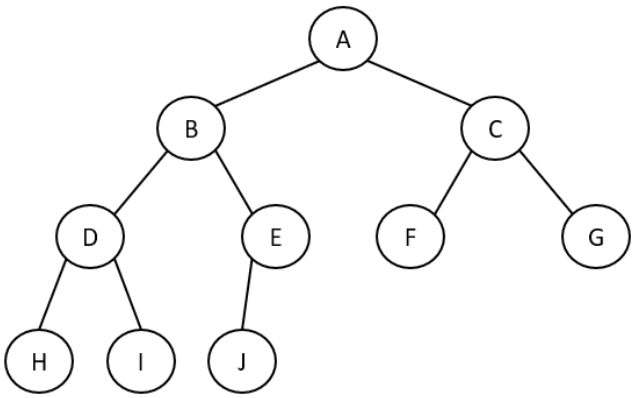
1)

a) False: the Turing test doesn't appeal to the concept of rationality

b)

i) A state space
ii) A start state
iii) A goal state or goal test
iv) A set of actions
v) An action function (mapping from a state and an action to a new state)

c)



Let's take this binary search tree above for example to answer this question. We assume for both algorithms, they traverse from left to right. Hence for the best case, the goal node must be the 1st node on the left. With that, let's take D as the goal node.

Depth First Search

To reach D, the algorithm will traverse from A to B then D, with a total of 3 nodes explored and at depth level 3. We can see that the number of nodes explored is equivalent to the depth level (k). It also works the same if H is the goal node. The algorithm will traverse from A to B to D then H, with a total of 4 nodes explored and at depth level 4.

Therefore, in the best-case scenario, **k** nodes will be traversed.

Breadth First Search

To reach D, the algorithm will traverse from A to B to C then D, with a total of 4 nodes explored and at depth level 3.

If the goal node is H, the algorithm will traverse from A to B to C to D to E to F to G then H, with a total of 8 nodes explored and at depth level 4.

We can see that the number of nodes explored is in the power of 2(2^2 for D and 2^3 for H). We can also see that the power is 1 less than the depth level (k).

Therefore, in the best-case scenario, $\mathbf{2^{k-1}}$ nodes will be traversed.

d) A heuristic h is admissible if, for every node n, it underestimates the cost of getting from n to the closest goal node: there is no path from n to a goal that has path cost less than h(n).

e) A* Search Algorithm

For A* Search Algorithm, it selects the path that minimizes f(n) = h(n) + g(n), where g(n) is the cost of the path from the start node to n and h(n) is a heuristic function that estimates the cost

of the cheapest path from n to the goal.

Node to be expanded	**Frontier [h(n) + g(n) = f(n)]**
	A (16 + 0 = 16)
A	C (11 + 6 = 17), B (16 + 3 = 19)
C	M (2 + 8 = 10), N (0 + 18 = 18), B (19), K (0 + 26 = 26)
M	N (18), B (19), K (26) [Since A is expanded, it is not added to the frontier for A* Search]
N	

From the table we can see that the paths explored for A* Search Algorithm will be:
a → c → m → n
From node a, it can go to node b or c. Node C is chosen as it has the lower f(n). After node c, it can go to node b, m, n, or k. Node m is chosen as it has the lowest f(n). After node m is explored, it should go to node a. But since node a is already visited, it won't traverse to that node again. Then it will go to the node with the lowest f(n) which is node n. Node n is one of the goal nodes. Hence the algorithm will stop here. Since the final path is a → c → n, the total cost will be 6 + 12 = **18**.

Branch and Bound Algorithm

If this algorithm does not have a pruning (will visit the same node all over again), it will not reach the goal node but instead will traverse from a → b → d → e → b → d → e and keep on repeating the same cycle. However, if it has a loop detection, the nodes explored for this algorithm will be a → c → n which is 6 + 12 = **18**.

[You can refer to AISpace on the given problem to have a better understanding.]

2)

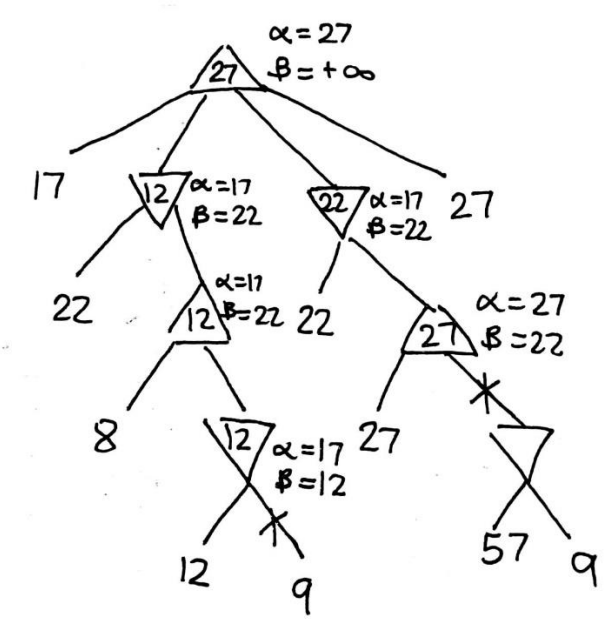
a)

i) Arc (C, E) removes 4 from C.
Arc (C, E) removes 1 from E.
Arc (D, E) removes 4 from D.
Arc (B, D) removes 3, 4 from B.
Arc (C, E) removes 4 from C.
Arc (B, D) removes 1 from D.
Arc (D, E) removes 2 from E.
Arc (A, B) removes 3, 4 from A.
Arc (B, C) removes 1 from C.

ii) This CSP do not have a solution. The constraint C ≠ D make it impossible to have a solution.

b)

i) Ans: 27

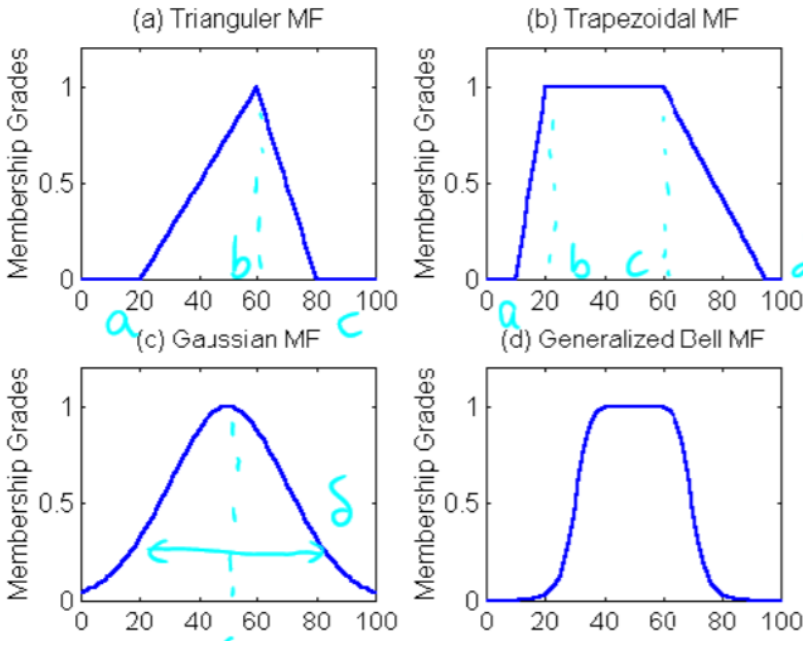


ii) Yes. 27, 17. (If max is the root then arrange it in descending order. If min is the root, then arrange it in increasing order.)

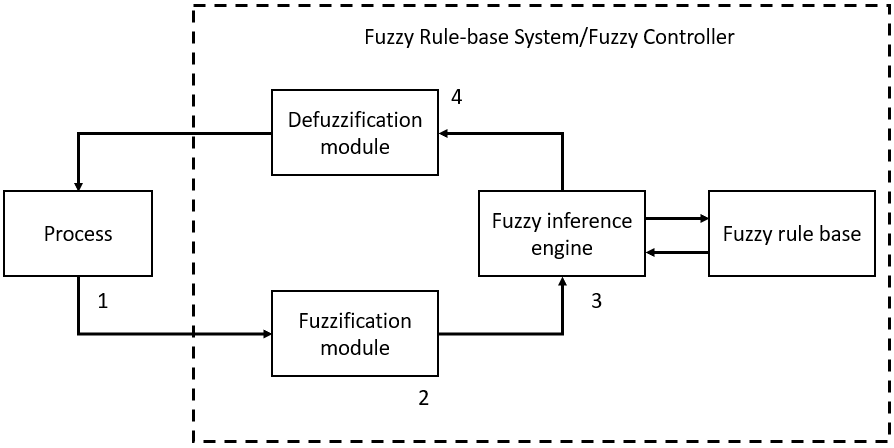
3) For Question 3 and 4, most of them is theory.

a) Triangular MF: $trimf(x; a, b, c) = \max\left(\min\left(\frac{x-a}{b-a}, \frac{c-x}{c-b}\right), 0\right)$

Trapezoidal MF: $trapmf(x; a, b, c, d) = \max\left(min\left(\frac{x-a}{b-a}, 1, \frac{d-x}{d-c}\right), 0\right)$

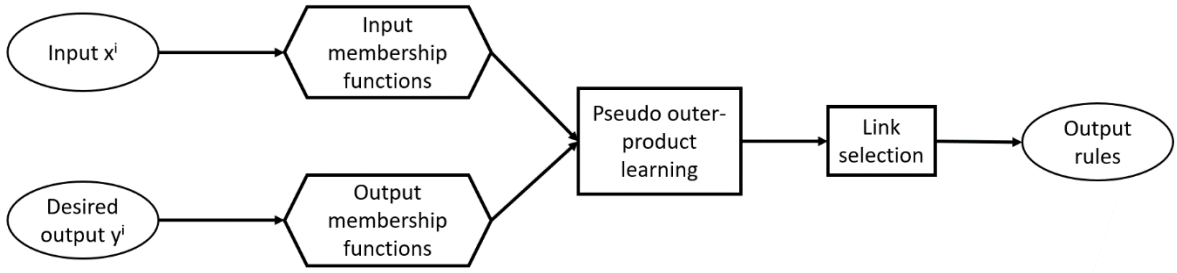


b) The fuzzy inference process overcomes the rule conflict resolution by using the AND operation to make sure all the multiple fuzzy rules are matched.



c) <Refer to diagram in lecture> The five layers are input linguistic layer, condition layer, rule layer, consequent layer and output linguistic layer.
The fuzzy operations performed by each layer of the POPFNN-have very close correspondence with the inference steps in the Compositional Rule of Inference (CRI) method.
Layer one: operation performs a fuzzy singleton fuzzification.
Condition Layer: Each individual antecedent, represented by the trapezoidal membership of an input-label node in the POP-CRI(S), has an associated fuzzy set as the semantic of a linguistic label for a linguistic node.
- actually measures the compatibility of each input fuzzy set with each fuzzy subset Xi,j that semantically represents a linguistic label used as an antecedent in the fuzzy rules in the rule-base layer.
Rule nodes: the rule-based layer computes the firing strength of the kth fuzzy rule using the standard t-norm operator, which is also the total degree of compatibility between the input fuzzy sets and the antecedents of the kth fuzzy rules.
Consequent nodes: the output-nodes perform the disjunction operation and computes the maximum firing strength rm,I of the consequence Ym,I from the degree of compatibility between the input fuzzy sets and the antecedents of all the rules
Output Linguistic Layer: the final output set Ym is composed of several consequences that semantically represent linguistic labels for ym. It implements the centre of maxima method.
- the output linguistic layer integrates the consequences derived from the previous layers and performs defuzzification of these consequences

Learning process:



4)

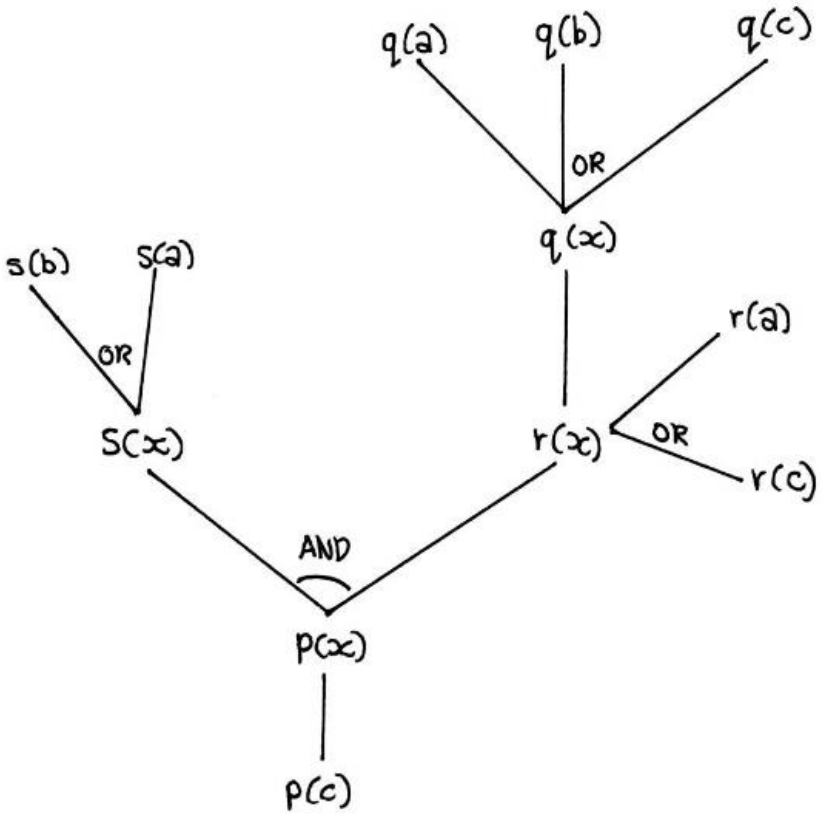
a) $(A \Rightarrow B) \vee \neg(A \Rightarrow B)$
$Let\ (A \Rightarrow B)\ be\ a$
$a \vee \neg a$ [tautology]
Hence, it is true.

b) Truth table. It will be costly if there are more variables.

A	**B**	**A⇒B**	**¬ (A⇒B)**	$(\boldsymbol{A \Rightarrow B}) \vee \neg(\boldsymbol{A \Rightarrow B})$
F	F	T	F	T
F	T	T	F	T
T	F	F	T	T
T	T	T	F	T

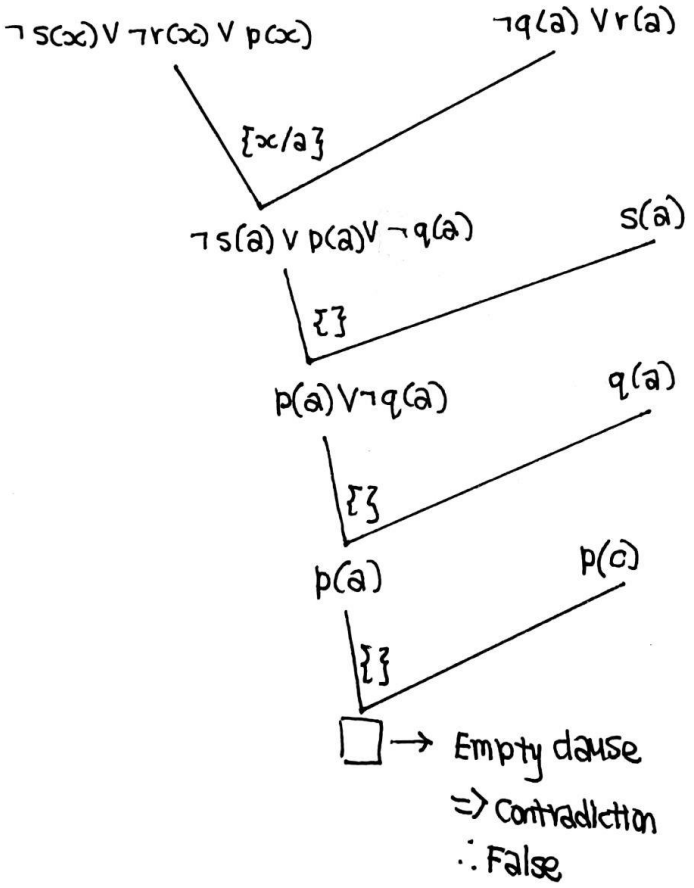
c) None: execute systematically all rules.
No duplication: do not execute the same rules on the same arguments twice.
Recency: favour rules that refer to elements recently created in WM
Specificity: favour rules that are more specific (have more constraints).
Operation priority: favour rules that yield high-priority actions.
The strategy adopted by Prolog is to execute its rules sequentially. It also adopts refutation proof by contradiction. This is done with the "?" symbol in Prolog. When we type in a query, Prolog will negate it. For example, we type in A. Prolog will negate it making it not A. Since A and not A contradicts each other, it means that it is available in KB.

d) A goal is satisfiable if the premises of a rule that match the query/goals are all satisfiable. The binding of the first goal are propagated to the rest of the premises and the bindings of the 1st and 2nd to the rest. Failure of one of the premises will move the search back to the previous attempted premise to look for alternative binding to continue to satisfy the reminding premises. This process repeats and is call goal directly dependency back tracking.



$s(x) \wedge r(x) \Rightarrow p(x)$
$\neg\big(s(x) \wedge r(x)\big) \vee p(x)$
$\neg s(x) \vee \neg r(x) \vee p(x)$

$q(x) \Rightarrow r(x)$
$\neg q(x) \vee r(x)$



--End of Answers--