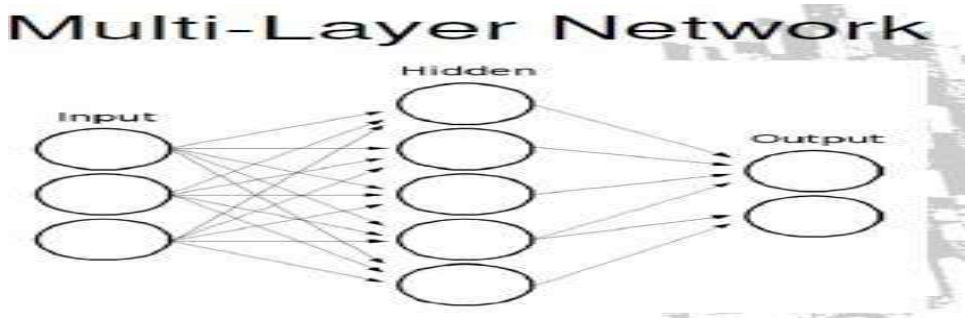
Linear Separability



Multi-Layer Networks

The structure of a multi-layer network is fairly straightforward:

- The input layer is the set of features (percepts)
- Next is a hidden layer, which has an arbitrary number of perceptrons called hidden units that take the features (input layer) as inputs
- The perceptron(s) in the output layer then takes the outputs of the hidden units as its inputs



Training Multi-Layer Network

Training multi-layer networks can be a bit complicated (the weight space is large!)

- The perceptron rule works fine for a single unit that mapped input features to the final output value
- But hidden units do not produce the final output
- Output unit(s) take other perceptrons – not known feature values – as inputs

The solution is to use the back-propagation algorithm, which is an intuitive extension of the perceptron training algorithm

Machine learning is particularly attractive in several real life problem because of the following reasons:

• Some tasks cannot be defined well except by example
• Working environment of machines may not be known at design time
• Explicit knowledge encoding may be difficult and not available
• Environments change over time
• Biological systems learn

Recently, learning is widely used in a number of application areas including,

• Data mining and knowledge discovery
• Speech/image/video (pattern) recognition
• Adaptive control
• Autonomous vehicles/robots

• Decision support systems
• Bioinformatics
• WWW

Classification & Clustering

The general idea of Classification and Clustering is to group inputs into (hopefully) distinct categories. They can also then use those categories to identify group membership of new input in the future.

Classification

Classification is a form of machine learning where two or more distinct groups, or classes, of things are known ahead of time and used to group additional things. The features of members of each class are analyzed or learned, and then generalized to build an understanding of what it means to be a member of each class. *Classification is an example of supervised learning because classes are known.*

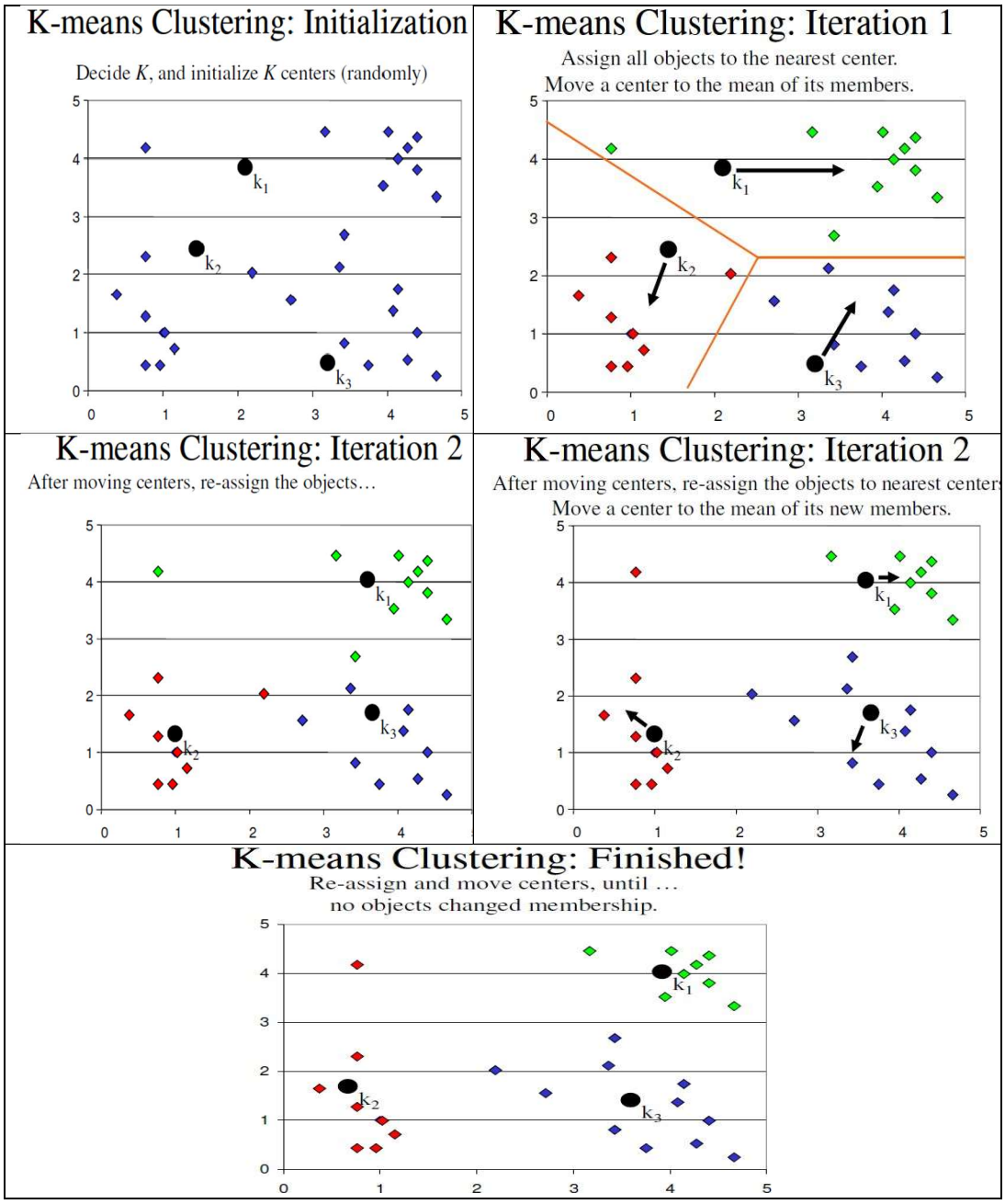
Clustering

Clustering is a form of machine learning where groups, or classes, are not known ahead of time so groupings are created by looking at similar and shared characteristics among the things being grouped.

Clustering is an example of unsupervised learning because classes are unknown at the start.

Clustering methods: Comparison

	Hierarchical	**K-means**	**GMM**
Running time	naively, $O(N^3)$	fastest (each iteration is linear)	fast (each iteration is linear)
Assumptions	requires a similarity / distance measure	strong assumptions	strongest assumptions
Input parameters	none	K (number of clusters)	K (number of clusters)
Clusters	subjective (only a tree is returned)	exactly K clusters	exactly K clusters


K-means Clustering: Initialization
Decide K, and initialize K centers (randomly)
K-means Clustering: Iteration 1
Assign all objects to the nearest center.
Move a center to the mean of its members.
K-means Clustering: Iteration 2
After moving centers, re-assign the objects…
K-means Clustering: Iteration 2
After moving centers, re-assign the objects to nearest centers
Move a center to the mean of its new members.
K-means Clustering: Finished!
Re-assign and move centers, until …
no objects changed membership.
k1
k2
k3

Algorithm *k-means*

1. Decide on a value for K, the number of clusters.

2. Initialize the K cluster centers (randomly, if necessary).

3. Decide the class memberships of the N objects by assigning them to the nearest cluster center.

4. Re-estimate the K cluster centers, by assuming the memberships found above are correct.

5. Repeat 3 and 4 until none of the N objects changed membership in the last iteration.

Comments on *K-Means*

- Strength
  - Simple, easy to implement and debug
  - Intuitive objective function: optimizes intra-cluster similarity
  - *Relatively efficient*: $O(tkn)$, where n is # objects, k is # clusters, and t is # iterations. Normally, $k, t << n$.
- Weakness
  - Applicable only when *mean* is defined, then what about categorical data?
  - Often terminates at a *local optimum*. Initialization is important.
  - Need to specify K, the *number* of clusters, in advance
  - Unable to handle noisy data and *outliers*
  - Not suitable to discover clusters with *non-convex shapes*

Reinforcement learning

Reinforcement learning is a computational approach to understanding and automating goal-directed learning and decision making. It is distinguished from other computational approaches by its emphasis on learning by an agent from direct interaction with its environment, without relying on exemplary supervision or complete models of the environment. In our opinion, reinforcement learning is the first field to seriously address the computational issues that arise when learning from interaction with an environment in order to achieve long-term goals.

Reinforcement learning:

"Reinforcement learning is a branch of machine learning concerned with using experience gained through interacting with the world and evaluative feedback to improve a system's ability to make behavioural decisions."

	Value Iteration	Passive Learning	Active Learning
States and rewards	Observes all states and rewards in environment	Observes only states (and rewards) visited by agent	Observes only states (and rewards) visited by agent
Transitions	Observes all action-transition probabilities	Observes only transitions that occur from chosen actions	Observes only transitions that occur from chosen actions
Decisions	N/A	Learning algorithm does not choose actions	Learning algorithm chooses actions

Deep learning:

"Deep learning allows computational models that are composed of multiple processing layers to learn representations of data with multiple levels of abstraction."

MCQ

1. When a computer program exhibits what appears to be human-like intelligence, it is probably using an approach known as: A Electronic thought B Digital mimicry
C Intelligent learning
D *Artificial intelligence*

2. Specific examples of Machine Learning where a computer is able to improve its own performance over time include:
A Performing millions of mathematical computations per second and drawing impressive 3D gaming graphics
B *Filtering out spam email messages and converting handwriting into computer text*

C Turning on a computer screen saver after a period of inactivity and automatically dimming a cell phone screen in low-light situations

D Finding the shortest route home on your GPS device and analyzing paint samples to get a perfect color match

3. Classification and clustering are which of these types of machine learning?

A Supervised and unsupervised learning

B Categorized and de-categorized learning C Repetitive and experiential learning

D Filtered and identified learning

4. Which one of the following everyday situations is the most like Classification? A Forming teams when the captains don't know any of the players

B Figuring out where to sit during lunchtime in a high school cafeteria

C *Deciding whether to pay using cash or credit*

D Reorganizing an accidentally dumped-out box of 64 crayons

5. What makes Clustering more difficult than Classification? A Not knowing the class labels ahead of time

B Doing lots of comparisons until you finally find the best clusters C Dealing with items that don't seem to fit well into any cluster *D All of the above*

Sample questions and answer

1. Explain the concept of learning from example.

Each person will interpret a piece of information according to their level of understanding and their own way of interpreting things.

2. What is meant by learning?

Learning is a goal-directed process of a system that improves the knowledge or the Knowledge representation of the system by exploring experience and prior knowledge.

3. How statistical learning method differs from reinforcement learning method? Reinforcement learning is learning what to do--how to map situations to actions-so as to maximize a numerical reward signal. The learner is not told which actions to take, as in most forms of machine learning, but instead must discover which actions yield the most reward by

trying them. In the most interesting and challenging cases, actions may affect not only the immediate reward but also the next situation and, through that, all subsequent rewards. These two characteristics--trial-and-error search and delayed reward--are the two most important distinguishing features of reinforcement learning.

4. Define informational equivalence and computational equivalence. A transformation from on representation to another causes no loss of information; they can be constructed from each other.

The same information and the same inferences are achieved with the same amount of effort.

5. Define knowledge acquisition and skill refinement. knowledge acquisition (example: learning physics)—learning new symbolic information coupled with the ability to apply that information in an effective manner

skill refinement (example: riding a bicycle, playing the piano)— occurs at a subconscious level by virtue of repeated practice

6. What is Explanation-Based Learning?

The background knowledge is sufficient to explain the hypothesis of Explanation-Based Learning. The agent does not learn anything factually new from the instance. It extractsgeneral rules from single examples by explaining the examples and generalizing the explanation.

7. Define Knowledge-Based Inductive Learning.

Knowledge-Based Inductive Learning finds inductive hypotheses that explain set of observations with the help of background knowledge.

8. What is truth preserving?

An inference algorithm that derives only entailed sentences is called sound or truth preserving.

9. Define Inductive learning. How the performance of inductive learning algorithms can be measured?

Learning a function from examples of its inputs and outputs is called inductive learning.

It is measured by their learning curve, which shows the prediction accuracy as a function of the number of observed examples.

10. List the advantages of Decision

Trees The advantages of Decision

Trees are,

It is one of the simplest and successful forms of learning algorithm. It serves as a good introduction to the area of inductive learning and is easy to implement.

11. What is the function of Decision Trees?

A decision tree takes as input an object or situation by a set of properties, and outputs a yes/no decision. Decision tree represents Boolean functions.

12. List some of the practical uses of decision tree learning. Some of the practical uses of decision tree learning are, Designing oil platform equipment Learning to fly

13. What is the task of reinforcement learning?

The task of reinforcement learning is to use rewards to learn a successful agent function.

14. Define Passive learner and Active learner.

A passive learner watches the world going by, and tries to learn the utility of being in various states.

An active learner acts using the learned information, and can use its problem generator to suggest explorations of unknown portions of the environment.

15. State the factors that play a role in the design of a learning system. The factors that play a role in the design of a learning system are,

- Learning element
- Performance element
- Critic
- Problem generator

16. What is memorization?

Memorization is used to speed up programs by saving the results of computation. The basic idea is to accumulate a database of input/output pairs; when the function is called, it first checks the database to see if it can avoid solving the problem from scratch.

17. Define Q-Learning.

The agent learns an action-value function giving the expected utility of taking a given action in a given state. This is called Q-Learning.

18. Define supervised learning & unsupervised learning.

Any situation in which both inputs and outputs of a component can be perceived is called supervised learning.
Learning when there is no hint at all about the correct outputs is called unsupervised learning.

19. Define Bayesian learning.
Bayesian learning simply calculates the probability of each hypothesis, given the data, and makes predictions on that basis. That is, the predictions are made by using all the hypotheses, weighted by their probabilities, rather than by using just a single "best" hypothesis.

20. What is utility-based agent?
A utility-based agent learns a utility function on states and uses it to select actions that maximize the expected outcome utility.

21. What is reinforcement learning?
Reinforcement learning refers to a class of problems in machine learning which postulate an agent exploring an environment in which the agent perceives its current state and
takes actions. The environment, in return, provides a reward (which can be positive or negative).
Reinforcement learning algorithms attempt to find a policy for maximizing cumulative reward for the agent over the curse of the problem.

22. What is the important task of reinforcement learning?
The important task of reinforcement learning is to use rewards to learn a successful agent function.

Test your skills

1. What is the difference between the terms Classification and Clustering?
2. Explain whether Classification or Clustering is harder and why.

MCQ with explanation (VVI)

Artificial Intelligence

Questions 1 to 10

1. What is Artificial intelligence? Putting your intelligence into Computer Programming with your own intelligence Making a Machine intelligent Playing a Game
Putting more memory into Computer

2. Which is not the commonly used programming language for AI?
(a) PROLOG (b) Java (c) LISP (d) Perl (e) Java script.

3. What is state space? The whole problem
Your Definition to a problem
Problem you design
Representing your problem with variable and parameter A space where You know the solution.

4. A production rule consists of
(a) A set of Rule (b) A sequence of steps
(c) Both (a) and (b) (d) Arbitrary representation to problem
(e) Directly getting solution.

5. Which search method takes less memory?
(a) Depth-First Search (b) Breadth-First search (c) Both (a) and (b)
(d) Linear Search.
(e) Optimal search.

6. A heuristic is a way of trying
To discover something or an idea embedded in a program To search and measure how far a node in a search tree seems to be from a goal To compare two nodes in a search tree to see if one is better than the other Only (a) and (b) Only (a), (b) and (c).

7. A* algorithm is based on
(a) Breadth-First-Search (b) Depth-First –Search
(c) Best-First-Search (d) Hill climbing.
(e) Bulkworld Problem.

8. Which is the best way to go for Game playing problem?
(a) Linear approach (b) Heuristic approach
(c) Random approach (d) Optimal approach
(e) Stratified approach.

9. How do you represent "All dogs have tails".
(a) ∀x: dog(x)→hastail(x) (b) ∀x: dog(x)→hastail(y)
(c) ∀x: dog(y)→hastail(x) (d) ∀x: dog(x)→has→tail(x)
(e) ∀x: dog(x)→has→tail(y)

10. Which is not a property of representation of knowledge?
(a) Representational Verification (b) Representational Adequacy
(c) Inferential Adequacy (d) Inferential Efficiency
(e) Acquisitional Efficiency.

Answers

1. Answer : (c)
What is Artificial intelligence?
: Because AI is to make things work automatically through machine without using human effort. Machine will give the result with just giving input from human. That means the system or machine will act as per the requirement.
2. Answer : (d)
: Because Perl is used as a script language, and not of much use for AI practice. All others are used to generate an artificial program to a great extent.
3. Answer : (d) **Q. What is state space? Solve the water jug problem.**
: Because state space is mostly concerned with a problem, when you try to solve a problem, we have to design a mathematical structure to the problem which can only be through variables and parameters. Ex. You have given a 4-gallon jug and another 3gallon jugs. Neither has measuring marker on it. You have to fill the jugs with water .How can you get exactly 2 gallons of water in to 4gallons.Here the *state space can* defined as set of ordered pairs *integers(x,y),such that x=0,1,2,3*

or 4 and y=0,1,2 or 3;X represents the number of gallons in 4galoon jug and y represents quantity of water in the 3-gallon jug.

4. Answer : (c)
: When you are trying to solve a problem, you should design how to get a step by step solution with constraints condition to your problem, e.g Chess board problem.

---

5. Answer : (a)
: Depth-First Search takes less memory since only the nodes on the current path are stored, but in Breadth First Search, all of the tree that has generated must be stored.
6. Answer : (e)
: In a heuristic approach we discover certain idea and use heuristic functions to search for a goal and predicates to compare nodes.
7. Answer : (c)
: Because Best-first-search is giving the idea of optimization and quick choose of path, and all these characteristic lies in A* algorithm.
8. Answer : (b)
: We use Heuristic approach as it will find out brute force computation ,looking at hundreds of thousands of positions. e.g Chess competition between Human and AI based Computer.
9. Answer : (a)
: We represent the statement in mathematical logic taking ‘x ‘as Dog and which has tail. We can not represent two variable x, y for the same object Dog which has tail. The symbol “ᵛ “represent all.
10. Answer : (a)
: There is nothing to go for Representational verification, the verification comes under Representational adequacy.

Artificial Intelligence

Questions 11 to 20

11. What are you predicating by the logic: ᵛx: €y: loyalto(x, y).

(a) Everyone is loyal to some one (b) Everyone is loyal to all
(c) Everyone is not loyal to someone (d) Everyone is loyal (e) Everyone is not loyal.

12. Which is not Familiar Connectives in First Order Logic?
(a) and (b) iff (c) or (d) not (e) either a or b.

13. Which is not a type of First Order Logic (FOL) Sentence?
(a) Atomic sentences (b) Complex sentences
(c) Quantified sentence (d) Quality Sentence
(e) Simple sentence.

14. Which is not a Goal-based agent?
(a) Inference (b) Search (c) Planning
(d) Conclusion (e) Dynamic search.

---

15. A plan that describe how to take actions in levels of increasing refinement and specificity is
(a) Problem solving (b) Planning
(c) Non-hierarchical plan (d) Hierarchical plan (e) Inheritance.

16. A constructive approach in which no commitment is made unless it is necessary to do so, is (a) Least commitment approach (b) Most commitment approach
(c) Nonlinear planning (d) Opportunistic planning
(e) Problem based planning.

17. Partial order planning involves Searching over the space of possible plans Searching over possible situations Searching the whole problem at once Searching the best Searching the goal.

18. Which is true for Decision theory? Decision Theory = Probability theory + utility theory Decision Theory = Inference theory + utility theory Decision Theory = Uncertainty + utility theory Decision Theory = Probability theory + preference
Decision Theory = Probability theory + inference.

19. Uncertainty arises in the wumpus world because the agent's sensors give only
(a) Full & Global information (b) Partial & Global Information
(c) Partial & local Information (d) Full & local information
(e) Global information only.

20. A Hybrid Bayesian network contains
Both discrete and continuous variables Only Discrete variat
Only Discontinuous variable
Both Discrete and Discontinuous variable Continous variable only.

Answers

11. Answer : (a)
    : ∀x denotes Everyone or all, and ∈y someone and loyal to is the proposition logic making map x to y.
12. Answer : (d)
    : “not” is coming under propositional logic and is therefore not a connective.
13. Answer : (d)
    Reason : Quantity structure is not a FOL structure while all other are.
14. Answer : (d)
    : Conclusion is a statement to Goal-based agent, but is not considered as Goal- based agent.

---

15. Answer : (d)
    : A plan that describes how to take actions in levels of increasing refinement and specificity is Hierarchical (e.g., "Do something" becomes the more specific "Go to work," "Do work," "Go home.") Most plans are hierarchical in nature.
16. Answer : (a)
    : Because we are not sure about the outcome.
17. Answer : (a)
    : Partial order planning involves searching over the space of possible plans, rather than searching over possible situations. The idea is to construct a plan piece-by- piece. There are two kinds of steps we can take in constructing a plan: add an operator (action), or add an ordering constraint between operators. The name "partial order planning" comes from the fact that until we add the ordering constraints, we don't specify the order in which actions are taken. This (sometimes) allows a partial order planning to avoid lots of backtracking that would slow down a state-space planner.
18. Answer : (a)
    : Utility theory to represent and reason with preference. Preference is expressed by utilities. Utilities are combined with probabilities in the general theory of rational decisions called decision theory. Decision theory, which combines probability theory with utility theory, provides a formal and complete frame work for decisions (economic or otherwise) made under uncertainty-that is, in case where
    probabilistic descriptions appropriately capture the decision-maker’s environment.
19. Answer : (c)

: The Wumpus world is a grid of squares surrounded by walls, where each square can contain agents and objects. The agent (you) always starts in the lower left corner, a square that will be labeled [1, 1]. The agent's task is to find the gold, return to [1, 1] and climb out of the cave. So uncertainty is there as the agent gives partial and local information only. Global variable are not goal specific problem solving.

20. Answer : (a)

: To specify a Hybrid network, we have to specify two new kinds of distributions: the conditional distribution for continuous variables given discrete or continuous parents, and the conditional distribution for a discrete variable given continuous parents.

Artificial Intelligence

Questions 21 to 30

21. Which is not a desirable property of a logical rule-based system?
(a) Locality (b) Attachment (c) Detachment
(d) Truth-Functionality (e) Global attribute.

---

22. How is Fuzzy Logic different from conventional control methods? (a) IF and THEN Approach (b) FOR Approach
(c) WHILE Approach (d) DO Approach
(e) Else If approach.

23. In an Unsupervised learning Specific output values are given
Specific output values are not given No specific Inputs are given
Both inputs and outputs are given Neither inputs nor outputs are given.

24. Inductive learning involves finding a
(a) Consistent Hypothesis (b) Inconsistent Hypothesis
(c) Regular Hypothesis (d) Irregular Hypothesis
(e) Estimated Hypothesis.

25. Computational learning theory analyzes the sample complexity and computational complexity of
(a) UnSupervised Learning (b) Inductive learning
(c) Forced based learning (d) Weak learning (e) Knowledge based learning.

26. If a hypothesis says it should be positive, but in fact it is negative, we call it

(a) A consistent hypothesis (b) A false negative hypothesis
(c) A false positive hypothesis (d) A specialized hypothesis
(e) A true positive hypothesis.

27. Neural Networks are complex ----------with many parameters.
(a) Linear Functions (b) Nonlinear Functions (c) Discrete Functions (d) Exponential Functions
(e) Power Functions.

28. A perceptron is a ----------.
(a) Feed-forward neural network (b) Back-propagation alogorithm (c) Back-tracking algorithm (d) Feed Forward-backward algorithm (e) Optimal algorithm with Dynamic programming.

29. Which is true?
Not all formal languages are context-free All formal languages are Contex free
All formal languages are like natural language Natural languages are contextoriented free Natural language is formal.

30. Which is not true?
(a) The union and concatenation of two context-free languages is context-free
(b) The reverse of a context-free language is context-free, but the complement need not be
(c) Every regular language is context-free because it can be described by a regular grammar
(d) The intersection of a context-free language and a regular language is always context-free
(e) The intersection two context-free languages is context-free.

Answers

21. Answer : (b)
: Locality: In logical systems, whenever we have a rule of the form A => B, we can conclude B, given evidence A, without worrying about any other rules. Detachment: Once a logical proof is found for a proposition B, the proposition can be used regardless of how it was derived .That is, it can be detachment from its justification. Truth-functionality: In logic, the truth of complex sentences can be computed from the truth of the components. But there are no Attachment

properties lies in a Rule-based system. Global attribute defines a particular problem space as user specific and changes according to user's plan to problem.

22. Answer : (a) **Q.What is fuzzy logic? Explain.**

: FL incorporates a simple, rule-based IF X AND Y THEN Z approach to a solving control problem rather than attempting to model a system mathematically. The FL model is empirically-based, relying on an operator's experience rather than their technical understanding of the system. For example, rather than dealing with temperature control in terms such as "SP =500F", "T <1000F", or "210C <TEMP <220C", terms like "IF (process is too cool) AND (process is getting colder) THEN (add heat to the process)" or "IF (process is too hot) AND (process is heating rapidly) THEN (cool the process quickly)" are used. These terms are imprecise and yet very descriptive of what must actually happen. Consider what you do in the shower if the temperature is too cold: you will make the water comfortable very quickly with little trouble. FL is capable of mimicking this type of behavior but at very high rate.

23. Answer : (b**)** **Q. What is Unsupervise learning?**

: The problem of unsupervised learning involves learning patterns in the input when no specific out put values are supplied. We can not expect the specific output to test your result. Here the agent does not know what to do, as he is not aware of the fact what propose system will come out. We can say an ambiguous unproposed situation.

24. Answer : (a)

: Inductive learning involves finding a consistent hypothesis that agrees with examples. The difficulty of the task depends on the chosen representation.

25. Answer : (b)

: Computational learning theory analyzes the sample complexity and computational complexity of inductive learning. There is a trade off between the expressiveness of the hypothesis language and the ease of learning.

26. Answer : (c) **Q. What is false positive and false negetive hypotheses?**

: Consistent hypothesis go with examples, If the hypothesis says it should be negative but infact it is positive, it is false negative. If a hypothesis says it should be positive, but in fact it is negative, it is false positive. In a specialized hypothesis we need to have certain restrict or special conditions.

---

27. Answer : (b) **Q.Why neural network use non linear funftion?**

: Neural networks parameters can be learned from noisy data and they have been used for thousands of applications, so it varies from problem to problem and thus use nonlinear functions.

28. Answer : (a) **Q. What is perceptron?**

: A perceptron is a Feed-forward neural network with no hidden units that can be represent only linear separable functions. If the data are linearly separable, a simple weight updated rule can be used to fit the data exactly.

29. Answer : (a)

: Not all formal languages are context-free — a well-known counterexample is

This particular language can be generated by a parsing expression grammar, which is a relatively new formalism that is particularly well-suited to programming languages.

30. Answer : (e)

: The union and concatenation of two context-free languages is context-free; but intersection need not be.

Artificial Intelligence

Questions 31 to 40

31What is a Cybernetics?

. (a) Study of communication between two machines

(b) Study of communication between human and machine

(c) Study of communication between two humans

(d) Study of Boolean values

(e) Study of communication between logic circuits.

32What is the goal of artificial intelligence?

. (a) To solve real-world problems

(b) To solve artificial problems (c)

(d) To explain various sorts of intelligence

To extract scientific causes

(e) To restrict problems.

33. An algorithm is complete if

(a) It terminates with a solution when one exists

(b) It starts with a solution

(c) It does not terminate with a solution

(d) It has a loop

(e) It has a decision parameter.

34Which is **true** regarding BFS?

. (a) BFS will get trapped exploring a single path (b) The entire tree so far been generated must be stored in BFS

(c) BFS is not guaranteed to find a solution, if exists

(d) BFS is nothing but Binary First Search (e) BFS is one type of sorting.

35What is a heuristic function?

. (a) A function to solve mathematical problems

(b) A function which takes parameters of type string and returns an integer value

(c) A function whose return type is nothing (d) A function which returns an object

(e) A function that maps from problem state descriptions to measures of desirability.

36The traveling salesman problem involves n cities with paths connecting the cities. The

. time taken for traversing through all the cities, without knowing in advance the length of a minimum tour, is

(a) O(n)

(b) O(n2)

(c) O(n!)

(d) O(n/2) (e) O(2n).

37The problem space of means-end analysis has

. (a) An initial state and one or more goal states

(b) One or more initial states and one goal state

(c) One or more initial states and one or more goal state

(d) One initial state and one goal state (e) No goal state.

38An algorithm A is admissible if

. (a) It is not guaranteed to return an optimal solution when one exists

(b) It is guaranteed to return an optimal solution when one exists

(c) It returns more solutions, but not an optimal one (d) It guarantees to return more optimal solutions (e) It returns no solutions at all.

39Knowledge may be .

I. Declarative.

II. Procedural. III. Non-procedural.

(a) Only (I) above
(b) Only (II) above
(c) Only (III) above
(d) Both (I) and (II) above
(e) Both (II) and (III) above.

40. Idempotency law is
I. P P = P. II. P P = P.
III. P + P = P.

(a) Only (I) above
(b) Only (II) above
(c) Only (III) above
(d) Both (I) and (II) above
(e) Both (II) and (III) above.

Answers

31. Answer : (b)
: Cybernetics is Study of communication between human and machine

32. Answer : (c) **Q. What is the goal of artificial intelligence?**

: The scientific goal of artificial intelligence is to explain various sorts of intelligence

33. Answer : (a)
: An Algorithm is complete if It terminates with a solution when one exists.

34. Answer : (b)
: Regarding BFS-The entire tree so far been generated must be stored in BFS.

35. Answer : (e) **Q. What is heuristic function?**
: Heuristic function is a function that maps from problem state descriptions to measures of desirability

36. Answer : (c)

: The traveling salesman problem involves n cities with paths connecting the cities. The time taken for traversing through all the cities, without knowing in advance the length of a minimum tour, is O(n!)

37. Answer : (a)

: The problem space of means-end analysis has an initial state and one or more goal states

38. Answer : (b) **Q. What is the condition for admissibility of an algorithm?**

: An algorithm A is admissible if It is guaranteed to return an optimal solution when one exists.

39. Answer : (d)

: Knowledge may be declarative and procedural

40. Answer : (a)

: Idempotency Law is P V P = P

Artificial Intelligence

Questions 41 to 50

41. Which of the following is **true** related to 'Satisfiable' property?

(a) A statement is satisfiable if there is some interpretation for which it is false (b) A statement is satisfiable if there is some interpretation for which it is true

(c) A statement is satisfiable if there is no interpretation for which it is true

(d) A statement is satisfiable if there is no interpretation for which it is false

(e) None of the above.

42. Two literals are complementary if

(a) They are equal

(b) They are identical and of equal sign (c) They are identical but of opposite sign (d) They are unequal but of equal sign

(e) They are unequal but of opposite sign.

43. Consider a good system for the representation of knowledge in a particular domain. What property should it possess?

(a) Representational Adequacy

(b) Inferential Adequacy

(c) Inferential Efficiency (d) Acquisitional Efficiency (e) All the above.

44. What is Transposition rule?

(a) From P → Q, infer ~Q → P
(b) From P → Q, infer Q → ~P
(c) From P → Q, infer Q → P
(d) From P → Q, infer ~Q → ~P
(e) None of the above.

45. Third component of a planning system is to (a) Detect when a solution has been found
(b) Detect when solution will be found
(c) Detect whether solution exists or not
(d) Detect whether multiple solutions exist
(e) Detect a solutionless system.

46. Which of the following is **true** in Statistical reasoning?
(a)The representation is extended to allow some kind of numeric measure of certainty to be associated with each statement
(b)The representation is extended to allow 'TRUE or FALSE' to be associated with each statement
(c) The representation is extended to allow some kind of numeric measure of certainty to be associated common to all statements
(d)The representation is extended to allow 'TRUE or FALSE' to be associated common to all statements (e)None of the above.

47. In default logic, we allow inference rules of the form
(a) (A : B) / C
(b) A / (B : C)
(c) A / B (d) A / B : C
(e) (A: B) :C.

48. In Baye's theorem, what is the meant by P(Hi|E)?
(a) The probability that hypotheses Hi is true given evidence E
(b) The probability that hypotheses Hi is false given evidence E
(c) The probability that hypotheses Hi is true given false evidence E
(d) The probability that hypotheses Hi is false given false evidence E
(e) The probability that hypotheses Hi is true given unexpected evidence E.

49. Default reasoning is another type of
(a) Monotonic reasoning
(b) Analogical reasoning

(c) Bitonic reasoning

(d) Non-monotonic reasoning

(e) Closed world assumption.

50. Generality is the measure of

(a) Ease with which the method can be adapted to different domains of application

(b) The average time required to construct the target knowledge structures from some specified initial structures

(c) A learning system to function with unreliable feedback and with a variety of training examples

(d) The overall power of the system (e) Subdividing the system.

Answers

41. Answer : (b)
: 'Satisfiable' property is a statement is satisfiable if there is some interpretation for which it is true.

42. Answer : (c)
: Two literals are complementary if They are identical but of opposite sign.

43. Answer : (e)
: Consider a good system for the representation of knowledge in a particular
domain. The properties should be Representational Adequacy,Inferential Adequacy,Inferential Efficiency and Acquisitional Efficiency

44. Answer : (d)
: Transposition rule- From P → Q, infer ~Q → ~P

45. Answer : (a)
: Third component of a planning system is to detect when a solution has been found.

46. Answer : (a)
: Statistical reasoning is the representation is extended to allow some kind of numeric measure of certainty to be associated with each statement.

47. Answer : (a)
: In default logic, we allow inference rules of the form:(A : B) / C

48. Answer : (a)
: In Baye's theorem, P(Hi|E) is the probability that hypotheses Hi is true given evidence E.

49. Answer : (d)
: Default reasoning is another type of non-monotonic reasoning

50. Answer : (a)
: Generality is the measure of ease with which the method can be adapted to different domains of application.

Artificial Intelligence

Questions 51 to 60

51. Machine learning is

(a) The autonomous acquisition of knowledge through the use of computer programs

(b) The autonomous acquisition of knowledge through the use of manual programs
(c) The selective acquisition of knowledge through the use of computer programs
(d) The selective acquisition of knowledge through the use of manual programs
(e) None of the above.

52. Factors which affect the performance of learner system does **not** include
(a) Representation scheme used
(b) Training scenario
(c) Type of feedback (d) Good data structures (e) Learning algorithm.

53. Different learning methods does **not** include
(a) Memorization
(b) Analogy
(c) Deduction
(d) Introduction (e) Acceptance.

54. In language understanding, the levels of knowledge that does **not** include (a) Phonological
(b) Syntactic
(c) Semantic
(d) Logical (e) Empirical.

55. A model of language consists of the categories which does **not** include
(a) Language units
(b) Role structure of units
(c) System constraints (d) Structural units (e) Components.

56. Semantic grammars
(a) Encode semantic information into a syntactic grammar
(b) Decode semantic information into a syntactic grammar
(c) Encode syntactic information into a semantic grammar
(d) Decode syntactic information into a semantic grammar
(e) Encode syntactic information into a logical grammar.

57. What is a top-down parser?
(a) Begins by hypothesizing a sentence (the symbol S) and successively predicting
lower level constituents until individual preterminal symbols are written

(b) Begins by hypothesizing a sentence (the symbol S) and successively predicting upper level constituents until individual preterminal symbols are written

(c) Begins by hypothesizing lower level constituents and successively predicting a sentence (the symbol S)

(d) Begins by hypothesizing upper level constituents and successively predicting a sentence (the symbol S) (e) All the above.

58. Perception involves

(a) Sights, sounds, smell and touch (b) Hitting

(c) Boxing

(d) Dancing (e) Acting.

59. Among the following which is **not** a horn clause?

(a) p (b) p V q

(c) $p \rightarrow q$

(d) $p \rightarrow q$

(e) All the above.

60. The action 'STACK(A, B)' of a robot arm specify to

(a) Place block B on Block A

(b) Place blocks A, B on the table in that order

(c) Place blocks B, A on the table in that order

(d) Place block A on block B (e) POP A, B from stack.

Answers

51. Answer : (a) **Q. What is machine learning?**

: Machine learning is the autonomous acquisition of knowledge through the use of computer programs.

52. Answer : (d)

: Factors which affect the performance of learner system does not include good data structures

53. Answer : (d)

: **D**ifferent learning methods does not include introduction

54. Answer : (e)

: In language understanding, the levels of knowledge that does not include empirical knowledge

55. Answer : (d)

: A model of language consists of the categories which does not include structural units

56. Answer : (a)

: Semantic grammars encode semantic information into a syntactic grammar.

57. Answer : (a)

: A top-down parser begins by hypothesizing a sentence (the symbol S) and successively predicting lower level constituents until individual preterminal symbols are written.

58. Answer : (a)

: Perception involves Sights, sounds, smell and touch.

59. Answer : (d)

: $p \rightarrow \sim q$ is not a horn clause

60. Answer : (d)

: The action 'STACK(A,B)' of a robot arm specify to Place block A on block B.

Module-5

Elements of robots

Elements of robots: Position and orientation of a rigid body, Homogeneous transformations, Representation of joints, link representation using D-H parameters, Examples of D-H parameters and link transforms, different kinds of actuators – stepper, DC servo motors, Purpose of sensors– tachometers, strain gauge based force-torque sensors, proximity sensors and vision.

Position and orientation of a rigid body

The position and orientation of a rigid body can be described by attaching a frame to it. . After defining a reference coordinate system, the position and orientation of the rigid body are fully described by the position of the frame's origin and the orientation of its axes, relative to the reference frame. In geometry the orientation, angular position, or attitude of an object such as a line, plane or rigid body is part of the description of how it is placed in the space it occupies. ... Euler's rotation theorem shows that in three dimensions any orientation can be reached with a single rotation around a fixed axis.

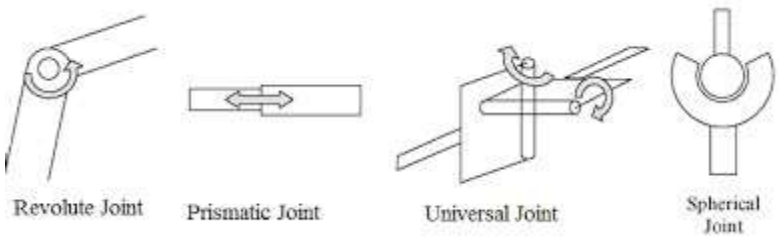
Homogeneous transformations, Representation of joints:

Homogeneous transformations:

To represent any position and orientation of a body is represented by rigid-body homogeneous transformation matrix, If the first body is only capable of rotation via a revolute joint, then a simple convention is usually followed.

The purpose of this chapter is to introduce you to the Homogeneous Transformation. This simple 4 x 4 transformation is used in the geometry engines of CAD systems and in the kinematics model in robot controllers. It is very useful for examining rigid-body position and orientation (pose) of a sequence of robotic links and joint frames.

Representation of joints



Link representation using D-H parameters

In mechanical engineering, the **Denavit–Hartenberg parameters** (also called **DH parameters**) are the four parameters associated with a particular convention for attaching reference frames to the links of a spatial kinematic chain, or robot manipulator.

Jacques Denavit and Richard Hartenberg introduced this convention in 1955 in order to standardize the coordinate frames for spatial linkages.

Richard Paul demonstrated its value for the kinematic analysis of robotic systems in 1981. While many conventions for attaching reference frames have been developed, the Denavit–Hartenberg convention remains a popular approach.

Examples of D-H parameters and link transforms

DH frame transformations Examples – Three-link planar arm , RP and PR arms, Spatial arm

Different kinds of actuators – stepper, DC servo motors :

An **actuator** is a component of a machine that is responsible for moving and controlling a mechanism or system, for example by opening a valve. In simple terms, it is a "mover". An **actuator** requires a control signal and a source of energy.

A **stepper motor**, also known as **step motor** or **stepping motor**, is a brushless DC electric motor that divides a full rotation into a number of equal steps. The motor's position can then be commanded to move and hold at one of these steps without any position sensor for feedback (an open-loop controller), as long as the motor is carefully sized to the application in respect to torque and speed. Switched reluctance motors are very large stepping motors with a reduced pole count, and generally are closed-loop commutated.

D.C. Servo Motors may be used to provide a corrective action in remote control positioning systems or in controlled velocity drive mechanisms. They may also be used as a small power drive

operating in open loop controls. Normally a servo system will consist of an input command in the form of a voltage and a power amplifier to convert the signal into current to suit the drive motor. The D.C. Servo Motor may operate in open loop where friction damping is minimised and the effect of the permanent magnet can provide satisfactory performance. The linear operating characteristics enable relatively simple control circuits to be used. For fast acting and more efficient systems it may be necessary to provide additional damping and the most convenient form is an electrical feedback signal related to motor speed. This can be provided by a separate D.C. Tacho-Generator coupled to the load or the motor shaft. Provisionally, we have selected standard models. However, the possibilities for custom-designed motors are limitless. Please contact Muirhead Aerospace who will be pleased to discuss your individual requirement. Typical Applications: - Infra-red Detector Scanning - Optical Scanning Drives - Antenna Drive Systems - Valve Actuation - Camera Color Mechanisms

Purpose of sensors– tachometers, strain gauge based force-torque sensors, proximity sensors and vision

a **sensor** is a device, module, or subsystem whose **purpose** is to detect events or changes in its environment and send the information to other electronics, frequently a computer processor. A **sensor** is always used with other electronics.

A **tachometer** (revolution-counter, tach, rev-counter, RPM gauge) is an instrument measuring the rotation speed of a shaft or disk, as in a motor or other machine. The device usually displays the revolutions per minute (RPM) on a calibrated analogue dial, but digital displays are increasingly common.

Torque Sensors (or Torque Sensors) measure torque in a variety of methods. The basic principle is, in essence, a very simple mechanical process, it's a measure of the 'force', being used in turning (or attempting to turn) an element. When a force or 'torque' is applied to a shaft; the shaft twists (by a very small amount). This twisting produces a 'stretch' in the material of the shaft, in a direction at 45 degrees to the axis of the shaft, between points on the shaft that are moved apart by the twisting motion. The material of the shaft also sees a 'compression' in the opposite 45-degree direction.

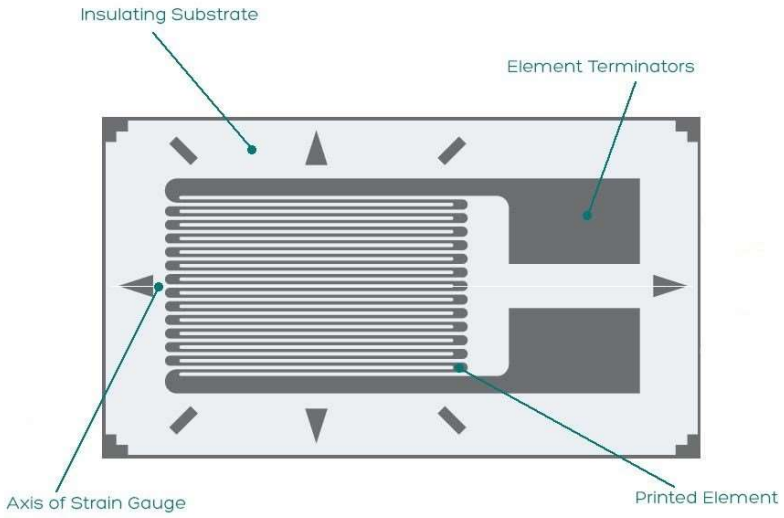
Datum Electronics torque sensors and torque transducers utilise this shaft bending and measure the change in order to calculate the torque. This measurement is achieved by the use strain gauges bonded to the shaft, measuring the strain, induced in the shaft by the applied torque or 'force'. There are various methods of measuring this twist through a shaft but strain gauges are recognised as one of the most reliable methods, if you have the expertise to achieve this.

Strain Gauge Based Torque Sensors & Torque Transducers

Strain Gauges are incredibly accurate if you have the expertise and experience of using and understanding how they work. They have been traditionally associated with inaccurate and unreliable measurements. If used correctly with the right understanding, they are very reliable, stable, robust and a cost-effective method of torque measurement.

Definition of a Strain Gauge

A strain gauge is a small electrical 'element' printed on a non-conductive substrate. The pattern of the element is arranged so that if the gauge is stretched (or compressed) in one direction (along an operating axis of the gauge), the resistance of the element increases (or decreases) in relation to that stretch. A stretch perpendicular to the axis of the strain gauge has little effect on the resistance of the element.



If a gauge is bonded to the shaft, with its axis aligned with the direction in which the shaft material stretches when a torque is applied, the strain gauge will also stretch and therefore the element will increase in resistance. If a gauge is bonded to the shaft, with its axis aligned with the direction in which the shaft material compresses when a torque is applied, the strain gauge will also compress and therefore the element will decrease in resistance. In the Torque transducer, strain gauges making up four resistive elements are bonded to the shaft. Two elements are aligned with the direction of Tension (stretch). The remaining two are aligned with the direction of Compression.

The four resistive elements are electrically connected in a 'Wheatstone Bridge' configuration. The Wheatstone Bridge configuration is appropriate for measurement of the small resistance changes produced in the strain gauges, as the combination increases and decreases in resistance it produces a change in output voltage which is only proportional to excitation voltage and change of resistance between opposing elements, not to any overall change in resistance such as might be produced by a change in temperature.

proximity sensors and vision.

A **proximity sensor** is a sensor able to detect the presence of nearby objects without any physical contact.

A proximity sensor often emits an electromagnetic field or a beam of electromagnetic radiation (infrared, for instance), and looks for changes in the field or return signal. The object being sensed is often referred to as the proximity sensor's target. Different proximity sensor targets demand different sensors. For example, a capacitive proximity sensor or photoelectric sensor might be suitable for a plastic target; an inductive proximity sensor always requires a metal target. Proximity sensors can have a high reliability and long functional life because of the absence of mechanical parts and lack of physical contact between the sensor and the sensed object.

Proximity sensors are also used in machine vibration monitoring to measure the variation in distance between a shaft and its support bearing. This is common in large steam turbines, compressors, and motors that use sleeve-type bearings.

International Electrotechnical Commission (IEC)defines the technical details of proximity sensors.

A proximity sensor adjusted to a very short range is often used as a touch switch.

Proximity sensors are commonly used on mobile devices. When the target is within nominal range, the device lock screen user interface will appear, thus emerging from what is known as sleep mode. Once the device has awoken from sleep mode, if the proximity sensor's target is still for an extended period of time, the sensor will then ignore it, and the device will eventually revert into sleep mode. For example, during a telephone call, proximity sensors play a role in detecting (and skipping) accidental touchscreen taps when mobiles are held to the ear.

Proximity sensors can be used to recognise air gestures and hover-manipulations. An array of proximity sensing elements can replace vision-camera or depth camera based solutions for the hand gesture detection.

Applications

Proximity sensor installed on the front of an iPhone 5 next to the earpiece automatically turning off the touchscreen when the sensor comes within a predefined range of an object (such as a human ear) when using the handset. Parking, systems mounted on car bumpers that sense distance to nearby cars for parking, Ground proximity warning system for aviation safety, Vibration measurements of rotating shafts in machinery, Top dead center (TDC)/camshaft sensor in engines., Sheet break sensing in paper machine. Warfare, Roller, Conveyor, beverage and food can making lines, devices, Touch that come in close proximity to the face, Attenuating radio power in close proximity to the body, in order to reduce radiation exposure, Automatic faucets.

Module – 6

Kinematics of Robots

6.1 Introduction

Kinematics of manipulators is a branch where the geometry of moving body or relative motion between connected bodies is studied without knowing the cause of the motion, means the forces, torques which cause the motion of the links. A manipulator is broadly classified as serial or parallel and consists of rigid links and joints.

Same way Robot Kinematics means the study of the movement of multi-degree of freedom kinematic chains that form the structure of robotic systems as shown in Fig. 6.1.

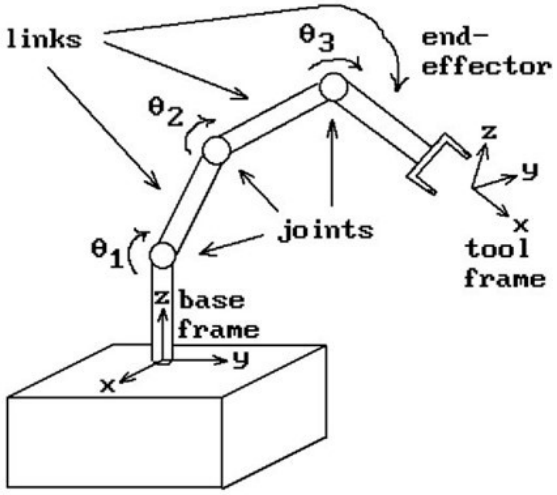


Fig. 6.1 Kinematic chain

In this module, we first go through the kinematics of serial manipulator followed by the parallel manipulator.

Tool frame and Base frame: Since in kinematic the geometry of motion is studied so a coordinate system is required to calculate the kinematic parameters such as displacement, acceleration, velocity etc. In Robot Kinematics two types of reference frames are used:

i) Tool frame: It moves with the end effectors or the last open link of the kinematic chain.

ii) Base frame: It is a fixed reference frame here usually the origin is taken on the base of the robotic chain. Above fig shows both the frames.

6.2 Degrees of Freedom of a (DOF) of a manipulator:

The degrees of freedom of a manipulator can be written as

$$dof = \lambda(N - J - 1) + \sum_{i=1}^{J} F_i \quad (6.1)$$

Where

N=Total number of links including the fixed link.

J= Total number of joints connecting only two links (if a joint connects three links, then it is counted as two joints)

F_i=degrees of freedom at the ith joint.

l=6 (for spatial manipulators) =

3 (for planar manipulators)

The quantity dof obtained from Eqn (6.1) is the number of independent actuators. It also describes the capability of a mechanism or a manipulator in terms of l. Here, l represents the dimension of the ambient space in which the motion is taking place. We have the following possibilities:

(i) dof=l, here the end effector of a manipulator can be positioned and oriented arbitrarily. l=3 for planar motion and 6 for motion in $\hat{A}^3$.

(ii) dof<l, here the arbitrary position and orientation of the end effector is not achievable.

(iii) dof>l, here any position and orientation of the end effector can be achieved in infinite ways.

6.3 Serial manipulators:

- Serial manipulators: One end fixed → Links and joints → Free end with end-effector.
- Kinematics → Motion of (rigid) links without considering force and torques.
- Serial manipulators modelled using Denavit-Hartenberg (DH) parameters
- Two main problems: Direct Kinematics and Inverse Kinematics.

In fig. 6.2, some examples of serial robots are shown.

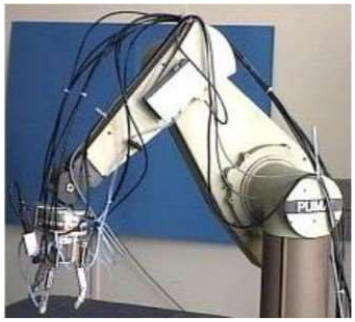


Fig. 6.2 Example of serial robots

6.4 Direct kinematics of a serial manipulator:

The direct kinematics problem for serial manipulators can be stated as follows:

Given the link parameters and the joint variable, a_{i-1}, a_{i-1}, d_i and q_i, find the position and orientation of the last link in the fixed or reference coordinate system.

- Since all D-H parameters are known → All 4×4 link transforms $^{i-1}_{i}[T]$, i = 1,..,n are known.
- If the fixed coordinate system is {0} and the coordinate system of the end-effector is {n}, we can write

$$0[\mathrm{T}] = 0[\mathrm{T}]\ 1[\mathrm{T}] \ldots n-1[\mathrm{T}] \quad (6.2)$$

n 1 2 n

It is noted that the link transformation matrices n the right hand side are functions of a_{i-1}, a_{i-1}, d_i and q_i

- For another reference {Base }, we can get (Fig. 6.3)

$$Base_{Tool}[\mathrm{T}] = {}^{Base}0[\mathrm{T}]\ {}^{n}0[\mathrm{T}] \ldots {}^{Tool}n\ [\mathrm{T}] \quad (6.3)$$

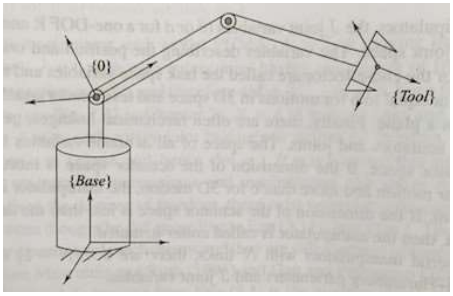


Fig. 6.3 The {Base} and {Tool} frames

The direct kinematics problem can be solved for two serial manipulators are as below:

(i) **The planar 3R manipulator**:

The planar 3R manipulator is shown in Fig. 6.4. The orientation of the tool or the gripper can be described by an angle j as shown in Fig. 6.4.

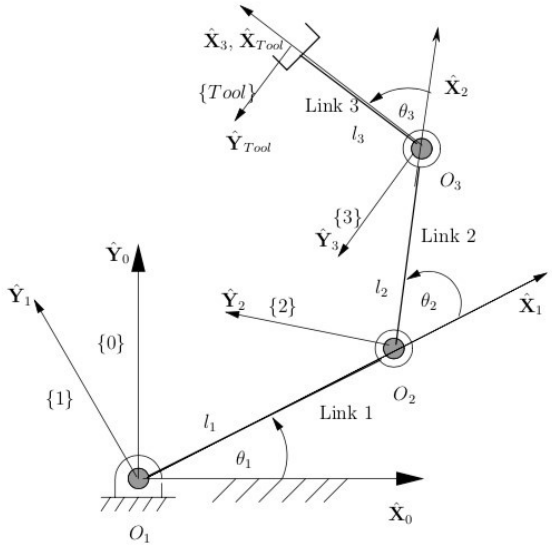


Fig. 6.4 The planar 3R manipulator

Now, one can write the position x, y and the orientation of the tool as

$$\begin{aligned} x &= l_1 c_1 + l_2 c_{12} + l_3 c_{123} \\ y &= l_1 s_1 + l_2 s_{12} + l_3 s_{123} \\ \phi &= \theta_1 + \theta_2 + \theta_3 \end{aligned} \tag{6.4}$$

Here, c(.) and s(.) represent cos(.) and sin(.) respectively.

(ii) **A SCARA manipulator:**
The SCARA manipulator is shown in Fig. 6.5.

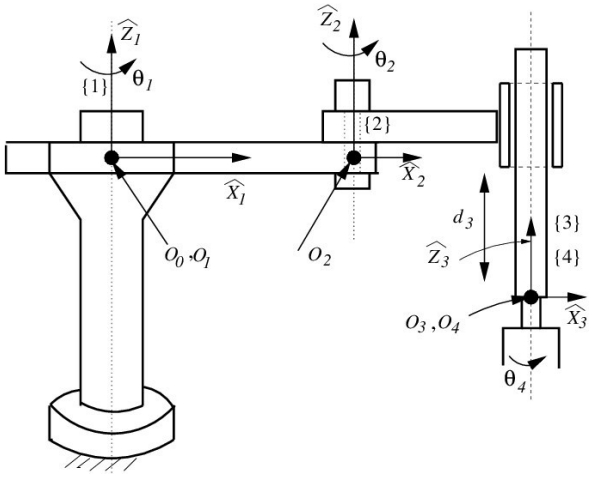


Fig. 6.5 A SCARA manipulator

For the SCARA manipulator, the matrix 0[T] is give as below:

$$^{0}_{4}[T] = {}^{0}_{1}[T]\ {}^{1}_{2}[T]\ {}^{2}_{3}[T]\ {}^{3}_{4}[T] \quad (6.5)$$

The orientation of the {4} can be described by the angle j, and the position (x,y,z) and the orientation can be given as

$$\begin{aligned} x &= a_1 c_1 + a_2 c_{12} \\ y &= a_1 s_1 + a_2 s_{12} \\ \phi &= \theta_1 + \theta_2 + \theta_4 \end{aligned} \quad (6.6)$$

It is noted that the direct kinematics problem for a serial manipulator is well defined, easily solvable and unique for any number of links. It means that as long as all the D-H parameters

are given, we can obtain 0[T] for and *n*.

n

Summary:

- Direct kinematics: Given D-H parameters, find position and orientation of end-effector.
- Direct kinematics problem can always be solved for any number of links.

- Direct kinematics can be solved using matrix multiplication.
- Direct kinematics in serial manipulators is unique.
- Direct kinematics problem for serial manipulators is the simplest problem

6.5 Inverse kinematics of a serial manipulators:

The inverse kinematics problem for serial manipulators can be stated as follows:

Given the link parameters and the position and orientation of {n} with respect to the fixed frame {0}, find the joint variables.

For the planar 3R manipulator model, the inverse kinematics problem would be to obtain $q_{i.}$(where i=1,2,3), given the position of the end effector x, y and its orientation j.

Here, we take n=6 for motion in 3D (n=3 for planar). Now, we obtain the inverse kinematics equations of a simple serial manipulators as below:

Following cases possible:

- n=6 for 3D motion or n=3 for planar motion → Same number of equations as unknowns.
- n<6 for 3D motion or n<3 for planar motion → Number of task space variables larger than number of equations and hence there must be (6−n) ((3−n) for planar) relationships involving the task space variables.
- n>6 for 3D motion or n>3 for planar motion → More unknowns than equations and hence infinite number of solutions — Redundant manipulators.

Here, we describe the simplest case of planar 3R manipulator.

The planar 3R manipulator:

The direct kinematics equations are given in Eqn. (6.4) where l_i is the link length for the 3R manipulator. The direct kinematics equations are given as

$$x = l_1c_1 + l_2c_{12} + l_3c_{123}$$
$$y = l_1s_1 + l_2s_{12} + l_3s_{123}$$
$$\phi = \theta_1 + \theta_2 + \theta_3$$

In inverse kinematics, given (x,y,j), obtain q_1, q_2 and q_3. Here, no general methods (as in linear equations) exists, hence, solution procedure depends on problem.

Inverse kinematics algorithm:

- Define X=x-l_3c_j and Y=y-l_3s_j-X and Y are known since x, y, j and l_3 are known.
- Squaring and adding

$$X^2 + Y^2 = l_1^2 + l_2^2 + 2l_1l_2c_2 \quad (6.7)$$

- From above equation

$$\theta_2 = \pm cos^{-1}\left(\frac{X^2+Y^2-l_1^2-l_2^2}{2l_1l_2}\right) \quad (6.8)$$

- Once q_2 is known

$$\theta_1 = atan2(Y, X) - atan2(k_2, k_1) \quad (6.9)$$

Where $k_2=l_2s_2$ and $k_1=l_1+l_2c_2$.

- Finally, q_3 is obtained from

$$q_3=j - q_1 - q_2 \quad (6.10)$$

6.6 3R manipulator workspace:

We define the workspace of the planar 3R manipulator as the set of values of {x, y, j} for which the inverse kinematics solution exists. Fig. 6.6 shows the 3D plot of the region in {x, y, j}.

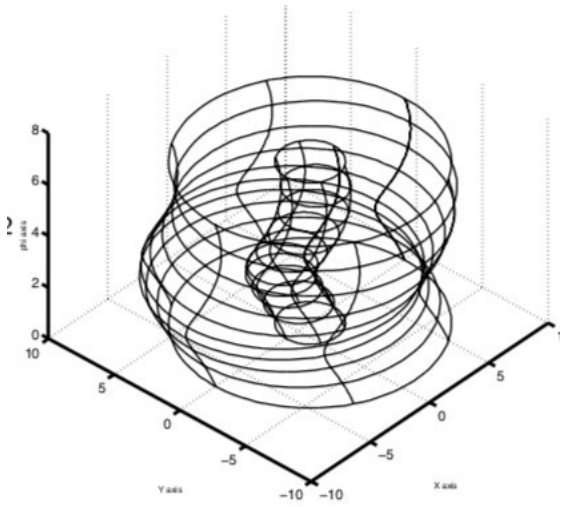


Fig. 6.6 Workspace of a planar 3R robot

From Eq. (6.8)

$$-1 \leq \left(\frac{X^2+Y^2-l_1^2-l_2^2}{2l_1l_2}\right) \leq +1 \tag{6.11}$$

or

$$(l_1 - l_2)^2 \leq (X^2 + Y^2) \leq (l_1 + l_2)^2 \tag{6.12}$$

where $X = x - l_3c_\phi$ and $Y = y - l_3s_\phi$

Fig. 6.6 satisfies the inequalities in Eq. (6.12). The projection of the workspace is shown in Fig. 6.7. It is seen that the maximum reach of the planar 3R manipulator are points on the circle of radius $l_1+l_2+l_3$ and the closest it can reach from the origin are points on the circle of radius $l_1-l_2-l_3$.

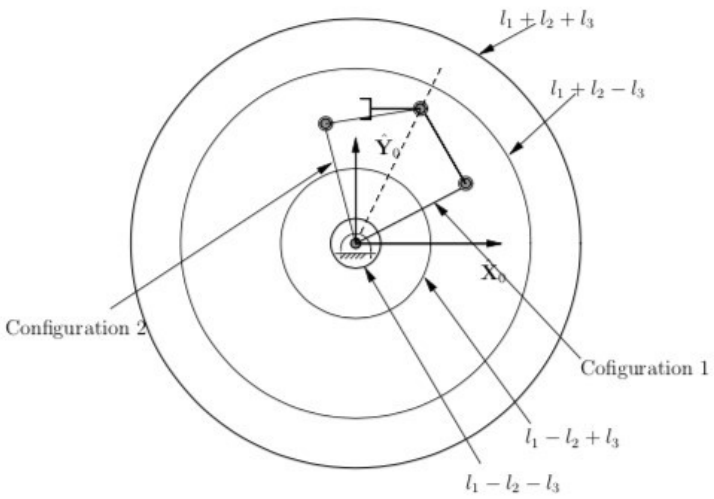


Fig. 6.7 Projection of workspace of a planar 3R robot

We observed for the Fig. 6.7 as below:

- Reachable workspace : All (x,y) between maximum reach (l_1+l_2+l_3) and minimum reach (l_1-l_2-l_3)
- Dexterous workspace:All (x,y) between radii (l_1+l_2-l_3) and (l_1-l_2+l_3).
- All points inside dexterous workspace can be reached with any j.
- As size of the end-effector l_3 increases, rechable workspace increases and dexterous workspace decreases.

Summary:

- Inverse kinematics: Given end-effector position and orientation and constant D-H parameters, obtain joint variables.
- Number of joint variables must be 3 (for planar motion) and 6 (for 3D motion).
- Involve solution of a set of non-linear (transcendental) equations — No general approach IK of serial manipulators.
- Planar 3R manipulator — IK can be solved easily, reachable and dexterous workspace.

6.7 Inverse kinematics of constrained and redundant robots:

In the case of a redundant manipulator, the number of joint variables is more than the number of equations. One such example is the planar 3R robot as shown in Fig. 6.4. Note that here, we are not interested the orientation of the last link. So, the (x,y) coordinate of the end-effector can be written as

$$\begin{aligned} x &= l_1 c_1 + l_2 c_{12} + l_3 c_{123} \\ y &= l_1 s_1 + l_2 s_{12} + l_3 s_{123} \end{aligned} \tag{6.13}$$

To obtain unique q_1, q_2 and q_3, we need another equation or constraint involving q_1, q_2 and q_3. To illustratr this, we use the example of 3R manipulator.

Now, the constraint optimization problem is set as below:

$$\begin{aligned} &Minimize f(\theta) = \theta_1^2 + \theta_2^2 + \theta_3^2 \\ &\text{subject to} \\ &g_1(\theta) = -x + l_1 c_1 + l_2 c_{12} + l_3 c_{123} \\ &g_2(\theta) = -x + l_1 s_1 + l_2 s_{12} + l_3 s_{123} \end{aligned} \tag{6.14}$$

Where q denotes the three joints variables (q_1, q_2, q_3).

The constraint optimization problem can be solved using Lagrange multipliers method. For the planar 3R manipulator, one can get

$$l_1 l_2 \theta_3 s_2 + l_2 l_3 (\theta_1 - \theta_2) s_3 + l_3 l_1 (\theta_3 - \theta_2) s_{23} = 0 \tag{6.15}$$

Eq. (6.15) can be solved numerically. Fig. 6.8 shows the plot of joint variables for redundant planar 3R robot.

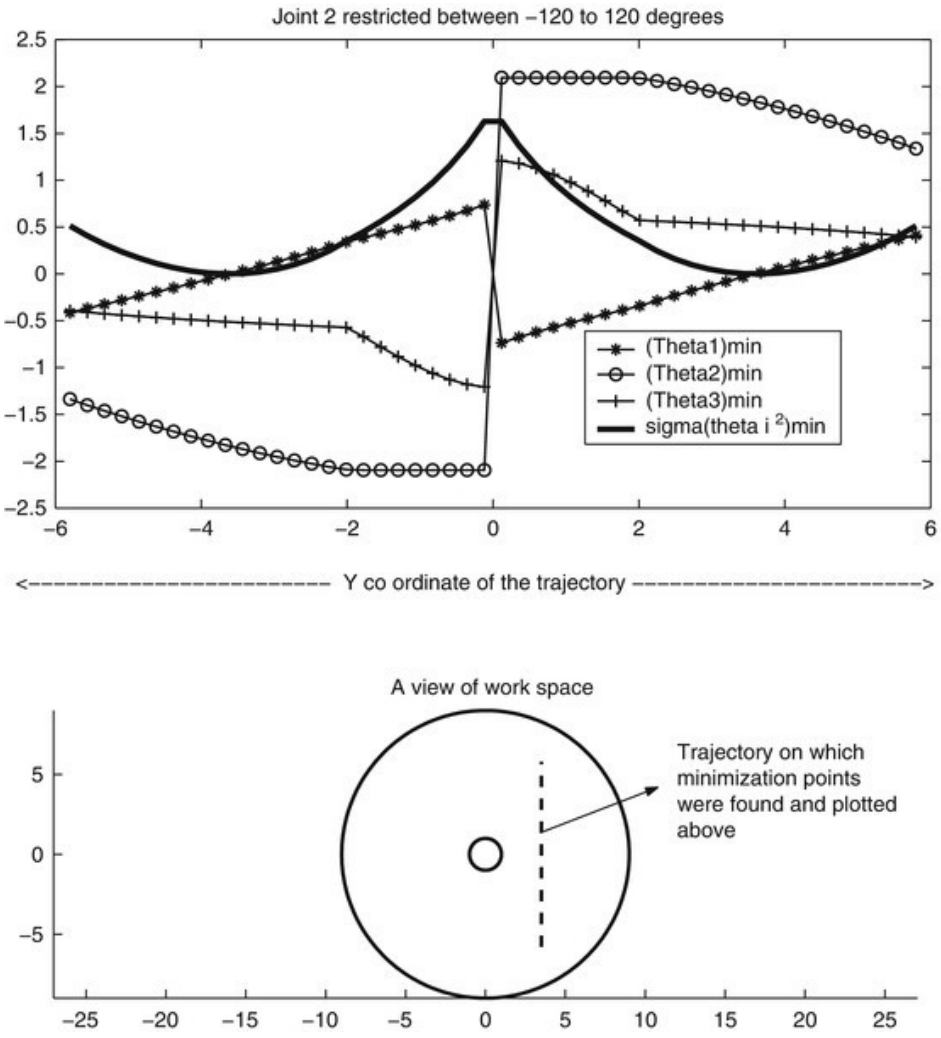


Fig. 6.8 joint variables for redundant planar 3R robot

6.8 Degrees of freedom of parallel mechanisms and manipulators:

The degrees of freedom of a parallel manipulator or a closed loop mechanism can be obtained from the Grübler criterion:

$$dof = \lambda(N - J - 1) + \sum_{i=1}^{J} F_i \qquad (6.16)$$

Parameters detail are discussed in section 6.2.

It indicates the number of independent actuators that can be used in a closed loop mechanism or a manipulator. In case of four bar mechanism, it is shown that l=3, N=4, J=4 and F_i=1. It means, only one out of the four rotary joints can be actuated.

6.9 Constraint and loop closure equations:

The configuration space of a closed loop mechanism is denoted by a vector of dimension (n+m) where the n are actuated and m are the additional constraint equations.Loop closure implies going around a loop in the closed loop mechanism shown in Fig. 6.9.

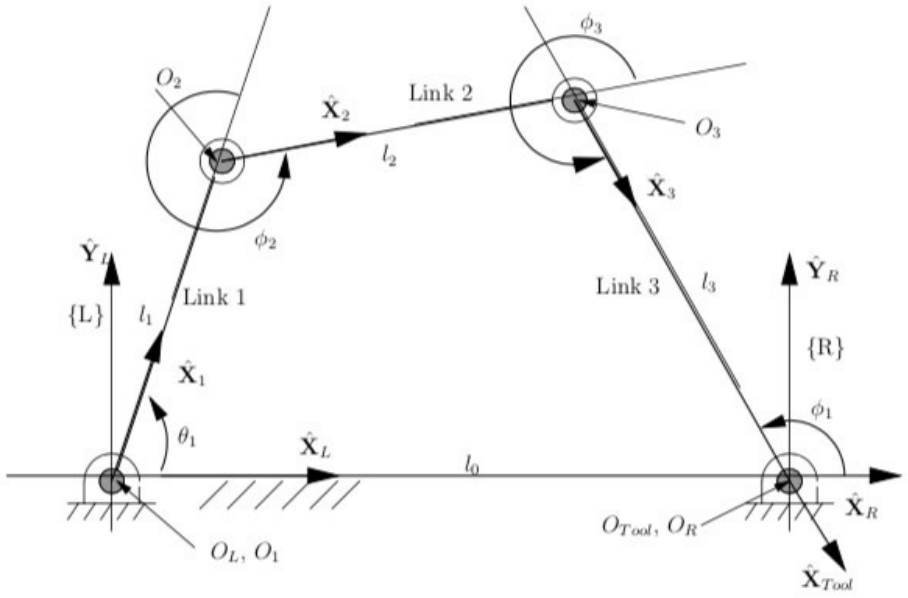


Fig. 6.9 The four bar mechanism

Here we have m passive joint variables and m independent equations required to solve for f for given n actuated variable, q_I, where i=1,2,…,n. The general approach to derive m loop closure constraint equations are as below:

- Break parallel manipulator into 2 or more serial manipulators
- Determine DH parameters for serial chains and obtain position and orientation of the Break for each chain.
- Use joint constraint at the Break(s) to rejoin (close) the parallel manipulator.

Note the following points to Break:

- The number of passive variables m is least

- Minimum number of constraint equations

One loop contains fixed frames {L} and {R} where {R} is translated by l_0 along the X axis. {1}, {2} ,{3} and {Tool} are as shown in Fig. 6.9. For convenient only X direction has been shown. The squence O_L-O_1-O_2-O_3-O_{Tool} can be thought of as a planar 3R manipulator.

The DH parameters of the planar 3R manipulator are shown below:

i	α_{i-1}	a_{i-1}	d_i	θ_i
1	0	0	0	θ_1
2	0	l_1	0	ϕ_2
3	0	l_2	0	ϕ_3

From the DH table estimate $^{3}_{0}[T]$. For planar 3R and tool of length l_3, estimate $^{Tool}_{3}[T]$.

The $^{Tool}_{R}[T]$ is given as

$$^{Tool}_{R}[T] = \begin{pmatrix} -\cos\phi_1 & -\sin\phi_1 & 0 & 0 \\ \sin\phi_1 & -\cos\phi_1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad (6.17)$$

The loop closure equations for the four bar mechanism is

$$^{L}_{1}[T]^{1}_{2}[T]^{2}_{3}[T]^{3}_{Tool}[T]^{Tool}_{R}[T] = ^{L}_{R}[T] \quad (6.18)$$

Since the loop is planar only X and Y components of the position vector and the rotation about the Z axis are useful. So we can write

$$\begin{aligned} l_1\cos\theta_1 + l_2\cos(\theta_1+\phi_2) + l_3\cos(\theta_1+\phi_2+\phi_3) &= l_0 \\ l_1\sin\theta_1 + l_2\sin(\theta_1+\phi_2) + l_3\sin(\theta_1+\phi_2+\phi_3) &= 0 \\ \theta_1+\phi_2+\phi_3+(\pi-\phi_1) &= 4\pi \end{aligned} \quad (6.19)$$

Here the last equation is derived based on the assumption that {L} is parallel to {R}.

Direct Kinematics Problem:

There are two part problem in direct kinematics. They are

(i) Given the geometry of the manipulator and the actuated joint variables, obtain passive joint variables

(ii) Obtain position and orientation of a chosen output link.

The first part, for most closed loop mechanisms and parallel manipulators, has a complexity similar to the inverse kinematics problem in serial manipulators. It involves determining the m loop closure constraint equations and the solution of non linear equations for the m passive variablesin terms of the n actuated variables.

It is much harder than DK problem for a serial manipulator. It leads to the notion of mobility and assembe-ability of a parallel manipulator or a closed loop mechanism.

6.10 Mobility of parallel manipulators:

In case of serial manipulator, one can define the concept of workspace as the position and orientations. For parallel manipulators and closed loop mechanisms, importance is given to notion of limits on the possible range of motion of the joints present in the closed loop mechanism or the parallel manipulator. So in parallel manipulator both mobility and workspace concepts is used. Workspace dependent on the choice of the output link whereas mobility deals with range of possible motion of the actuated joints in a parallel manipulator. So mobility is more important in parallel manipulators. Mobility is determined by geometry/linkage dimensions that uses loop-closure constraint equations. Mobility is related to the ability to assemble a parallel manipulator at a configuration. All values of actuated variables such that real value(s) of passive variables exists is determined by direct kinematics. Very few parallel manipulators where the direct kinematics can be reduced to the solution of a univariate polynomial of degree 4 or less. In most cases mobility determined numerically using search.
In 4-bar mechanism, mobility can be obtained in closed-form.

MOBILITY OF 4-BAR MECHANISM

Loop-closure constraint equation of a 4-bar is given as

$$\eta_1(\theta_1, \phi_1) = (l_1 \cos\theta_1 - l_0 - l_3 \cos\phi_1)^2 + (l_1 \sin\theta_1 - l_3 \sin\phi_1)^2 - l_2^2 = 0 \qquad (6.20)$$

On simplification $h_1$becomes

$$P\cos\phi_1 + Q\sin\phi_1 + R = 0 \tag{6.21}$$

Where P, O and R are given by

$$\begin{aligned} P &= 2l_0l_3 - 2l_1l_3c_1,\ Q = -2l_1l_3s_1 \\ R &= l_0^2 + l_1^2 + l_3^2 - l_2^2 - 2l_0l_1c_1 \end{aligned} \tag{6.22}$$

Using tangent half-angle substitutions

$$\phi_1 = 2\tan^{-1}\left(\frac{-Q \pm \sqrt{P^2+Q^2-R^2}}{R-P}\right) \tag{6.23}$$

For real f_1, $P^2+Q^2-R^2 \geq 0$
Limiting case: $P^2+Q^2-R^2 = 0$

In the limiting case, the bounds on q_1 are

$$c_1 = \frac{l_0^2 + l_1^2 - l_3^2 - l_2^2 \pm 2l_3l_2}{2l_0l_1} \tag{6.24}$$

For full rotatability of q_1, one cannot have any bounds. The condition $c_1>1$ and $c_1<-1$ leads to

$$(l_0 - l_1)^2 > (l_3 - l_2)^2 \tag{6.25}$$

$$(l_0 + l_1) < (l_3 + l_2) \tag{6.26}$$

Two additional conditions from $c_1>1$, $c_1<-1$ lead to $l_3+l_2+l_1 < l_0$ and $l_0+l_1+l_2 < l_3$, which violets the triangle inequality.

Eqn. (6.25) gives rise to four inequalities

$$\begin{aligned} l_0 - l_1 &> l_3 - l_2 \\ l_0 - l_1 &> l_2 - l_3 \\ l_1 - l_0 &> l_3 - l_2 \\ l_1 - l_0 &> l_2 - l_3 \end{aligned} \tag{6.27}$$

For the case of $l_1 < l_0$

$$\begin{aligned} l_0 + l_2 &> l_1 + l_3 \\ l_0 + l_3 &> l_1 + l_2 \end{aligned} \tag{6.28}$$

Equation (6.26) and (6.28) imply that l_0, l_2 and l_3 are all larger than l_1.

Likewise, for $l_1 > l_0$

$$l_1 + l_2 > l_0 + l_3$$
$$l_1 + l_3 > l_0 + l_2 \quad (6.29)$$

Hence, l_0 is the shortest link in this case and either of l_1, l_2 and l_3 must be largest link. So for the joint q_1 to rotate fully, we must have the condition that the sum of the shortest and the largest link must be less than the sum of the two intermediate links.

Exercises:

1. What is a manipulator? Differentiate between serial and paralle manipulator.
2. What is degree of freedom? How it is used separate spatial and planar manipulators.
3. Deduce the loop closure contraint equation.
4. Represent the direct kinematics of a serial manipulator in terms of Base as reference ponit.
5. Find the position of a planar 3R manipulator.
6. What is an inverse kinematics of a serial manipulators?
7. Write down the inverse kinematics algorithm.
8. What is the purpose of a workspace?
9. Explain 3R manipulator workspace.
10. What do mean by redundant robot?
11. Deduce the loop closure equations.
12. Explain the direct kinematics problem.

References:

1. Robotics-Fundamental concepts and analysis by Ashitava Ghosal, Oxford University Press.
2. Robotics technology and flexible automation by S R Deb, TMH.

Note: In module-6 all the concepts, figures and equations has been taken from reference-1.

Module – 7

Motion planning and Control

7.1 Introduction:

In this module we deal with algorithms to plan and generate trajectories which describe the motion of a robot. Trajectory planning and control is an essential part in robotics. Here, the

trajectory means the position, velocity and acceleration of either actuated joints or end effector. Most of the time the start and final position of joint is mentioned by the operator. Sometimes the desired velocities or acceleration has been provided. The robot control machine and the motion planning algorithm should generate the exact path, velocity and acceleration beteen the start, intermediate and end points. Now, if someone has planned to store all the intermediate position of a robot from start and end, it may be raised an issue of storage, as it requires large memory. So, it needs an efficient way to handle these huge data. Most of the time the desired motion points are calculated by certain rate which is typically 50 to 200 Hz. Now a days, advanec label computers with fast processor and memory can handle this issue successfully.

There are two schemes to design trajectory planning and generation. One is joint space schemes and the other is cartesian schemes. Here we discuss one after another.

7.2 Joint space trajectory planning and generation:

Let us consider an initial point $q_i(t_0)$ and a final point $q_i(t_f)$ where t_0 and t_f indicates the initial and final time of the trajectory. Here, our aim is to define a smooth trajectories between these two points. There are many ways to solve this problem. For, simplicilty, we go for a polynomial curves. The simplest possible polynomial curve can be written as

$$\theta_i(t) = \frac{\theta_i(t_f)-\theta_i(t_0)}{t_f-t_0}(t-t_f) + \theta_i(t_f) \tag{7.1}$$

Now there are several ways to define the intermediate points as shown in Fig. 7.1. Here, we get a piece wise linear curve. The second plot in Fig. 7.1 shows the joint velocity where it is shown that there is a change in sign in steps between two segments. The third plot shows the acceleration plot where we get the spikes in each changes.

A polynomial can be used as below:

$$\theta_i(t) = a_0 + a_1 t + a_2 t^2 + a_3 t^3 \tag{7.2}$$

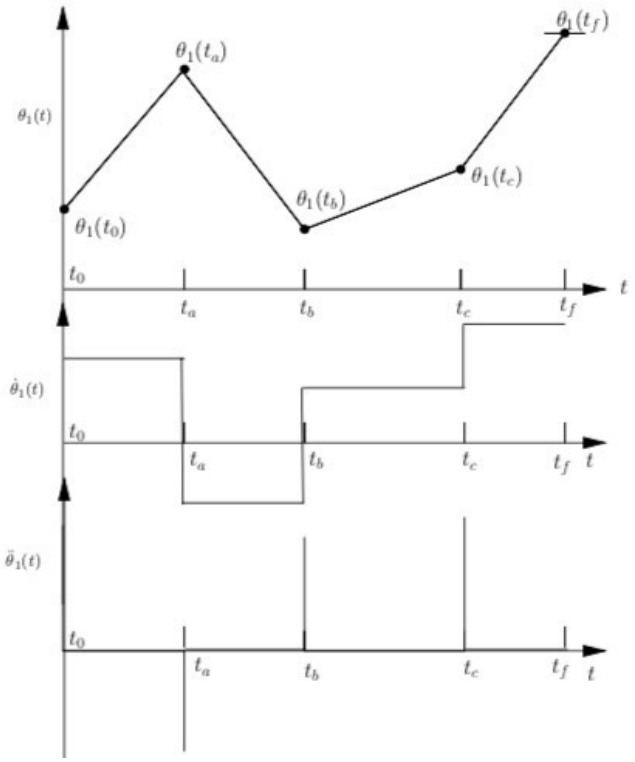


Fig 7.1 Piece-wise linear joint trajectory

Where, a_0, a_1, a_2 and a_3 are four constant coefficients. Let the initial time is $t=t_0$ and the final time $t=t_f$. So

$$\theta_i(t_0) = \theta_i(0)$$
$$\theta_i(t_f) = \theta_i(f) \tag{7.3}$$

The velocity at iniial and final times are

$$\dot{\theta}_i(t_0) = \dot{\theta}_i(0)$$
$$\dot{\theta}_i(t_f) = \dot{\theta}_i(f) \tag{7.4}$$

The velocity and acceleration can be obtain by differentiating Eq. (7.2) as below:

$$\dot{\theta}_i(t) = a_1 + 2a_2t + 3a_3t^2$$
$$\ddot{\theta}_i(t) = 2a_2 + 6a_3t \tag{7.5}$$

Here, we assume that $t_0=0$ and using Eq, (7.3) and (7.4) in (7.2) and (7.5), we get

$$\theta_i(0) = a_0$$
$$\theta_i(f) = a_0 + a_1t_f + a_2t_f^2 + a_3t_f^3$$
$$\dot{\theta}_i(0) = a_1$$
$$\dot{\theta}_i(f) = a_1 + 2a_2t_f + 3a_3t_f^2 \tag{7.6}$$

Solving the above equation, we get

$$
\begin{aligned}
a_0 &= \theta_i(0) \\
a_1 &= \dot{\theta}_i(0) \\
a_2 &= \tfrac{3}{t_f^2}[\theta_i(f) - \theta_i(0)] - \tfrac{2}{t_f}\dot{\theta}_i(0) - \tfrac{1}{t_f}\dot{\theta}_i(f) \\
a_3 &= -\tfrac{2}{t_f^3}[\theta_i(f) - \theta_i(0)] + \tfrac{1}{t_f^2}\dot{\theta}_i(0) + \tfrac{1}{t_f}\dot{\theta}_i(f)
\end{aligned}
\tag{7.7}
$$

Finally, the Eq. (7.7) give us the coefficients for the desired polynomial which connecting the iniial and final q_i and q_f respectively.

7.3 Cartesian space trajectory planning and generation:

In joint space scheme, it is useful fo a single joint or a group of joints that are moving between two positions. But, it is dificult to visualize the motion at all joints. So, it is more convenient to represent the traectoy in cartesian space. Note that, this scheme is computationally expensive as it involves the computation of inverse kinematics of the manipulator. Any cartesian path can be approximated by either linear or circular segments. Here, we discuss the linear interpolation only.

Let us consider a cartesian straight line path for the end effector of a robot from an initial point $(x_0, y_0, z_0)^T$ at t=0 to a final point $(x_f, y_f, z_f)^T$ at time t_f. The equation of a straight line in 3D cartesian space can be written as

$$
\frac{x(t)-x_f}{x_f-x_0} = \frac{y(t)-y_f}{y_f-y_0} = \frac{z(t)-z_f}{z_f-z_0} \tag{7.8}
$$

Or we can write it as

$$
\begin{aligned}
y(t) &= \left(\tfrac{y_f-y_0}{x_f-x_0}\right)(x(t) - x_f) + y_f \\
z(t) &= \left(\tfrac{z_f-z_0}{x_f-x_0}\right)(x(t) - x_f) + z_f
\end{aligned}
\tag{7.9}
$$

So, first we obtain x(t) and then using Eq. (7.9) we can calculate y(t) and z(t) as well. Hence we are able to plan a smooth trajectory for x(t) using a polynomial as below:

$$x(t) = a_0 + a_1 t + a_2 t^2 + a_3 t^3 \tag{7.10}$$

Here, a_0, a_1, a_2 and a_3 are the constant coefficients. To estimate the coefficients, we can follow the same approach as used in the joint trajectory and the corresponding coefficiemts are

$$
\begin{aligned}
a_0 &= x_0 \\
a_1 &= \dot{x}_0 \\
a_2 &= \frac{3}{t_f^2}(x_f - x_0) - \frac{2}{t_f}\dot{x}_0 - \frac{1}{t_f}\dot{x}_f \\
a_3 &= -\frac{2}{t_f^3}(x_f - x_0) + \frac{1}{t_f^2}(\dot{x}_0 + \dot{x}_f)
\end{aligned}
\tag{7.11}
$$

Since, x(t), y(t) and z(t) satisfy the equation of a straight line, all trajectory points lie on a straight line.

7.4 Feedback control of a single link manipulator:

Fig. (7.2) shows the model of a single link containing a DC servo motor through a gear. The rated speed of a typical DC motor is 2000 rpm and more, but here the desired rpm is around 60. So we need a large speed reduction.

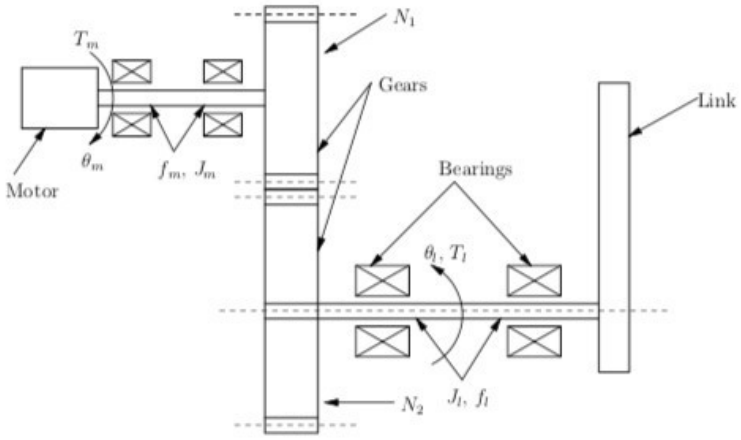


Fig. 7.2 Model of a single link

Let us assume two spur gears that provide required speed reduction. The link rotation q_l related to motor rotation q_m is given by

$$
\frac{\theta_l}{\theta_m} = n \tag{7.12}
$$

So, the one degree of freedom system is

$$
\theta_l = n\theta_m, \quad \dot{\theta}_l = n\dot{\theta}_m, \quad \ddot{\theta}_l = n\ddot{\theta}_m \tag{7.13}
$$

The equation of a motion of gear-1 can be written as

$$
J_m\ddot{\theta}_m + f_m\dot{\theta}_m + T_1 = T_m \tag{7.14}
$$

Where J_m, f_l and T_1 are the inertia of load. T_2 denotes the torque transmitted to gear2 by gear1. Assuming no energy loss at gear tooth contacts, we can write

$$T_1\theta_m = T_2\theta_l \tag{7.15}$$

Therefore, the equation of motion for a system is

$$(J_m + n^2 J_l)\ddot{\theta}_m + (f_m + n^2 f_l)\dot{\theta}_m = T_m + nT_l \tag{7.16}$$

Here, n is small (around 0.01). Effect of T_1 is also reduced by a factor of n. The Fig. (7.3) shows the mechanism of a DC servo.

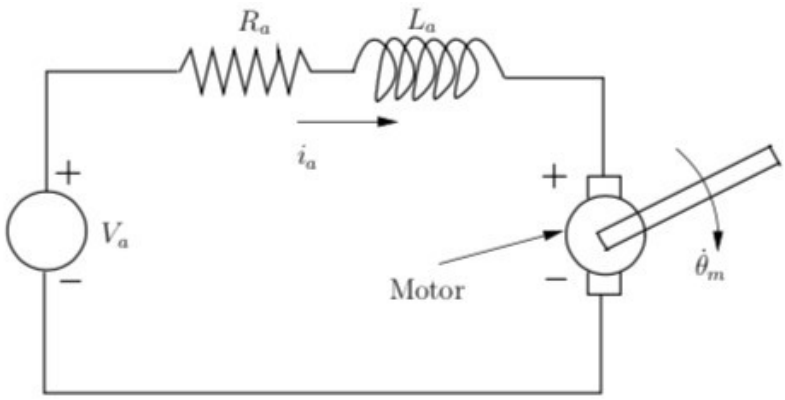


Fig. 7.3 Permamnent magnet DC servo motor

The torque generated by the motor

$$T_m = K_t i_a \tag{7.17}$$

Back emf generated by coil is

$$V = K_g \dot{\theta}_m \tag{7.18}$$

Here, K_t and K_g are the torque and back emf constant. The dynamics of a motor is given as

$$L_a \dot{i}_a + R_a i_a + K_g \dot{\theta}_m = V_a \tag{7.19}$$

For small DC srvo motors, L_a is small and can be ignored. Combining equations of motion and the dynamics of motor with L_a=0, one can write

$$(J_m + n^2 J_l)\ddot{\theta}_m + (f_m + n^2 f_l)\dot{\theta}_m = K_t\left(\frac{V_a - K_g\dot{\theta}_m}{R_a}\right) + nT_l \tag{7.20}$$

In a compact form

$$J\dot{\Omega} + F\Omega = KV_a + T_d \tag{7.21}$$

Where

$$K = K_t/R_a, \quad F = (f_m + n^2 f_l) + K_t K_g/R_a$$
$$J = J_m + n^2 J_l, \quad T_d = nT_l, \quad \Omega = \dot{\theta}_m$$

Eqn. (7.21) describes the mechatronic behavior of the single link manipulator.

Applying Laplace transform in Eqn. (7.21),

$$Js\Omega(s) + F\Omega(s) = KV_a(s) + T_d(s) \tag{7.22}$$

The transfer function of the system is

$$\frac{\Omega(s)}{V_a(s)} = \frac{K}{Js+F}, \quad \frac{\Omega(s)}{T_d(s)} = \frac{1}{Js+F} \tag{7.23}$$

Where W(s) is the output and $V_a(s)$ and $T_d(s)$ are the inputs. So, the system can be shown in block diagram as in Fig. (7.4).

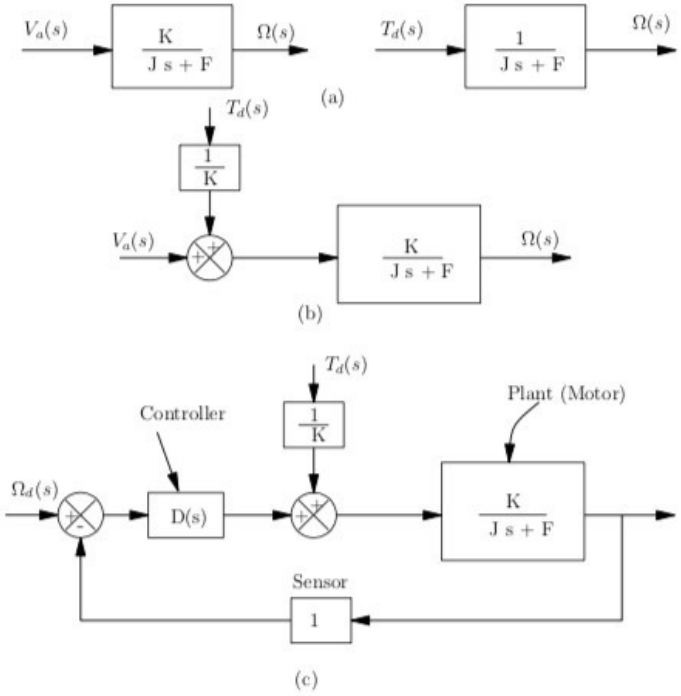


Fig. 7.4 Transfer function of asingle link manipulator

Fig. (7.5) shows the representation of a first oredr system where T_d=0. It is known as first order because the Eq. (7.21) is first order differential equation.

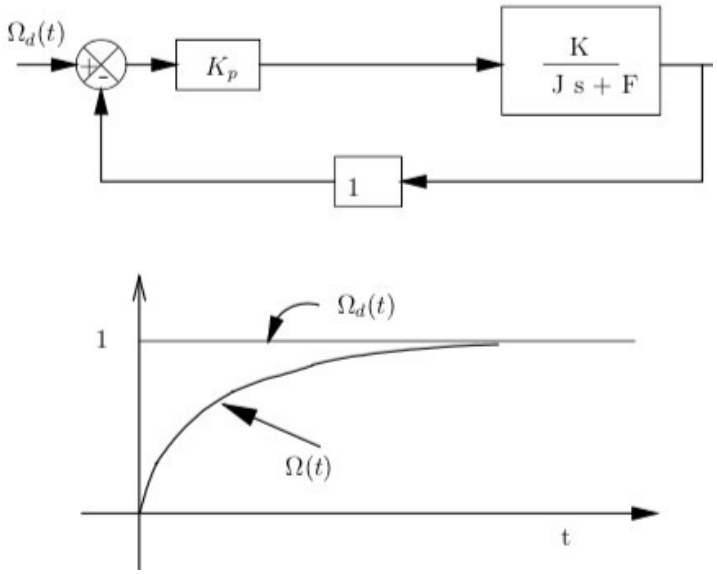


Fig. 7.5 First order system and its step response

Again the open loop transfer function with T_d=0 is given as

$$\frac{\theta(s)}{V_a(s)} = \frac{K}{s(Js+F)} \tag{7.24}$$

The transfer function os called second order as the polynomial in the deomenator is of degree 2. The closed loop diagram of the system is shwn in Fig. 7.6.

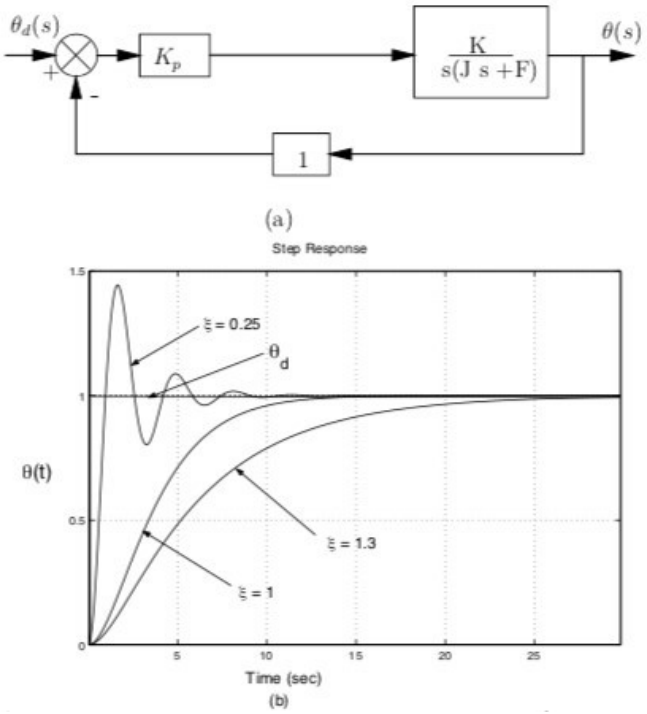


Fig. 7.6 Second order system and its step response

The closed loop transfer function can be written as

$$\frac{\theta(s)}{\theta_d(s)} = \frac{KK_p}{s(Js+F)+KK_p} = \frac{\omega_n^2}{s^2+2\xi\omega_n s+\omega_n^2} \tag{7.25}$$

where $\omega_n^2 = (KK_p/J)$, $F/J = 2\xi\omega_n$ and $\xi = \frac{F}{2\sqrt{JKK_p}}$.

7.5 PID control of a single link manipulator:

Fig. 7.7 shows the closed loop feedback control system for a single link robot.

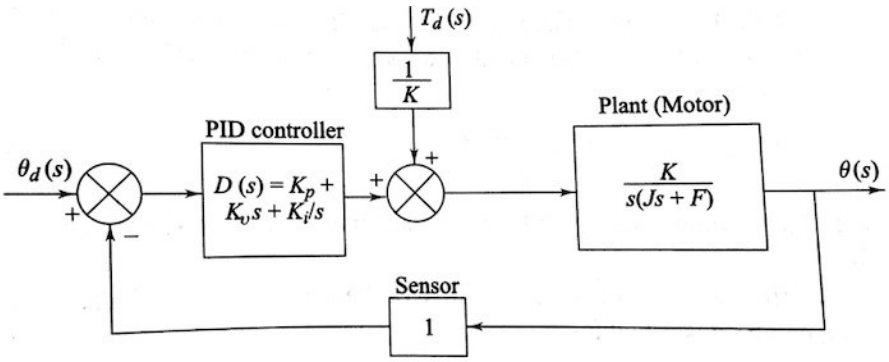


Fig. 7.7 PID control of asingle link manipulator

The controller transfer function $D(s)=K_p+K_vs$ where K_v is the derivative gain. The closed loop transfer function is given as

$$\frac{\theta(s)}{\theta_d(s)} = \frac{KK_p+sKK_v}{Js^2+s(F+KK_v)+KK_p} \tag{7.26}$$

Now, to decrease the steady state error, integral term (K_i/s) is used where K_i is called the controller gain and Kp is called proportional gain. Note that large K_i leads to an unstable system. Thus we can write

$$D(s) = K_p + \frac{K_i}{s} + \frac{K_v s}{1+T_v s} \tag{7.27}$$

T_v is a time constant and $s/(1+T_vs)$ represents a filter.

In time domain, one can write

$$V_a(t) = K_p e(t) + K_v \dot{e}(t) + K_i \int_0^t e(\tau) d\tau. \tag{7.28}$$

Most of the time, feed forward term added for improved trajectory tracking. So the updated PID controller is

$$V_a(t) = \ddot{\theta}_d(t) + K_p e(t) + K_v \dot{e}(t) + K_i \int_0^t e(\tau) d\tau \tag{7.29}$$

7.6 PID control of a multi link manipulator:

Note that, the Multi-link manipulator, unlike single link, is a non linear system. So the linear control scheme is not applicable here. It is not possible to analyse exact the performance of a PD (or PID) scheme for a non linear system. However, in most industrial robot, PD (or PID) works sufficiently well and thus applied on most industrial robots. Here we go through the PD scheme to control multi-link manipulators.

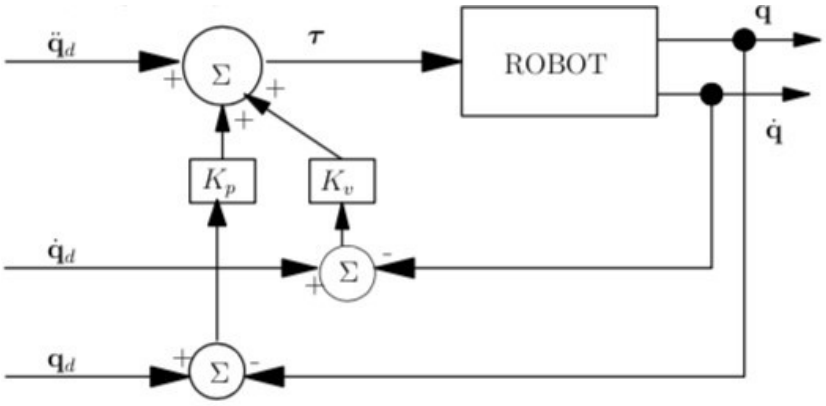


Fig. 7.8 PD control of a multi-link robot

The n´1 output of the controller is calculated as

$$V_a(t) = \ddot{q}_d(t) + K_v \dot{e}(t) + K_p e(t), \tag{7.30}$$

where K_p and K_v are n´n positive definite proportional and derivative gain respectively. The n´1 error vector between the desired joint rotation $\mathbf{q}_d(t)$ and the measured actual joint rotation $\mathbf{q}(t)$ is given as

$$e(t) = q_d(t) - q(t) \tag{7.31}$$

Fig. (7.8) shows the block diagram of the PD control scheme for a multi-link manipulator. The output of the controller is t(t) which is a n´1 vector of joint torques. In commonly used control

system of single-input/single-output systems, the desired trajectory is a single input $q_d(t)$ and its derivative is either not available or not used. For an advanced robot , the trajectory planner generates $q_d(t)$, its first order and second order derivative for all joints and so one can use them as shown in Fig. (7.8).

7.7 Non linear control of manipulators:

Till now we are dealing with single and multi-link manipulators which are the linear systems and the inertia of the links were constant. Note that a linear control system can not be used to describe non linear systemas it gives non uniform performance at every point in the workspace of the robot. Here we are going to deal with non linear controllers which cancel out the non linearties by reducing non linear equations to the set of decoupled linear equations.

The dynamic equations of motion of an n-DOF robot can be written as

$$\tau = [\mathbf{M}(\mathbf{q})]\ddot{\mathbf{q}} + \mathbf{C}(\mathbf{q},\dot{\mathbf{q}}) + \mathbf{G}(\mathbf{q}) + \mathbf{F}(\mathbf{q},\dot{\mathbf{q}}) \tag{7.32}$$

where,

$[\mathbf{M}(\mathbf{q})]$ is an $n \times n$ mass matrix and $\mathbf{C}(\mathbf{q},\dot{\mathbf{q}})$, $\mathbf{G}(\mathbf{q})$, and $\mathbf{F}(\mathbf{q},\dot{\mathbf{q}})$ are $n \times 1$ vectors representing Coriolis/centripetal, gravity, and friction terms, respectively.

One ca n write n´1 vector t of joint torques as

$$\tau = [\alpha]\tau' + \beta \tag{7.33}$$

We select

$$[\alpha] = [\mathbf{M}(\mathbf{q})], \quad \beta = \mathbf{C}(\mathbf{q},\dot{\mathbf{q}}) + \mathbf{G}(\mathbf{q}) + \mathbf{F}(\mathbf{q},\dot{\mathbf{q}}) \tag{7.34}$$

Substituting Eq. (7.33) and (7.34) in (7.32), we get

$$\tau' = \ddot{\mathbf{q}} \tag{7.35}$$

The Eq. (7.35) represents a unit inertia with input t'. All non-linearities and coupling are cancelled and original non-linear equations transformed to n decoupled linear equations.

If t' is chosen as

$$\tau' = \ddot{\mathbf{q}}_d(t) + [K_v]\dot{\mathbf{e}}(t) + [K_p]\mathbf{e}(t) \tag{7.36}$$

with e(t)=$\mathbf{q}_d$-**q,** then a linear error equation for the unit inertia system as

$$\ddot{\mathbf{e}}(t) + [K_v]\dot{\mathbf{e}}(t) + [K_p]\mathbf{e}(t) = \mathbf{0} \tag{7.37}$$

We choose positive definite, diagonal matrices [K_p] and [K_v], we will get a linear decoupled error equation and we can set [K_p] and [K_v] to get critically damped performance at every point in the workspace of the robot.

Exercises:

1. Consider a non-linear dynamical system

$$\ddot{x} + 7\dot{x}^2 + x\dot{x} + x^3 = u(t)$$

where u(t) is the control input. Design a control system such that the error response is critically damped and the natural frequency w_n is 1 rad/sec. Draw a block diagram of the system.

2. For a single link manipulator, as discussed in (7.4), select J=K=F=1, K_p=1 and $q_d(t)$ as a step input. Vary K_v between 1 and 3 and numerically obtain the plots of q(t).

3. For a single link manipulator, as discussed in (7.4), select J=K=F=1, K_p=1 and K_v=2. Assume T_d=0.1. Plot q(t).
4. What is the difference between single-link and multi-link manipulator?
5. What is the purpose of joint and cartesian space?
6. What is the advantage of a feedback in control system?
7. Distinguish between first order and second order system by their performance.
8. Explain the application of PD and PID controller.
9. Obtain the coefficients of a cubic polynomial

$$x(t) = a_0 + a_1 t + a_2 t^2 + a_3 t^3$$

if $\theta(0), \dot{\theta}(0), \ddot{\theta}(0)$ and $\theta(t_f)$ are specified

10. Find the step response of a closed loop system.

References:

1. Robotics-Fundamental concepts and analysis by Ashitava Ghosal, Oxford University Press.
2. Robotics technology and flexible automation by S R Deb, TMH.

Note: In module-7 all the concepts, figures and equations has been taken from reference-1.