**3.1. Reasoning under Uncertainty**

Conventional Reasoning systems such as FOPL are designed to work with information that has 3 important properties:

* Complete: nothing is missing.
* Consistent: no contradictory facts.
* Monotonicity: nothing will ever be retracted from the set of facts that are known to be true.

For reasoning with information which do not have one or more of these properties, we have to find some other mechanisms: Non-Monotonic Reasoning and Statistical Reasoning.

**3.1.1. Non-Monotonic Reasoning**

Non-Monotonic Reasoning is one of the methods for reasoning under uncertainty. In this method, we can use either the minimalist approach or the default approach.

1. Minimalist Reasoning
   * Closed World Assumption (CWA)
   * Circumscription
2. Default Reasoning
   * Non-Monotonic Logic
   * Default Logic

**Closed World Assumption (CWA)**

CWA says that “only objects that satisfy any predicate P are those that must”.

For example, an employee database can safely assumed to list all of the company’s employees. If someone asks whether Smith works for a company, we should reply “no” unless he is explicitly listed as an employee.

**Circumscription**

Given, A(Joe) ∨ B(Joe).

If we circumscribe only A, then this assertion describes exactly those models in which A is true of no one and B is true of at least Joe.

If we circumscribe only B, then this assertion describes exactly those models in which B is true of no one and A is true of at least Joe.

If we circumscribe A and B together, then we will admit only those models in which A is true of only Joe and B is true of no one or those in which B is true of only Joe and A is true of no one.

In default reasoning, we consider some given default value to be true unless there is some explicit statement which opposes the default value.

**Default Logic**

In this approach, we allow inference rules of the form

Such a rule should be read as “If A is provable and it is consistent to assume B then conclude C”.

Example

**Non-Monotonic Logic (NML)**

In NML, the language of FOPL is augmented with the modal operator M, which can be read as “is consistent”.

∀x∀y: related(x,y) ∧ **M** get\_along(x,y) → will\_defend(x,y)

This should be read as “For all x and y, if x and y are related and if the fact that x gets along y is consistent with everything else that is believed, then conclude that x will defend y”.

**Inheritance**

We can write inheritable knowledge as rules in default logic

This assigns a default value to height of basketball players.

**Deduction**

∀x: A(x)→B(x)

Given A(c), we conclude B(c).

**Abduction**

∀x: A(x)→B(x)

Given B(c), we conclude A(c).

Abduction is not licensed by the rules of standard logic and it may be wrong.

**3.1.2. Statistical Reasoning**

Statistical Reasoning is another approach for reasoning under uncertainty where we use the concepts of probability and conditional probabilities.

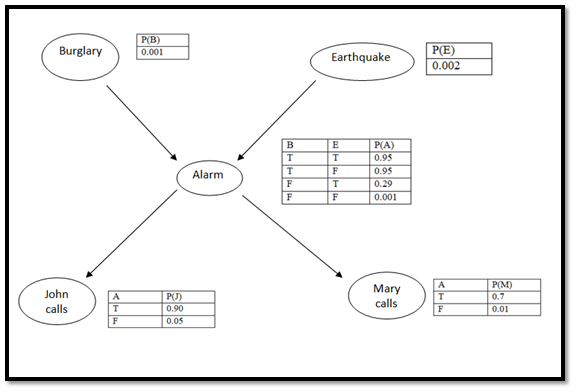
* Bayesian Reasoning
* Dempster-Shafer Theory
* Fuzzy Logic

1. Bayes Theorem

Consider a scenario where a patient has high fever and goes to a doctor where doctor identifies the most probable disease from which the patient is suffering. Here, fever is the evidence (E) and the disease diagnosed by the doctor is the hypothesis (H).

**Bayesian Network**

A Bayesian network is a probabilistic graphical model for representing knowledge about an uncertain domain where each node corresponds to a random variable and each edge represents the conditional probability for the corresponding random variables.



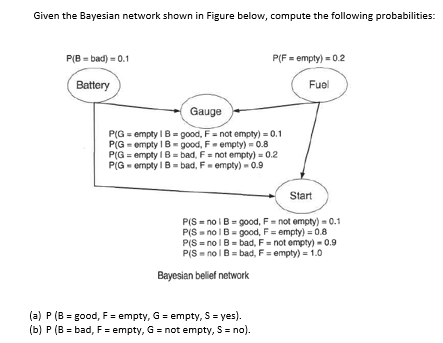
Consider the following set of propositions:

* Patient has spots
* Patient has measles
* Patient has high fever
* Patient has Rocky Mountain Spotted Fever
* Patient has previously been inoculated against measles
* Patient was recently bitten by a tick
* Patient has an allergy

1. Create a network that defines the causal connections among these nodes.
2. Make it a Bayesian network by constructing the necessary conditional probability matrix.







ANSWER

P(B=good, F=empty, G=empty, S=yes) =

P(B=good)\* P(F=empty) \* P(G=empty|B=good, F=empty) \* P(S=yes|B=good,F=empty) =

0.9 \* 0.2 \* 0.8 \* 0.2 = 0.0288

P(B=bad, F=empty, G=not empty, S=no) =

P(B=bad)\* P(F=empty) \* P(G= not empty|B=bad, F=empty) \* P(S=no|B=bad,F=empty) =

0.1 \* 0.2 \* 0.1 \* 1.0 = 0.002

**Dempster-Shafer Theory**

This approach considers a set of propositions and assigns them an interval [Belief, Plausibility].

Belief measures the strength of the evidence in favor of a set of propositions, ranging from 0 to 1.

Plausibility is defined as Pl(s) = 1- Belief(not s)

We start with an exhaustive universe of mutually exclusive hypothesis called as “the frame of discernment”, and represent it as .

For example, let = {Allergy, Flu, Cold, Pneumonia}

The key function we use a probability density function, which we denote as “m”.

Suppose, we acquire a piece of evidence that suggests with probability of 0.6 that the correct diagnosis is in the set {Flu, Cold, Pneumonia}. We update m as follows:

{Flu, Cold, Pneumonia} (0.6)

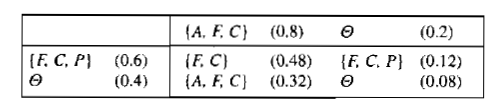
{} (0.4)

Suppose we are given two belief functions m1 and m2, where m1 corresponds to our belief after observing fever, and m2 corresponds to our belief after observing a runny nose.

m1: {Flu, Cold, Pneumonia} (0.6), {} (0.4)

m2: {Allergy, Flu, Cold} (0.8), {} (0.4)

Then we can compute their combination as



**Fuzzy Logic**

**Crisp set**: each element has full membership e.g. set of vowels

**Fuzzy set**: each element has partial membership.

In fuzzy set, each element is assigned a membership degree from 0-1.

Each element of a fuzzy set is represented as “membership degree / element”.

If an element belongs to two (or more) sets then :

* + For union, we take max of membership degree
  + For intersection, we take min of membership degree
  + For complement, we take (1 – membership degree).

In fuzzy logic, we classify a variable in terms of fuzzy linguistic variables e.g. we classify height as short, medium, tall; age as infant, child, adolescent, adult, young, old, etc.

For large fuzzy sets, it is not possible to manually assign a membership degree, so for that we use membership functions like triangular, trapezoidal, Gaussian, etc.

A fuzzy rule may look like: If temperature is HOT and fan motor speed is LOW then flow rate is HIGH-POSITIVE.

The fuzzy output is defuzzified using methods like centre of gravity, composite maxima, , etc.

The two popular fuzzy reasoning systems are Mamdani and Sugeno. In Mamdani model output is a fuzzy set wheras in Sugeno model output is either a constant or linear function of inputs.

**3.2. Learning**

* Act or process of acquiring knowledge or skill.
* A process which leads to improved performance.
* Learning is the lifelong process, both conscious and unconscious, of transforming information and experience into knowledge, skills, behaviors, and attitudes.

**Types of Learning**

**Rote Learning**: remembering without understanding.

**Learning by taking advice**: ask someone else how to do a task.

**Learning by analogy**: relating a hard problem with a easier problem solved earlier.

**Learning by examples/induction**: learn by taking multiple examples of same concept.

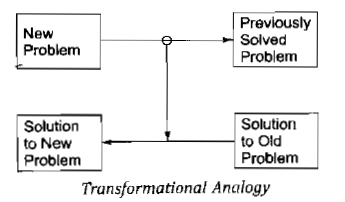
**Learning by problem-solving**: solve multiple problems to learn a concept.

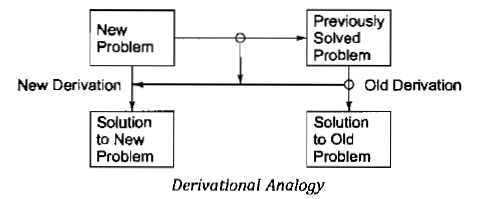
**Learning by Analogy**

**Transformational Analogy**: transform a solution to a previous problem into a solution for the current problem.

It does not look at how the old problem was solved, it just substitutes the values.

**Derivational Analogy**: use the solution of a previously solved problem to derive the solution for the current problem e.g. re-write a Java code in Python.





**Learning from Examples / Learning by Induction**

1. Begin with a structural definition of one known instance of the concept. Call that description the concept definition.
2. Examine descriptions of other known instances of the concept. Generalize the definition to include them.
3. Examine descriptions of near misses of the concept. Restrict the definition to exclude these.

**Learning by Problem-solving**

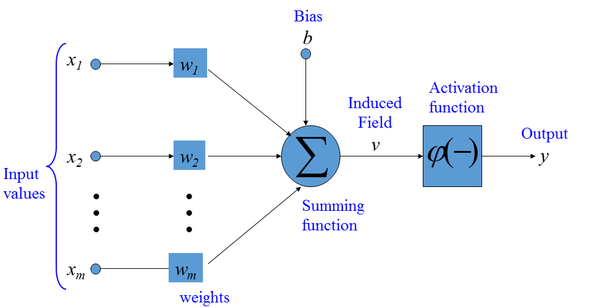
Many programs rely on an evaluation procedure that combines information from several sources into a single summary statistic.

In designing such programs, it is often difficult to know apriori how much weight should be attached to each feature being used.

One way of finding the correct weights is to begin with some estimate of the correct settings and then to let the program modify the settings on the basis of its experience.

Features that appear to be good predictors of overall success will have their weights increased, while those that do not will have their weights decreased, perhaps even to the point of being dropped entirely.

**Artificial Neural Networks**



* 1. **Planning**

The use of methods which focus on decomposing the original problem into appropriate sub-parts and on ways of recording and handling interactions among the subparts as they are detected during the problem-solving process, is often called as planning.

**Components of a planning system**

* Choosing the best rule to apply
* Applying the rules
* Detecting a solution
* Detecting dead ends
* Repairing an almost correct solution

**STRIPS solver**

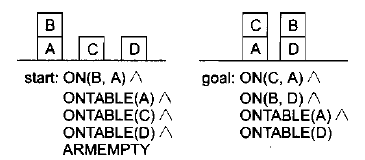
Stanford Research Institute Problem Solver (**STRIPS**) is a popular planning technique.

In this method, the problem solver makes use of a single stack that contains both goals and operators that have been proposed to satisfy these goals.

The problem solver also relies on a database that describes the current situation and a set of operators described as PRECONDITION, ADD and DELETE lists.

PRECONDITION list contains those predicates that must be true for the operator to be applied. ADD contains predicates which become true after the operation, and DELETE contains predicates which become false after the given operation.

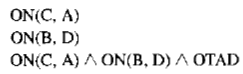
**STRIPS for blocks-world problem**



We separate the problem into 4 sub-problems, one for each component of the goal.

Two of the subproblems, ONTABLE(A) and ONTABLE(D) are already true in initial state.

So, our goal stack is



# OTAD = ONTABLE(A) and ONTABLE(B)

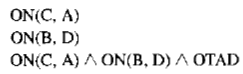
Let the operators for solving the problem are

* STACK(x,y): place block x on block y
* UNSTACK(x,y): remove block x from its current position on block y
* PICKUP(x): pickup block x from table
* PUTDOWN(x): put block x down on table

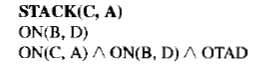
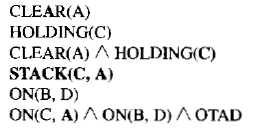
To specify the pre-conditions and post-conditions for operators, let predicates be:

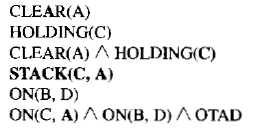
* ON(x,y): block x is on block y
* ONTABLE(x): block x is on table
* CLEAR(x): block x has no block on top of it
* HOLDING(x): robotic arm is holding block x
* ARMEMPTY: robotic arm is empty

As our goal stack is

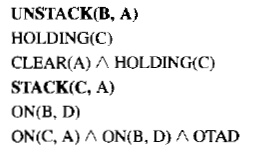
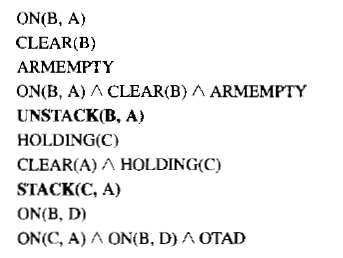


We first check to see whether ON(C,A) is true in the current state. Since it is not, we check for operators that could cause it to be true i.e. STACK(x,y). So, we place STACK(C,A) in place of ON(C,A). But to apply STACK(C,A) its pre-conditions CLEAR(A) and HOLDING (C) must be true.

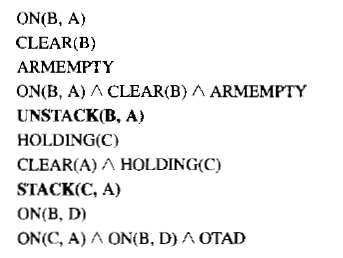
 

So, now our new goal stack is  


Next, we check to see if CLEAR(A) is true. It is not. The only operator that could make it true is UNSTACK(B,A). So, we replace CLEAR(A) with UNSTACK(B,A).

For the goal stack



The conditions ON(B,A), CLEAR(B), ARMEMPTY are satisfied in start state. So, we can pop these predicates from the goal state as well as the combined goal i.e. preconditions of UNSTACK(B,A). We can then also pop UNSTACK(B,A) as its pre-conditions are met.

We keep recording the operators which are being popped off the stack and remember their sequence.

The sequence for the given problem is:

* + UNSTACK(B,A)
  + STACK(B,D)
  + PICKUP(C)
  + STACK(C,A)