IT5005 Artificial Intelligence

Sirigina Rajendra Prasad
AY2025/2026: Semester 1

Consultation Session

4 October 2025

When writing proofs using forward chaining, do we need to present all the detailed intermediate steps, or is it acceptable to write them in a more concise way (e.g., directly stating the conclusion followed by the applied rule in brackets such as "… (transitivity from R1)")?

- No need, you won’t be asked to write the trace of the algorithm. The detailed trace in tutorial solutions is only to enable in-depth understanding of algorithm.

  Usually, you would be asked to answer specific questions related to algorithm.

If we need to prove the consistency or admissibility of multiple heuristics, when showing that a heuristic does not satisfy the condition, is providing a single counterexample sufficient, or do we need to list all of them? For example, a heuristic violates admissibility at several nodes.

- Single counter example is sufficient

Both BFS and DFS can be implemented as either graph/tree search, and using either delayed/eager checking. The slides present one method for each (e.g. BFS is graph with eager checks). Should we expect to use the exact same method as in the slides?

- If the question requires development of variants of these algorithms, you would be provided with base variants. Pseudocodes for the algorithms and logical equivalence tables would be provided in the question paper.

In GBFS trace (slide 73), "frontier" is subscripted by h, while "s" and "reached" is subscripted by path cost. We end up losing information, as frontier does not track path cost. i.e. when we pop from frontier, we may not know what the path cost to that node was. How to resolve this? Can we track it

- By definition, each node has path_cost as the attribute. GBFS is ignoring this readily available information. At the end, we can use path_cost information from the goal node.
- Tables are provided to help trace the algorithms. Notations and subscripts are not important. Decisions, identification of nodes from frontier is important

please explain a bit on the Q.5 in mock paper. 5. Given the following statements, pick the odd one.

A. $P \Longrightarrow (P \vee Q)$

B. $(P \wedge Q) \Longrightarrow P$

C. $(P \Longrightarrow P) \wedge Q$

D. $Q \vee (Q \Longrightarrow P)$

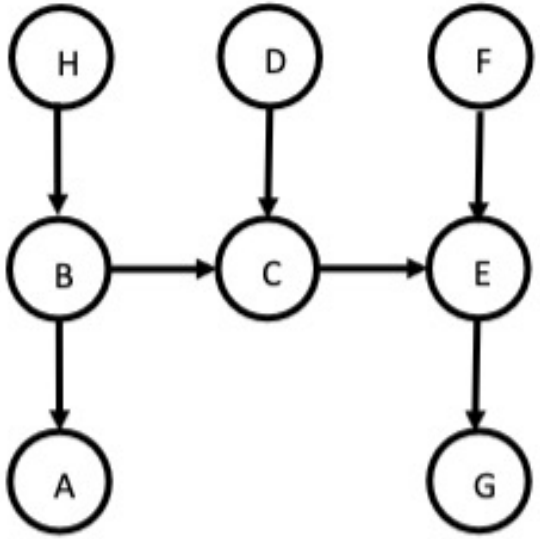
E. $False \Rightarrow True$

- C is not tautology. Rest of the options are tautologies

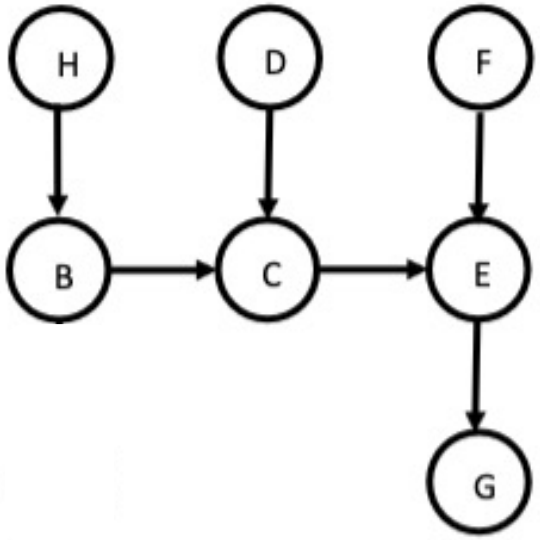
How to draw the moralized diagram?

- Check for a node with two parent nodes. If the parent nodes are not connected, join them.

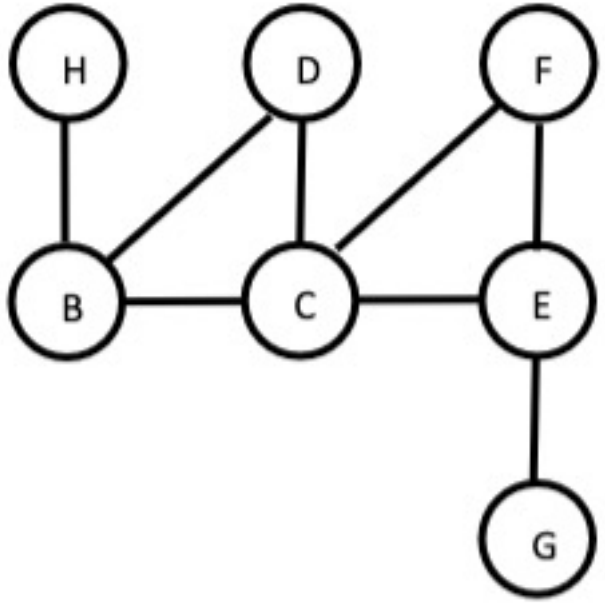
b(i). Prove/Disprove $H \perp F|G$



Ancestral Subgraph



Moralized Ancestral Subgraph



Do we need to provide detailed calculation to prove heuristics are admissible and consistent? e.g. Q7 in past paper. Or just provide the rationale for those which are not? Because it's quite time consuming during exam

- Yes, one counter example is enough

Hi Prof, could you help to explain why do we use bi-implication instead of implication to represent R1: A says B is lying. in Q8 of past paper

Three persons A, B, and C make the following statements:
R1: A says B is lying.
R2: B says C is lying.
R3: C says both A and B are lying.
We also know the following fact.
R4: Only one of the A, B, C is telling the truth.

A says B is lying.

If A is telling the truth, then B is not telling the truth

$$a \Rightarrow \neg b$$

If A is not telling the truth, then B is telling the truth

$$\neg a \Rightarrow b$$

Which is equivalent to $\neg b \Rightarrow a$

Symbol	Definition
a	A is telling the truth
b	B is telling the truth
c	C is telling the truth

- R1: $a \Leftrightarrow \neg b$
  R2: $b \Leftrightarrow \neg c$
  R3: $c \Leftrightarrow (\neg a \wedge \neg b)$
  R4: $(a \vee b \vee c) \wedge \neg(a \wedge b) \wedge \neg(a \wedge c) \wedge \neg(b \wedge c)$

How do we differentiate and tell if the statement is an "if and only if" statement or not, if those words are not explicitly stated? Considering the JAN MYE, Q8

Hi Prof, could you help to explain why do we use bi-implication instead of implication to represent R1: A says B is lying. in Q8 of past paper

Three persons A, B, and C make the following statements:
R1: A says B is lying.
R2: B says C is lying.
R3: C says both A and B are lying.
We also know the following fact.
R4: Only one of the A, B, C is telling the truth.

Symbol	Definition
a	A is telling the truth
b	B is telling the truth
c	C is telling the truth

- R1: $a \Leftrightarrow \neg b$
  R2: $b \Leftrightarrow \neg c$
  R3: $c \Leftrightarrow (\neg a \wedge \neg b)$
  R4: $(a \vee b \vee c) \wedge \neg(a \wedge b) \wedge \neg(a \wedge c) \wedge \neg(b \wedge c)$

C says both A and B are lying.

If C is telling the truth, then A is not telling the truth and B is not telling the truth

$$c \Rightarrow \neg a \wedge \neg b$$

If C is not telling the truth, then either A is telling the truth or B is telling the truth

$$\neg c \Rightarrow a \vee b$$

Dear professor, could you please go over the Q6 in last semester's midterm? Since there are many concepts involved in this question and may need thorough understanding of each concept. And also wondering what determines "efficiency"? Thank you very much!

You are advised to use best-first search algorithm for a state space with uniform costs, i.e., the costs of all actions at all states are the same. Moreover, you are told that the state space is a tree. Briefly describe what changes would you make such that the best-first search algorithm is both optimal and efficient. Assume that you have no heuristics for this problem. The Best-First search algorithm is provided in the appendix.

For uniform costs, breadth-first search is optimal.

The state space is a tree, i.e., no redundant paths and cycles in the state space. Therefore, we dor need *reached* data structure.

The delayed goal search is not efficient. We can do goal search as soon as we generate the node.

We can select $g(n)$ as the depth of the node, making uniform cost search a breadth-first search algorithm.

```
function BEST-FIRST-SEARCH(problem, f) returns a solution node or failure
  node ← NODE(STATE=problem.INITIAL)
  frontier ← a priority queue ordered by f, with node as an element
  reached ← a lookup table, with one entry with key problem.INITIAL and value node
  while not IS-EMPTY(frontier) do
    node ← POP(frontier)
    if problem.IS-GOAL(node.STATE) then return node
    for each child in EXPAND(problem, node) do
      s ← child.STATE
      if s is not in reached or child.PATH-COST < reached[s].PATH-COST then
        reached[s] ← child
        add child to frontier
  return failure
```

Hi Prof, can I check my understanding of the d-seperation principle? A ->{some node}-> B <- {some node}<- C. assuming, I know one know each in {some nodes}, as long as the path is blocked, A is conditional independent of C | {some nodes / B}? What happens if it is A -{nodes1}-> B -> {node2}<-C?

Hi Prof, can I check my understanding of the d-seperation principle? A ->{some node}-> B <- {some node}<- C. assuming, I know one know each in {some nodes}, as long as the path is blocked, A is conditional independent of C | {some nodes / B}? What happens if it is A -{nodes1}-> B -> {node2}<-C?