COSC1125/1127 Artificial Intelligence

School of Science (Computer Science)
RMIT University
Semester 1, 2018
A/Prof. Sebastian Sardina

Tutorial Sheet 1
Search

- Open Discussion: *What is Artificial Intelligence?*
- Watch the videos:
  - Holy Grail of AI: https://www.youtube.com/watch?v=tlS5Y2vm02c
  - Humans Need Not Apply: https://www.youtube.com/watch?v=7Pq-S557XQU
  - Artificial Intelligence: https://www.youtube.com/watch?v=oYqXQw2CryI
  - The long-term future of AI: https://www.youtube.com/watch?v=CK5w3wh4G-M

Exercises

1. (RN) Define in your own words the following terms: state, state space, search tree, search node, goal, action, successor function, and branching factor.

Answer

A **state** is a situation that an agent can find itself in. We distinguish two types of states: world states (the actual concrete situations in the real world) and representational states (the abstract descriptions of the real world that are used by the agent in deliberating about what to do).
A **state space** is a graph whose nodes are the set of all states, and whose links are actions that transform one state into another.
A **search tree** is a tree (a graph with no undirected loops) in which the root node is the start state and the set of children for each node consists of the states reachable by taking any action.
A **search node** is a node in the search tree.
A **goal** is a state that the agent is trying to reach.
An **action** is something that the agent can choose to do.
A **successor function** described the agents options: given a state, it returns a set of (action, state) pairs, where each state is the state reachable by taking the action.
The **branching factor** in a search tree is the number of actions available to the agent.

2. (RN) Whats the difference between a world state, a state description, and a search node? Why is this distinction useful?

Answer

A world state is how reality is or could be. In one world state were in Arad, in another were in Bucharest. The world state also includes which street were on, whats currently on the radio, and the price of tea in China. A state description is an agents internal description of a world state. Examples are In(Arad) and In(Bucharest). These descriptions are necessarily approximate, recording only some aspect of the state.
We need to distinguish between world states and state descriptions because state description are lossy abstractions of the world state, because the agent could be mistaken about21 how the world is, because the agent might want to imagine things that arent true but it could make true, and because the agent cares about the world not its internal representation of it.
Search nodes are generated during search, representing a state the search process knows how to reach. They contain additional information aside from the state description, such as the sequence of actions used to reach this state. This distinction is useful because we may generate different search nodes which have the same state, and because search nodes contain more information than a state representation.

3. Consider this problem: We have one 3 litre jug, one 5 litre jug and an unlimited supply of water. The goal is to be able to drink exactly one litre. Either jug can be emptied or filled, or poured into the other.

   For this problem give:

   (a) An appropriate data structure for representing a state.

   Answer

   The representation of the state can be simply: (nL, nR), where nL is the number of litres in the 3 litre jug, and nR is number of litres in the 5 litre jug.

   (b) The initial state.

   Answer

   The initial state is $(0, 0)$.

   (c) All the final goal state(s).

   Answer

   The goal states are $(1, n)$ or $(n, 1)$, for any number n, as we just need to get some jug with exactly one liter.

   (d) A specification of the operators (or actions) which includes the preconditions that must be satisfied before the operator can be used and the new state generated.

Answer

Operators could be (using FOL sentences):

$\underline{Fill(X)}$
Action: fill jug X
Precondition: jug X is not full
State generated: jug X is full - $(3, nR)$ or $(nL, 5)$

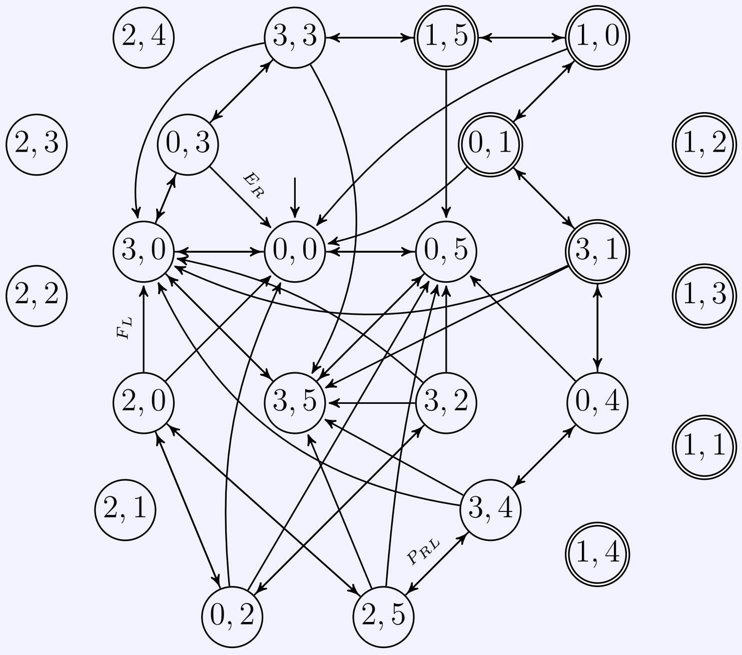
$\underline{Pour(X, Y)}$
Action: pour jug X into jug Y until either X is empty or Y is full
Precondtion: jug X is not empty and jug Y is not full
State Generated:

- jug X is empty and jug Y is partially full - $(0, nR)$ or $(nL, 0)$, or
- jug X is partially full and jug Y is full - $(3, nR)$ or $(nL, 5)$

$\underline{Empty(X)}$
Action: empty jug X
Precondition: jug X is not empty
State generated: jug X is empty - $(0, nR)$ or $(nL, 0)$

(e) Draw the full state space.

Answer



State transition labels:
F_X fill jug X
E_X empty jug Y
P_{XY} pour jug X into jug Y

Note: For clarity, not all state transition labels are not shown on this diagram, usually you should show all state transition labels above all arcs. Observe also that states that cannot be reached from the initial state, such as (1,4), are still part of the full state space. If you are not going to show inaccessible states, you should explicitly state so.

(f) What is the solution to the problem.

Answer

$(0, 0) \rightarrow (3, 0) \rightarrow (0, 3) \rightarrow (3, 3) \rightarrow (1, 5) \rightarrow (1, 0)$

(g) How did you find that solution? Would a computer be able to find it the same way? Explain either way. If not, how would you make an algorithm to find a solution given the search space?

Answer

If you just found it by "matching" a path connecting $(0,0)$ to any state $(1,n)$ or $(n,1)$, then it will probably be difficult to do with an algorithm, as it is not systematic approach. An algorithm would have to "visit" nodes in a systematic, regular, complete, and hopefully non-redundant manner.

Check Jonathon's nice Python solution `jug_problem.py` via a simple random search. What would you change there to make it non-random and systematic? How would you change it so as not to need a depth bound?

4. Does a finite state space always lead to a finite search tree? How about a finite state space that is a tree?

Answer

Finite state can yield an infinite state if there are loops in the state space; if the state space is a tree then there are no loops and the search tree will be finite

5. Consider the problem of getting from Arad to Bucharest in Romania. For this problem give:

- Search state descriptions.

Answer

A possible search state description could be $(\langle city \rangle, g(n))$, where $g(n)$ is the total path cost to that node. (Note that the real search state in a search algorithm will also contain the link to the parent of the state node, but we omit it here.)

- Initial State.

Answer

$(Arad, 0)$

- Final goal search states.

Answer

$(Bucharest, g(n))$, where $g(n)$ is total path cost from Arad to Bucharest.

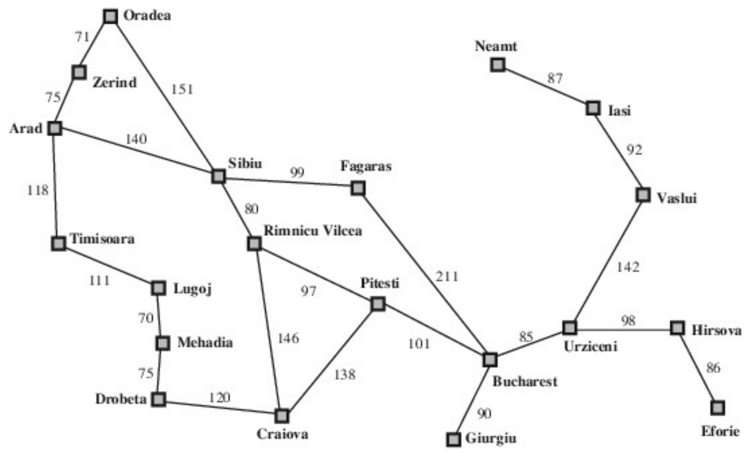
- Operators (or actions).

Answer

Only one operator exists (using FOL sentences):

$\underline{Go(X,Y)}$
Action: travel from city X to city Y
Precondition: currently in city X and city X has an edge connecting it to city Y
State generated: in city Y - $(Y, g(n))$, where $g(n))$ is total path cost to city Y.

The reason $Go(X)$ is not used, is that it does not take into account the fact that we must travel between connected cities.



Using your representation, how would you build a “search tree” starting from state “Arad”?

6. Why is search an important technique in AI and CS? When would you use search and when you wouldn’t? Give concrete examples and reasons.

Answer

It is important because it is an extremely general technique that can solve any (doable) problem, provided it is modelled as a search task.
One would use it when there is no direct technique to solve a problem (e.g., finding the route in a map or playing chess), but one would not use it if an effective algorithm exist for the problme (e.g., sorting a list or computing the factorial of a number), because, while general, is computationally demanding.

7. *You say more?* Lots of cool exercise in RN book, chapter 3....

COSC1125/1127 Artificial Intelligence

School of Science (Computer Science)
RMIT University
Semester 1, 2018
A/Prof. Sebastian Sardina

Tutorial Sheet 2
Search II

Exercises

1. **From Tutorial 1.** Consider this problem: We have one 3 litre jug, one 5 litre jug and an unlimited supply of water. The goal is to get exactly one litre of water into either jug. Either jug can be emptied or filled, or poured into the other.

   For this problem give:

   (a) An appropriate data structure for representing a state.

   (b) The initial state.

   (c) The final states (there are 2).

   (d) A specification of the operators (or actions) which includes the preconditions that must be satisfied before the operator can be used and the new state generated.

   (e) What is the solution to the problem.

   Answer

   See solution to tutorial 1.

2. In the previous exercise, a representation for states and the full state space were developed. For the same problem, apply search strategies and note:

   - *The order in which nodes are created in memory.*
   - *The nodes that are not created in memory at all.*

   for the following search strategies:

   (a) Breadth first search with no checking for duplicate states.

Answer

$$\begin{tikzcd}
 & & & & & (0,0) \arrow[dll] \arrow[drr] & & & \\
 & & & (3,0) \arrow[dll] \arrow[d] \arrow[drr] & & & & (0,5) \arrow[dl] \arrow[d] \arrow[dr] & \\
 & (0,0) \arrow[dl] \arrow[d] & & (0,3) \arrow[dl] \arrow[d] \arrow[dr] & & (3,5) \arrow[d, dotted, no head] & (0,0) & (3,2) \arrow[d, dotted, no head] & (3,5) \\
 (3,0) \arrow[d, dotted, no head] & (0,5) \arrow[d, dotted, no head] & (0,0) & (3,0) & (3,3) & \cdots & & \cdots & \\
 \cdots & \cdots & & & & & & &
\end{tikzcd}$$

Basically you need you generate every possible state from a parent state, and you do this state by state (from left to right), and by level (from top down). Note that operators applied are not shown over each arc, but you should do so normally.

(b) Breadth first search with checking for duplicate states.

Answer

$$\begin{tikzcd}
 & & & & (0,0) \arrow[dllll] \arrow[drr] & & & & & \\
(3,0) \arrow[d] \arrow[drrr] & & & & & & (0,5) \arrow[dlll] \arrow[dr] & & & \\
(0,3) \arrow[dr] & & & (3,5) & & & & (3,2) \arrow[dr] & & \\
 & (3,3) \arrow[dr] & & & & & & & (0,2) \arrow[dr] & \\
 & & (1,5) \arrow[drr] & & & & & & & (2,0) \\
 & & & & (1,0) \arrow[dr] & & & & & \\
 & & & & & (0,1) & & & &
\end{tikzcd}$$

Note that you still expand nodes level by level, but just not generating nodes that are already there.

(c) Depth first search with no checking for duplicate states.

Answer

$$\begin{tikzcd}
 & & & (0,0) \arrow[dll] \arrow[dr] & \\
 & (3,0) \arrow[dl] \arrow[d] \arrow[dr] & & & (0,5) \\
(0,0) \arrow[d] \arrow[dr] & (0,3) & (3,5) & & \\
(3,0) \arrow[d, dotted, no head] & (0,5) \arrow[d, dotted, no head] & & & \\
\ldots & \ldots & & &
\end{tikzcd}$$

Expand nodes in the order as displayed in (a), ie. DFS here keeps expanding the left-most branch branch of the search tree. Since it is not checking for duplicate states, it can go infinitely deep. If a stack data-structure is used, then it will expand the right-most branches.

(d) Depth first search with checking for duplicate states.

Answer

$$\begin{tikzcd}
 & & & (0,0) \arrow[dll] \arrow[drr] & & \\
 & (3,0) \arrow[dl] \arrow[dr] & & & & (0,5) \\
(0,3) \arrow[dr] & & (3,5) & & & \\
 & (3,3) \arrow[dr] & & & & \\
 & & (1,5) \arrow[dr] & & & \\
 & & & (1,0) \arrow[dr] & & \\
 & & & & (0,1) &
\end{tikzcd}$$

(e) Iterative deepening with no checking for duplicate states.

Answer

Step 1:

$$(0,0)$$

Step 2:

$$\begin{tikzcd}
(0,0) \arrow[d] \arrow[dr] & \\
(3,0) & (0,5)
\end{tikzcd}$$

Step 3:

$$\begin{tikzcd}
 & & (0,0) \arrow[dl] \arrow[drr] & & & \\
 & (3,0) \arrow[dl] \arrow[d] \arrow[dr] & & & (0,5) \arrow[dl] \arrow[d] \arrow[dr] & \\
(0,0) & (0,3) & (3,5) & (0,0) & (3,2) & (3,5)
\end{tikzcd}$$

This goes for as many more steps as is required to find the goal.

(f) Iterative deepening with checking for duplicate states.

Answer

Similar to (e) without showing the duplicated states.

(g) Is bi-directional search possible for this problem?

Answer

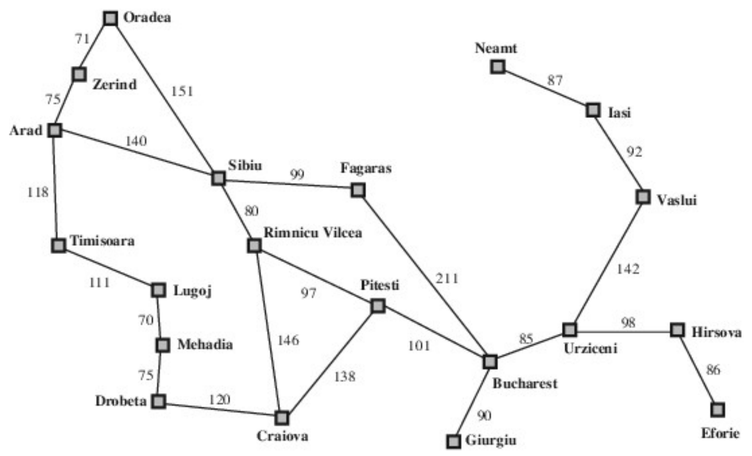
Yes, it is possible to use bi-directional search, since you know both the initial and goal states. Bi-directional search basically means that you expand nodes from the initial and goal nodes simultaneously (with or without checking for duplicates), until the two search trees are connected to form a single path from the initial to the goal state. One caveat to this is that, without being given the full state space, it can be difficult to search backwards as new transitions (eg. 'unfill', 'unempty' and 'unpour') need to be defined which are not all that intuitive.

3. Consider the problem of getting from Arad to Bucharest in Romania from Tutorial 1. Provide the part of the search space that is realized in memory and the order of node expansion if *uniform cost search* is used.

Answer

$$\begin{tikzcd}
 & & & & 1 & & & & & \\
 & & & & (A,0) \arrow[dllll, "3"'] \arrow[d, "4"] \arrow[drrrr, "2"] & & & & & \\
(T,118) \arrow[dr, "7"] & & & & (S,140) \arrow[d, "6"] \arrow[dr, "8"] & & & & (Z,75) \arrow[dr, "5"] & \\
 & (L,229) \arrow[dr, "9"] & & & (R,220) \arrow[d, "11"] \arrow[dr, "10"] & (F,239) \arrow[dr] & & & & (O,146) \\
 & & (M,299) \arrow[dr, "12"] & & (C,366) \arrow[d] & (P,317) \arrow[d] \arrow[dr, "13"] & (B,450) & & & \\
 & & & (D,374) & (D,486) & (C,455) & (B,418) & & &
\end{tikzcd}$$

Cities are represented by their first letter and the red numbers indicate the order in which nodes are expanded.



4. What are the dimensions we judge the various search algorithms? Discuss each of them for each algorithm.

Answer

Dimensions: Time required, space required, completeness, and optimality.
All search algorithms are worst-case exponential in time, but they vary across the other properties! Check slides and book.

5. (RN) Which of the following are true and which are false? Explain your answers.

- Depth-first search always expands at least as many nodes as A search with an admissible heuristic.
- $h(n) = 0$ is an admissible heuristic for the 8-puzzle.
- A* is of no use in robotics because percepts, states, and actions are continuous.
- Breadth-first search is complete even if zero step costs are allowed.
- Assume that a rook can move on a chessboard any number of squares in a straight line, vertically or horizontally, but cannot jump over other pieces. Manhattan distance is an admissible heuristic for the problem of moving the rook from square A to square B in the smallest number of moves.

Answer

- *False*: a lucky DFS might expand exactly d nodes to reach the goal. A largely dominates any graph-search algorithm that is guaranteed to find optimal solutions.
- *True*: $h(n) = 0$ is always an admissible heuristic, since costs are nonnegative.
- *False*: A* search is often used in robotics; the space can be discretized or skeletonized.
- *True*: depth of the solution matters for breadth-first search, not cost.
- *False*: a rook can move across the board in move one, although the Manhattan distance from start to finish is 8.

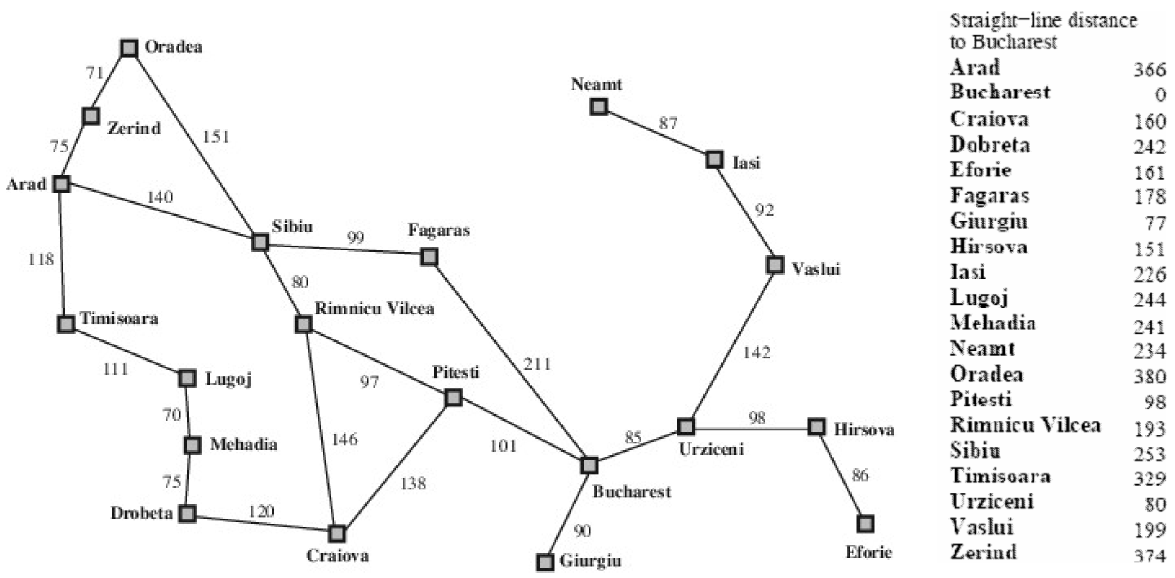
6. If you finish this sheet, you can start with the heuristic search algorithms in Tutorial Sheet 3. :-)
7. Finally, get your hands dirty by doing this fun Lab-Search sheet.
8. *You say more?* Lots of cool exercise in RN book, chapter 3....

COSC1125/1127 Artificial Intelligence

School of Science (Computer Science)
RMIT University
Semester 1, 2018
A/Prof. Sebastian Sardina

Tutorial Sheet 3
Heuristic and Adversarial Search

1. Consider the problem of getting from Arad to Bucharest in Romania and assume the straight line distance (SLD) heuristic will be used.



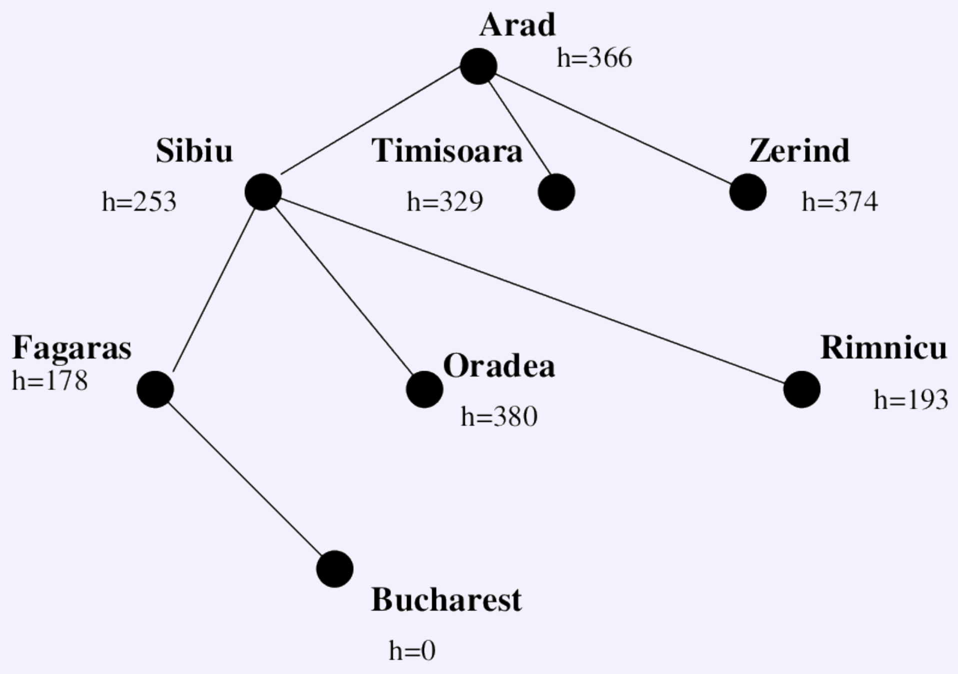
Straight−line distance to Bucharest

Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	178
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	98
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

(a) Give the part of the search space that is realized in memory and the order of node expansion for:

i. Greedy search assuming that a list of states already visited is maintained.

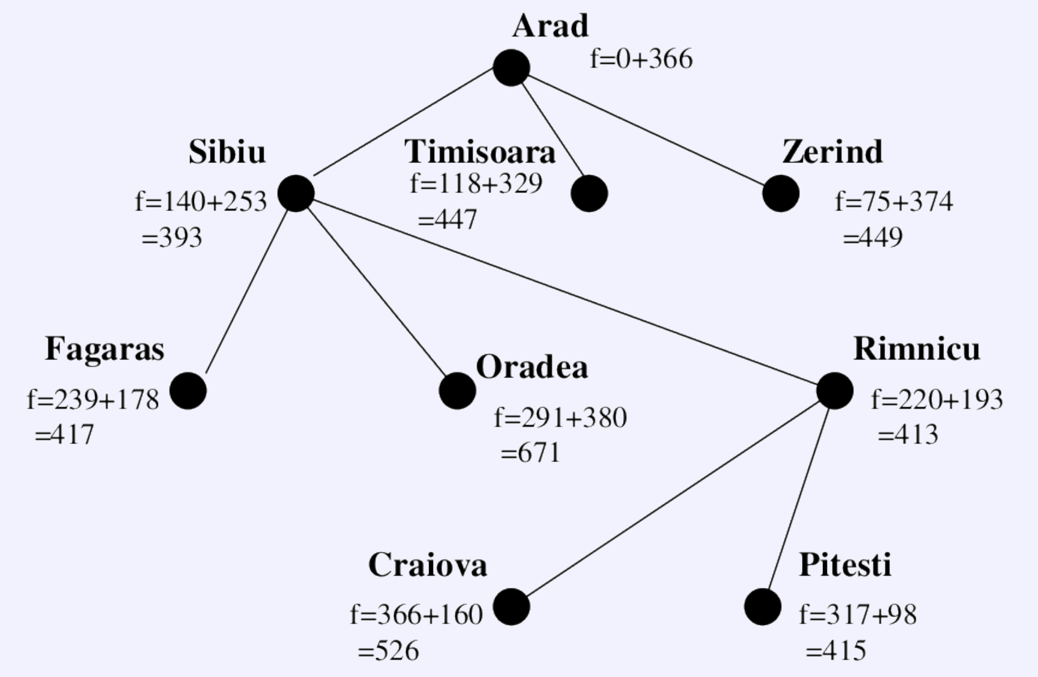
Answer



Greedy search for Bucharest, using the straight-line distance to Bucharest as the heuristic function h_{SLD}. Nodes are labelled with their h-values.
The order of nodes chosen for expansion: Arad, Sibiu, Fagaras, Bucharest.

ii. A* search assuming that a list of states already visited is maintained.

Answer

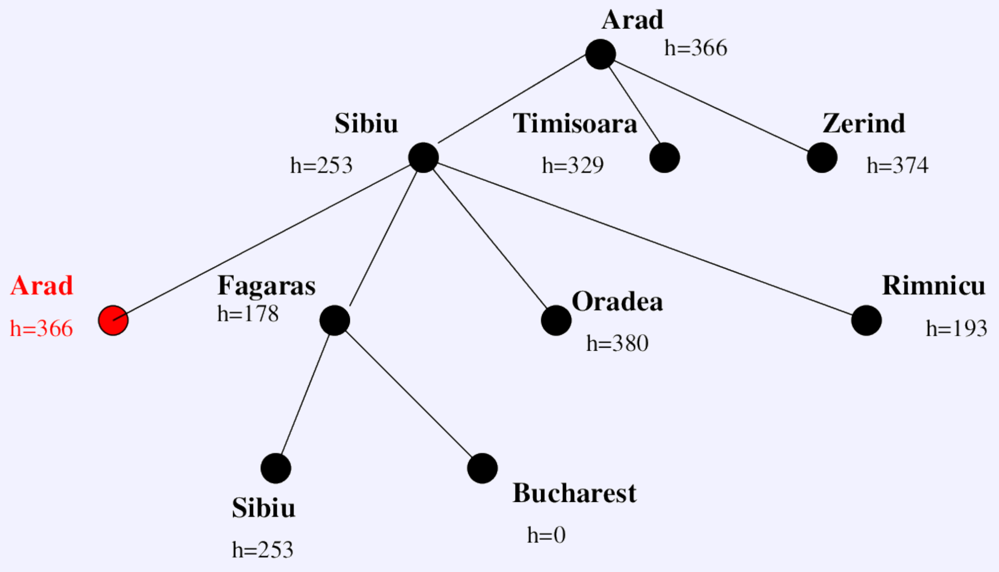


A* search for Bucharest. Nodes are labelled with $f = g + h$. the h values are the straight-line distances to Bucharest taken from the right column of Q1 Figure. The order of nodes chosen for expansion: Arad, Sibiu, Rimnicu, Pitesti.

(b) How would the above searches differ if the list of states already visited is NOT maintained?

Answer

If the list of states already visited is NOT maintained, then there should be an extra node ”Arad” added to both the above figures. For example, for greedy search:



(c) How do the above searches perform for planing a trip from Iasi to Fagaras? You do not need to do the detailed search (you are not given the heuristic function to Fagaras), but use the graphical map to extract the straight distance.

Answer

If the list of states visited is not maintained when appying the greedy search, the search will get stuck at "Neamt", because it gives the lowest h value (straight-line distance). If the visited states are checked and not allowed to be revisited, then both the greedy and A^* searches should be able to find Fagaras eventually, but they might perform differently (similar to the above two figures).

2. Suppose we run a greedy search algorithm with $h(n) = -g(n)$. What sort of search will result?

Answer

This heuristic tries to get as far away from the initial state as possible, but it has no concern with where the goal state is. This of course assumes that the algorithm tries to minimizes $f(n)$.

3. Suppose we run an A^* algorithm with $h(n) = 0$. What sort of search will result?

Answer

A^* with $h(n) = 0$ is the same as uniform cost search, which always expands the node with the smallest cost (or distance in the above example) to the initial state.

4. Explain why the set of states examined by A^* is a subset of those examined by breadth-first search when the cost of every step is always 1.

Answer

Breadth-first search can be seen as an A^* search with $g(n)$ being the depth of the search node n and $h(n) = 0$. So, in BFS, the decision for considering a state is based solely on its distance from the start state, so it will always be more "conservative" than any A*. Using any heuristic that is better than just zero, will cause the search to explore less nodes.

Refer to "heuristic domination" in the book (Section 4 in edition 2). Also, in section 4.2.3 of *Luger, G.F. Artificial Intelligence: Structures and Strategies for Complex Problem Solving, Addison-Wesley* (edition 2002) one can find a *proof* that shows that better heuristics makes A* explore less nodes.

5. Is there a heuristic that would be useful for the missionaries and cannibals problem? The generalized missionaries and cannibals problem (n missionaries and n cannibals)?

Answer

One obvious heuristic measuring the goodness of a state is “number of people on the starting bank” - initially 6, goal 0 (or counting the number of people on the opposite riverbank). In fact, we can simplify the representation used in the M&C example used in the lecture slides:

(# cannibals on the left, # missionaries on the left, (1 if boat is at left, else 0))

Moves are shown as #cannibals + #missionaries in boat - F = forward, B = back:

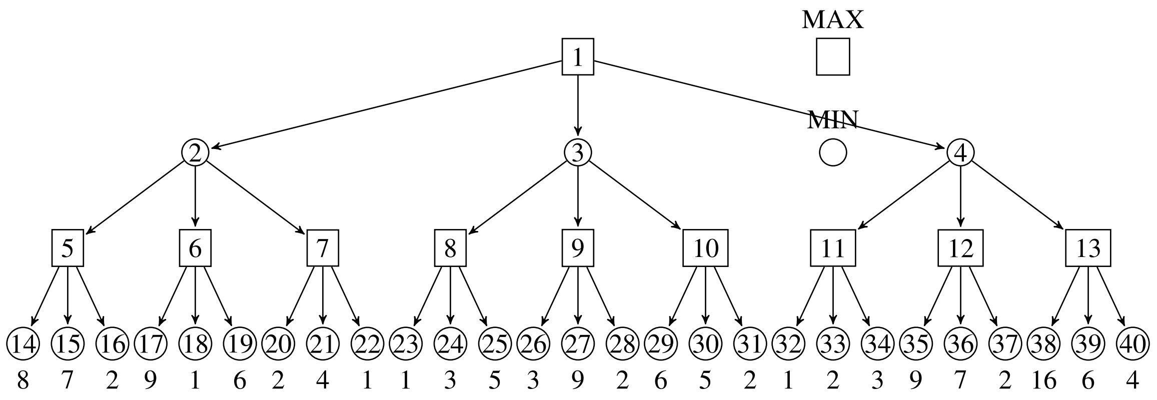
$(3,3,1) \underset{1+1F}{\rightarrow} (2,2,0) \underset{0+1B}{\rightarrow} (2,3,1) \underset{2+0F}{\rightarrow}$

$(0,3,0) \underset{1+0B}{\rightarrow} (1,3,1) \underset{0+2F}{\rightarrow} (1,1,0) \underset{1+1B}{\rightarrow}$

$(2,2,1) \underset{0+2F}{\rightarrow} (2,0,0) \underset{1+0B}{\rightarrow} (3,0,1) \underset{2+0F}{\rightarrow}$

$(1,0,0) \underset{1+0B}{\rightarrow} (2,0,1) \underset{2+0F}{\rightarrow} (0,0,0)$

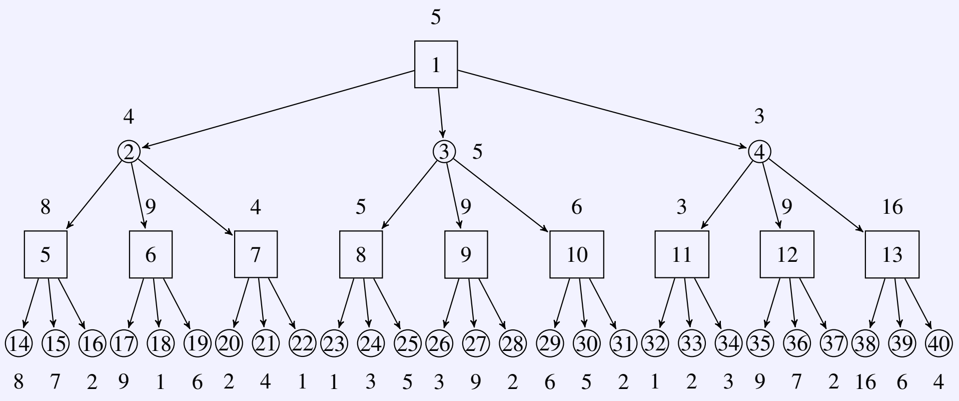
6. Consider the following game tree. Player MAX plays first and is represented with rectangles; MIN player is represented with circles. Numbers in each node are names used for convenience to refer to them (starting node is node 1). Finally, utility of leaf nodes are shown below them (e.g., the utility of node 21 is 4).



(a) Use mini-max to determine the best move for MAX.

Answer

MAX moves to the node 3 with value of 5:

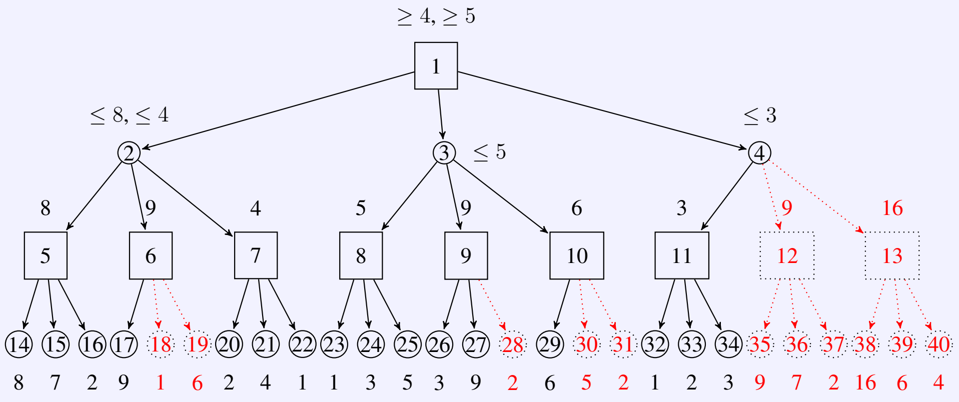


Try it in in http://kra.lc/projects/gamevisual/launch.php

(b) Which nodes will not be examined if the alpha-beta procedure is used?

Answer

The Figure below shows the nodes pruned by the alpha-beta procedure in red and dotted edges:



Try it in in http://kra.lc/projects/gamevisual/launch.php using $b = 3$, $d = 4$, and the following list of values: 8,7,2,9,1,6,2,4,1,1,3,5,3,9,2,6,5,2,1,2,3,9,7,2,16,6,4

(c) In which order will the nodes be examined by the alpha-beta procedure (assuming its depth-first implementation)?

Answer

14, 15, 16, 5, 17, 6, 20, 21, 22, 7, 2
23, 24, 25, 8, 26, 27, 9, 29, 10, 3,
32, 33, 34, 11, 4

(d) Did the alpha-beta procedure give the same best move as mini-max?

Answer

Yes.

7. Does A*'s search time always grow at least *exponentially* with the length of the optimal solution?

Answer

No. It will be exponential in the worst case but there are examples where it will not. For example, if $h(n) = h^*(n)$, that is, the heuristic is perfect, then it will be linear on the depth of the optimal solution as the search will go directly towards it. The better the heuristic, the more efficient the search will be.

8. If $h(\cdot)$ is admissible and s is the start node, how is $h(s)$ related to the cost of the solution found by A* search?

Answer

$h(s)$ will be less or equal than the solution found by A*. Because it is admissible, $h(s) \leq h^*(s)$, where $h^*(s)$ is the (real) cost of the optimal solution from s. Now, because $h(s)$ is admissible, we know that A* will find an optimal solution, that is, will find a solution with cost $h^(s)$.

9. Prove that if $h(n) = h^*(n)$ for all nodes n, then whenever A* expands a node x, x must lie on an optimal path to a goal.

Answer

There are many ways to prove this one. Here is one:

Suppose x is node that has been expanded but it is not in the optimal path to a goal. Suppose that $h(s_0) = h^*(s_0) = c^*$, that is, the cost of the optimal path from the initial state is c^*. First, if x_o is a node on an optimal path to the goal, we know that $f(x_o) = g(x_o)+h(x_o) = g(x_o) + h^*(x_o) = c^*$ So, if x was at any point expanded, it means that $f(x) < f(x_o)$ for some node x_o lying on an optimal path to the goal, that is, at some point of the search x was preferred over nodes lying on the optimal path. This means that $g(x) + h(x) < c^*$ and because $h(x) = h^*(x)$ we get that $g(x) + h^*(x) < c^*$. But this means that going thought node x is actually cheaper than c^* and hence $f(s_0) = h(s_0) = h^*(s_0) = c^* < c^*$, a contradiction. Thus, that node x expanded but not lying on an optimal path to the goal cannot exist.

10. Prove that if a heuristic is consistent, then it is also admissible. What about the converse?

Answer

(This is not the proof but a strong hint how to do it)

By induction on the length of the optimal path. The converse is not true (a heuristic can be admissible but not consistent): find a counter-example (hint: consider a graph that is a path from the initial state to the goal state).

11. Prove that A* without remembering nodes (i.e., without a closed list) is optimal when using an admissible heuristic.

Answer

The main idea of the proof is that at any point in the tree search ((i.e., not remembering nodes)) if a non-optimal goal node G is in the open list and a node N in the path to an optimal goal node G^* is in the open list, then N will be expanded first. This goes all the way towards G^* which ultimately will also be expanded before non-optimal node G.

See a sketchy but quite detailed proof that I did here. You can now make it step by step and reconstruct the reasoning!

It is important that we are relying on TREE SEARCH (i.e., not remembering nodes), so that if we get to a previously visited state x in a cheaper way, we keep and process it, as this could be the way to the goal!. (If we use a CLOSED list, i.e., GRAPH SEARCH, then x would be ignored and hence optimality lost (unless heuristic is monotonic/consistent)

COSC1125/1127 Artificial Intelligence

School of Science (Computer Science)
RMIT University
Semester 1, 2018
A/Prof. Sebastian Sardina

Tutorial Sheet 5
Propositional Logic

We use $\Rightarrow$ to denote logical implication and $\Leftrightarrow$ to denote logical equivalence. Other symbols typically used for these logical connectors are $\rightarrow$ and $\supset$ for logical implication, and $\equiv$ for logical equivalence.

1. Use truth tables to show that the following are valid (i.e. that the equivalences hold).

$P \wedge (Q \vee R) \Leftrightarrow (P \wedge Q) \vee (P \wedge R)$	Distribution of $\wedge$
$\neg(P \wedge Q) \Leftrightarrow \neg P \vee \neg Q$	de Morgan's Law
$\neg(P \vee Q) \Leftrightarrow \neg P \wedge \neg Q$	de Morgan's Law
$P \Rightarrow Q \Leftrightarrow \neg Q \Rightarrow \neg P$	Contraposition
$P \Rightarrow Q \Leftrightarrow \neg P \vee Q$	

2. For each of the following sentences, decide whether it is **valid**, **unsatisfiable**, or **neither**. Firstly, trying "guessing" the answer; then evaluate each properly (e.g. using truth tables). How did your guesses match up?

   (a) $Smoke \Rightarrow Smoke$

Answer

Basically, just draw the truth table for each sentence - if the whole column for the sentence is T, then it is valud, if the whole column for the sentence is F, then it is unsatisfiable; neither otherwise. Eg.:

S	$S \Rightarrow S$
T	T
F	T

Therefore, it is valid.

   (b) $Smoke \Rightarrow Fire$

Answer

S	F	$S \Rightarrow F$
T	T	T
T	F	F
F	T	T
F	F	T

Satisfiable.

(c) $(Smoke \Rightarrow Fire) \Rightarrow (\neg Smoke \Rightarrow \neg Fire)$

Answer

$A : (S \Rightarrow F)$
$B : (\neg S \Rightarrow \neg F)$

S	F	$\neg S$	$\neg F$	A	B	$A \Rightarrow B$
T	T	F	F	T	T	T
T	F	F	T	F	T	T
F	T	T	F	T	F	F
F	F	T	T	T	T	T

Satisfiable.

(d) $Smoke \vee Fire \vee \neg Fire$

Answer

$A : (S \vee F)$

S	F	$\neg F$	A	$S \vee A$
T	T	F	T	T
T	F	T	T	T
F	T	F	T	T
F	F	T	F	T

Valid.

(e) $((Smoke \wedge Heat) \Rightarrow Fire) \Leftrightarrow ((Smoke \Rightarrow Fire) \vee (Heat \Rightarrow Fire))$

Answer

$A : (S \wedge H)$
$B : (A \Rightarrow F)$
$C : (S \Rightarrow F)$
$D : (H \Rightarrow F)$
$E : (C \vee D)$

S	F	H	A	B	C	D	E	$B \Rightarrow E$
T	T	T	T	T	T	T	T	T
T	T	F	F	T	T	T	T	T
T	F	T	T	F	F	F	F	T
T	F	F	F	T	F	T	T	T
F	T	T	F	T	T	T	T	T
F	T	F	F	T	T	T	T	T
F	F	T	F	T	T	F	T	T
F	F	F	F	T	T	T	T	T

Valid.

(f) $(Smoke \Rightarrow Fire) \Rightarrow ((Smoke \wedge Heat) \Rightarrow Fire)$

Answer

S	F	H	$S \Rightarrow F$	$S \wedge H$	$S \wedge H \Rightarrow F$	$(S \Rightarrow F) \Rightarrow (S \wedge H \Rightarrow F)$
T	T	T	T	T	T	T
T	T	F	T	F	T	T
T	F	T	F	T	F	T
T	F	F	F	F	T	T
F	T	T	T	F	T	T
F	T	F	T	F	T	T
F	F	T	T	F	T	T
F	F	F	T	F	T	T

This it is a validity (or tautology) as it holds in every possible interpretation.
To see it from another angle, suppose we want to prove that the implication $(S \Rightarrow F) \Rightarrow (S \wedge H \Rightarrow F)$ is true always, that is, it is a validity/tautology:

- Since it is an implication, the only possible way this is true is if $(S \Rightarrow F)$ is true, but $(S \wedge H \Rightarrow F)$ is false.
- Now for $(S \wedge H \Rightarrow F)$ to be false, $S \wedge H$ has to be true and F false.
- But if $S \wedge H$ is true, then S is true.
- Because we already assumed in the first item that $(S \Rightarrow F)$, then together with S being true, we know that F has to be true, which contradicts our second item.
- Hence, the whole implication cannot actually be made false, that is, it is a validity: always true in every possible interpretation.

(g) $Big \vee Dumb \vee (Big \Rightarrow Dumb)$

Answer

$X : (B \vee D)$
$Y : (B \Rightarrow D)$

B	D	X	Y	$X \vee Y$
T	T	T	T	T
T	F	T	F	T
F	T	T	T	T
F	F	F	T	T

Valid.

(h) $(Big \wedge Dumb) \vee \neg Dumb$

Answer

$X : (B \wedge D)$

B	D	$\neg D$	X	$X \vee \neg D$
T	T	F	T	T
T	F	T	F	T
F	T	F	F	F
F	F	T	F	T

Satisfiable.

3. Represent the following sentences in propositional logic. Can you prove that the unicorn is mythical? What about magical? Horned?

   If the unicorn is mythical, then it is immortal, but if it is not mythical, then it is a mortal mammal. If the unicorn is either immortal or a mammal, then it is horned. The unicorn is magical if it is horned.

Answer

You can translate the sentences into the following propositional logic expressions:

(a) $mythical \Rightarrow \neg mortal$

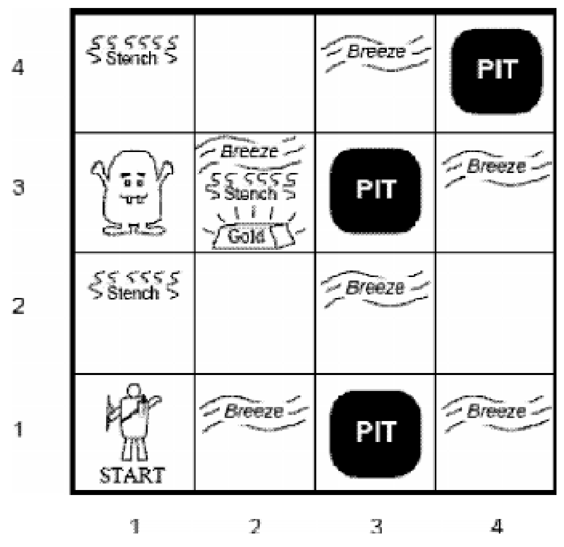
(b) $\neg mythical \Rightarrow mortal \wedge mammal$

(c) $\neg mortal \vee mammal \Rightarrow horned$

(d) $horned \Rightarrow magical$

From statements (a) and (b), we see that if it is mythical, then it is immortal; otherwise it is a mammal. So it must be either immortal or a mammal, and thus horned. That means it is also magical. However we cannot deduce anything about whether it is mythical.

4. For the following Wumpus world:



(a) Develop a notation capturing the important propositions.

Answer

$S_{xy}, W_{xy}, B_{xy}, G_{xy}$

The above denote a sentence, “there is a Stench/Wumpus/Breeze/Gold in square [x,y].

(b) How would you express in a propositional logic sentence:

i. If square $[2, 2]$ has no smell then the Wumpus is not in this square or any of the adjacent squares?

Answer

$$\neg S_{22} \Rightarrow \neg W_{22} \wedge \neg W_{21} \wedge \neg W_{12} \wedge \neg W_{32} \wedge \neg W_{23} \tag{1}$$

ii. If there is stench in square $[1, 2]$ there must be a Wumpus in this square or any of the adjacent squares?

Answer

$$S_{12} \Rightarrow W_{11} \vee W_{12} \vee W_{22} \vee W_{13} \tag{2}$$

(c) How can the agent deduce that the Wumpus is in square $[1, 3]$ using the laws of inference in propositional logic.

Answer

Given that $\neg S_{22} \wedge S_{12}$
Modus Ponens and *And Elimination* applied to the above Equation 1 gives:

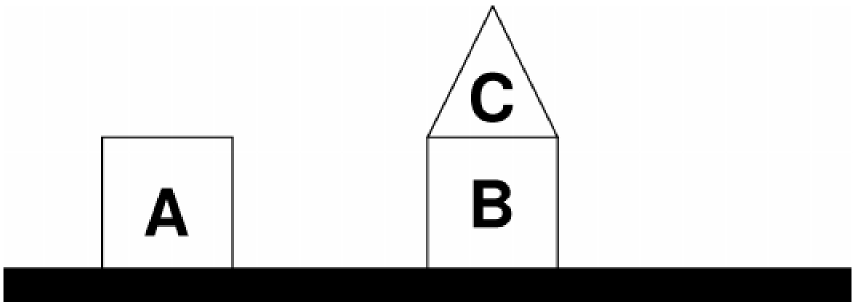
$$\neg W_{22}, \neg W_{21}, \neg W_{12}, \neg W_{32}, \neg W_{23}, \tag{3}$$

Modus Ponens applied to Equation 2 gives:

$$W_{11} \vee W_{12} \vee W_{22} \vee W_{13}, \tag{4}$$

Unit Resolution is applied to Equation 4 based on Equation 3, and also we know the agent has been to square $[1, 1]$, hence $\neg W_{11}$. Therefore, W_{13}

5. Represent the following scene in propositional calculus.



Answer

One example:

$A_{ontable} \wedge B_{ontable} \wedge C_{onB} \wedge A_{square} \wedge B_{square} \wedge C_{triangle}$

6. Consider a knowledge base build of just these three weird implications:

$$\neg A \Rightarrow B$$
$$B \Rightarrow A$$
$$A \Rightarrow (C \wedge D)$$

(a) Prove formula $A \wedge C \wedge D$ using Modus Ponens only, or explain why this is not possible.

Answer

It is not possible Dusing Modus Ponens only: Modus Ponens is not applicable to any pair of formulas in the knowledge base.

(b) Prove formula $A \wedge C \wedge D$ using resolution.

Answer

Proof by refutation:

1. $A \vee B$ premise.
2. $\neg B \vee A$ premise.
3. $\neg A \vee C$ premise.
4. $\neg A \vee D$ premise.
5. $\neg A \vee \neg C \vee \neg D$ negated thesis.
6. A resolution 1, 2.
7. C resolution 3, 6
8. D resolution 4, 6.
9. $\neg C \vee \neg D$ resolution 5, 6.
10. $\neg D$ resolution 7, 9.
11. [] resolution 8, 10

This can be depicted as a resolution tree, draw it! :-)

7. Given the following symbols and sentences:

C to indicate that Gianni is a climber;
F to indicate that Gianni is fit; L to indicate that Gianni is lucky;
E to indicate that Gianni climbs mount Everest.

(a) Formalize the above sentences in propositional logic:

If Gianni is a climber and he is fit, he climbs mount Everest.
If Gianni is not lucky and he is not fit, he does not climb mount Everest.
Gianni is fit.

Answer

$(C \wedge F) \Rightarrow E$
$(\neg L \wedge \neg F) \Rightarrow \neg E$
F

(b) Tell if the KB buit in above is consistent, and tell if some of the following sets are models for the above sentences:
$\{\}$; $\{C, L\}$; $\{L, E\}$; $\{F, C, E\}$; $\{L, F, E\}$.

Answer

The KB is consistent: it has at least a model, as the following check shows.

- $\{\}$ is not a model (it models 1 and 2 but not 3.
- $\{C, L\}$ is not a model (it models 1 and 2 but not 3).
- $\{L, E\}$ is not a model (it models 1 and 2 but not 3).
- $\{F, C, E\}$ is a model.
- $\{L, F, E\}$ is a model.

8. Tell whether the propositional formula $[(A \Rightarrow C) \vee (B \Rightarrow C)] \Rightarrow [(A \wedge B) \Rightarrow C]$ is:

   (a) satisfiable;

   (b) valid;

   (c) a contradiction.

Answer

Try validity first, as if it is valid, you have also proved that it is satisfiable and not contradictory. The formula is valid, and you can prove it with resolution.

9. Do (and test online!) Exercises 5.1, 5.2, and 5.3 in `http://intrologic.stanford.edu/notes/chapter_05.html`

10. Let A, B, C be propositional symbols. Given $KB = \{A \Rightarrow C, B \Rightarrow C, A \vee B\}$, tell whether C can be derived from KB or not. Use resolution.

Answer

C can be derived with Resolution:

(a) $\neg A \vee C$.

(b) $\neg B \vee C$.

(c) $A \vee B$.

(d) $\neg C$ negated thesis

(e) $B \vee C$ from 1 and 3.

(f) C from 2 and 5.

(g) $\{\}$ from 4 and 6.

11. Heads, I win. Tails, you lose. Use propositional resolution to prove that I always win.

Answer

We use H and T to signal heads or tails, resp. Also I_W and I_L to denote I win or lose, respectively, and Y_W and Y_L to denote you win or lose, resp. To formalize the problem we have:

(a) I win iff I don't lose: $I_W \Leftrightarrow \neg I_L$.

(b) You win iff you don't lose: $Y_W \Leftrightarrow \neg Y_L$.

(c) Coin either tails or heads: $H \vee T$.

(d) Zero-sum game: $I_W \Leftrightarrow Y_L$.

(e) "Heads, I win": $H \Rightarrow I_W$.

(f) "Tails, you lose": $T \Rightarrow Y_L$.

We need to prove that I always win, that is, that I_W is entailed by the above formulas. We convert all the above to clausal form and then do resolution with those clauses plus $\neg I_W$ and arrive to empty clause.

12. Do (and test online!) Exercises 5.4 in http://intrologic.stanford.edu/notes/chapter_05.html

13. There are three suspects for a murder: Adams, Brown, and Clark. Adams saysI didn't do it. The victim was old acquaintance of Brown's. But Clark hated him. Brown states I didn't do it. I didn't know the guy. Besides I was out of town all the week. Clark saysI didn't do it. I saw both Adams and Brown downtown with the victim that day; one of them must have done it. Assume that the two innocent men are telling the truth, but that the guilty man might not be. Write out the facts as sentences in Propositional Logic, and use propositional resolution to solve the crime.

COSC1125/1127 Artificial Intelligence

School of Science (Computer Science)
RMIT University
Semester 1, 2018
A/Prof. Sebastian Sardina

Tutorial Sheet 6
First Order Logic

For some of the questions below, you may want to check the great slides by Torsten Hahmann (CSC 384 at University of Toronto) on Skolemization, Most General Unifiers, First-Order Resolution. It includes the Skolemization process to remove existential quantification, the algorithm for finding MGUs, and FOL resolution; all with great fully detailed examples!

1. Choose a vocabulary of predicates, constants and functions appropriate for representing the following information in First Order Logic, then represent each sentence in FOL.

   (a) "There is someone who is loved by everybody"

   (b) "All cats are lazy"

   (c) "Some students are clever"

   (d) "No student is rich"

   (e) "Every man loves a woman"
   (represent both meanings)

   (f) "Everything is bitter or sweet"

   (g) "Everything is bitter or everything is sweet"

   (h) "Martin has a new bicycle"

   (i) "Lynn gets a present from John, but she doesn't get anything from Peter"

Answer

(a) $(\exists y)(\forall x) loves(x, y)$

(b) $(\forall x)(cat(x) \Rightarrow lazy(x))$

(c) $(\exists x)(student(x) \wedge clever(x))$

(d) $(\neg\exists x)(student(x) \wedge rich(x))$
or
$(\forall x)(student(x) \Rightarrow \neg rich(x))$
or
$(\forall x)\neg(student(x) \wedge rich(x))$

(e) $(\forall x)(man(x) \Rightarrow (\exists y) woman(y) \wedge loves(x, y))$ (each man will have its lover)
or
$(\exists y)(woman(y) \wedge (\forall x)(man(x) \Rightarrow loves(x, y))$ (meaning that there is a woman that every man loves)

(f) $(\forall x)(bitter(x) \vee sweet(x))$

(g) $(\forall x) bitter(x) \vee \forall y sweet(y)$

(h) $(\exists x)(bike(x) \wedge new(x) \wedge has(Martin, x))$

(i) $(\exists x)(present(x) \wedge gets(Lynn, x, John)) \wedge$
$((\neg\exists y)(present(y) \wedge gets(Lynn, y, Peter)) \wedge (x \neq y)$

2. Do the same as for the previous question.

(a) Anyone who is rich is powerful.
(b) Anyone who is powerful is corrupt.
(c) Anyone who is meek and has a corrupt boss is unhappy.
(d) Not all students take both Computing-Theory and OO-Programming.
(e) Only one student failed Computing-Theory.
(f) Only one student failed both Computing-Theory and OO-Programming.
(g) The best score in Computing-Theory was better than the best score in OO-Programming.
(h) Every person who dislikes all donkeys is smart.
(i) There is a woman who likes all men who are not footballers.
(j) There is a barber who shaves all men who do not shave themselves.
(k) No person likes a lecturer unless the lecturer is smart.
(l) Politicians can fool some of the people all of the time, and they can fool all of the people some of the time, but they can't fool all of the people all of the time.

Answer

(a) $(\forall x)(rich(x) \Rightarrow powerful(x))$

(b) $(\forall x)(powerful(x) \Rightarrow corrupt(x))$

(c) $(\forall x)(\exists y)(meek(x) \wedge isBoss(y,x) \wedge corrupt(y) \Rightarrow unhappy(x))$

(d) $(\neg\forall x)(student(x) \Rightarrow (takes(x,CT) \wedge takes(x,OO)))$

(e) $(\exists x)student(x) \wedge fails(x,CT) \wedge (\forall y)student(y) \wedge fails(y,CT) \Rightarrow x = y$

In this example, an **equality symbol** is used to make a statment about the effect that two terms refer to the same object. For the above first sentence,

$$(\exists x)student(x) \wedge fails(x,CT) \wedge (\forall y)student(y) \wedge fails(y,CT)$$

would only assert that all instances of y and there is at least one x that failed CT, but nothing says that x and y are referring to the same object.
you can imagine that you run a loop going over all possible y values, and there is an x value where $x = y$ is true. When this is the case, the implication stands.
Note that in a knowledge base, the existential quantifier is eliminated by replacing it with specific instances. Based on this assumption, the above sentence will refer to only one specific student who failed CT.

(f) $(\exists x)(student(x) \wedge fails(x,CT) \wedge fails(x,OO)) \wedge$
$(\forall y)(student(y) \wedge fails(y,CT) \wedge fails(y,OO)) \Rightarrow x = y$

(g) $(\exists x)(\forall y)score(x,CT) \wedge score(y,OO) \Rightarrow better(x,y)$

(h) $(\forall x)person(x) \wedge ((\forall y)donkey(y) \Rightarrow dislikes(x,y)) \Rightarrow smart(x)$

(i) $(\exists x)woman(x) \wedge ((\forall y)man(y) \wedge \neg footballer(y)) \Rightarrow likes(x,y)$

(j) $(\exists x)barber(x) \wedge ((\forall y)man(y) \wedge \neg shaves(y,y)) \Rightarrow shaves(x,y)$

(k) $(\forall x)(\forall y)person(x) \wedge lecturer(y) \wedge \neg smart(y) \Rightarrow \neg likes(x,y)$

(l) $(\forall x)politician(x) \Rightarrow$
$((\exists y)(\forall t)person(y) \wedge fools(x,y,t)) \wedge$
$((\exists t)(\forall y)person(y) \wedge fools(x,y,y)) \wedge$
$\neg((\forall t)(\forall y)person(y) \Rightarrow fools(x,y,t))$

3. Consider the following story:

I married a widow (let's call her w) who has a grown up daughter (d). My father (f), who visited us quite often, fell in love with my step-daugher and married her. Hence my father became my son-in-law and my step-daughter became my mother. Some months later, my wife gave birth to a son (s_1), who became the brother-in-law of my father, as well as my uncle. The wife of my father, that is, my step-daughter, also had a son (s_2).

Using predicate calculus, create a set of expressions that represent the situation in the above story. Extend the kinship relations to in-laws and uncle. Use generalised modus ponens to prove 'I am my own grandfather'.

Answer

This works fine (with a little violation of the accepted semantics for family relationships). Here is an example showing how we can prove this; we can have the following rules (among others):

$$(\forall x)(\forall y) child(x, y) \iff parent(y, x) \tag{1}$$
$$(\forall x)(\forall y)(\exists z) grandparent(x, y) \iff parent(x, z) \land parent(z, y) \tag{2}$$

From the given story we can get the following facts (once again, among others):

$$parent(F, I) \text{ (since } F \text{ is my father),} \tag{3}$$

$$child(F, I) \text{ (since my father becomes my son-in-law).} \tag{4}$$

Some of the other rules are: ($parent(W, D), spouse(I, W), spouse(F, D)$, etc), but we don't really need them since we can already prove our conclusion.

From 1 and 4 (take $y = F$ and $x = I$) we derive that

$$parent(I, F) \tag{5}$$

Finally, from 3 and 5 we derive (take $x = I, y = I$ and $z = F$):

$$grandparent(I, I)$$

Thus, "I am my own grandfather"!

4. Give the most general unifier (MGU) for the following pairs of expressions, or say why it does not exist. In all of these expressions variables are upper case x, y, or z.

   (a) $p(a, b, b), p(x, y, z)$.
   (b) $q(y, g(a, b)), q(g(x, x), y)$.
   (c) $older(father(y), y), older(father(x), john)$.
   (d) $knows(father(x), y), knows(x, x)$.
   (e) $r(f(a), b, g(f(a))), r(x, y, z)$.
   (f) $older(father(y), y), older(father(y), john)$.
   (g) $f(g(a, h(b)), g(x, y)), f(g(z, y), g(y, y))$.

Answer

(a) $\{x/a, y/b, z/b\}$. Observe that $\theta' = \{x/a, y/b, z/y\}$ is not an MGU because it is not even a unification of both structures. In fact, $p(a,b,b)\theta' = p(a,b,c) \neq p(x,y,z)\theta' = p(a,b,y)$.

(b) No unifier (x cannot bind to both a and b).

(c) $\{y/john, x/john\}$.

(d) No unifier, because the occurs-check prevents unification of y with $father(y)$.

(e) $\{x/f(a), y/b, z/g(f(a))\}$.

(f) $\{y/john\}$.

(g) $\{x/y, z/a, y/h(b)\}$.

Some more (y, w, z, v, u are all variables):

(a) $p(f(y), w, g(z)), p(v, u, v)$.

(b) $p(f(y), w, g(z)), p(v, u, x)$.

(c) $p(a, x, f(g(y))), p(z, h(w), f(w))$.

(d) $p(z, h(w), g(z)), p(v, u, v)$.

(e) $p(a, h(w), f(g(y))), p(z, x, f(w))$.

Answer

(a) Impossible as v has to match both $f(y)$ and $g(z)$ and f and g are different symbols.

(b) $\{v/f(y), u/w, x/g(z)\}$.

(c) $\{z/a, x/h(w), w/g(y)\}$.

(d) Impossible as v has to unify with both z and $g(z)$.

(e) $\{z/a, x/h(w), w(g(y))\}$.

5. Confirm your answers to the previous question using Prolog predicate `unifiable/3`? Try it! Do you notice anything special in the fourth query? (<u>Hint:</u> read about occurs check).

Answer

(a)
```
?- unifiable(p(a,b,b),p(X,Y,Z),U).
U = [Z=b, Y=b, X=a].
```

Another way to check this is asking for query `?- p(a,b,b) = p(X,Y,Z)`. This will give the following result:

```
X = a,
Y = Z, Z = b.
```

Note that the result states that for legibility SWI prints all the variables that are the same in one line. So, the line:

```
Y = Z, Z = b.
```

means that `Y = b` and `Z = b`.

To get the concrete MGU without any extra printing process, we use `unifiable/3`

(b)
```
?- unifiable(q(Y,g(a,b)), q(g(X,X),Y),U).
false.
```

(c)
```
?- unifiable(older(father(Y),Y), older(father(X),john),
U).
U = [Y=john, X=Y].
```

(d)
```
unifiable(knows(father(X),Y), knows(X,X), U).
U = [Y=father(X), X=father(X)].
```

Observe that this succeeds because SWI has occurs check disabled by default. We can unify with occurs check as follows:

```
?- unify_with_occurs_check(knows(father(X),Y),
knows(X,X)).
false.
```

Moreover, we can enable occurs check by default as follows:

```
?- set_prolog_flag(occurs_check,true).
true.
?- knows(father(X),Y) = knows(X,X).
false.
?- unifiable(knows(father(X),Y), knows(X,X), U).
false.
```

(e)
```
?- unifiable(r(f(a), b, g(f(a))), r(X, Y, Z), U).
U = [Z=g(f(a)), Y=b, X=f(a)].
```

(f)
```
?- unifiable(older(father(Y), Y), older(father(Y),
john), U).
U = [Y=john].
```

(g)
```
?- unifiable(f(g(a, h(b)), g(X, Y)), f(g(Z,Y), g(Y,
Y)), U).
U = [X=h(b), Y=h(b), Z=a].
```

6. Write down logical representations for the following sentences, suitable for use with generalised modus ponens. Pay special attention to implications and quantification.

   (a) Horses, cows and pigs are mammals.

   (b) An offspring of a horse is a horse.

   (c) Bluebeard is a horse.

   (d) Bluebeard is Charlie's parent.

   (e) Offspring and parent are inverse relations.

   (f) Every mammal has a parent.

Answer

$$\forall x[Horse(x) \supset Mammal(x)]$$
$$\forall x[Cow(x) \supset Mammal(x)]$$
$$\forall x[Pig(x) \supset Mammal(x)]$$

$$\forall x, y[Offspring(x, y) \wedge Horse(y) \supset Horse(x)]$$

$$Horse(bluebeard)$$

$$Parent(bluebeard, charlie)$$

$$\forall x, y[Offspring(x, y) \supset Parent(y, x)]$$
$$\forall x, y[Parent(x, y) \supset Offspring(y, x)$$

$$\forall x, \exists y[Mammal(x) \supset Parent(y, x)]$$

7. Using the logical expressions from the previous question do the following:

   (a) Draw the proof tree using backward chaining (that is, starting from your query) for the query $(\exists x)Horse(x)$.

   (b) How many solutions for H are there?

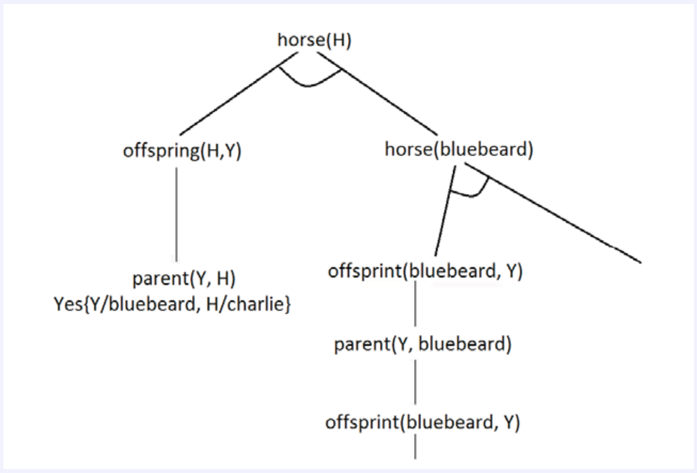
   (c) Repeat the proof using forward chaining (that is, start with the facts and derive further conclusions). Do you get the the same results as before?

   (d) Translate (a)-(f) logic formulas into Prolog.

   (e) Translate $(\exists x)Horse(x)$ into a Prolog query. What answers do you expect Prolog to return and how? Try it!

Answer

(a) The proof tree is shown below. The branch with $Offspring(bluebeard, y)$ and $Parent(y, bluebeard)$ repeats indefinitely, so the rest of the proof is never reached.



(b) Both $bluebeard$ and $charlie$ are horses.

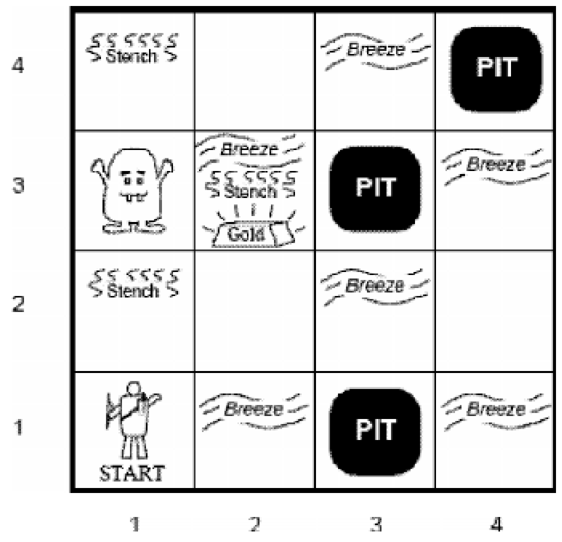
(c) yes.

8. A popular children's riddle is *"Brothers and sisters I have none, but that man's father is my father's son"*. Use the rules of the kinship domain to work out who that man is.

Answer

The son (of the man speaking).

9. For the following Wumpus world:



(a) Develop a notation capturing the important aspects of this domain in first order logic.

(b) How would you express in a first order logic sentence:

i. If a square has no smell then the Wumpus is not in this square or any of the adjacent squares?

ii. If there is stench in a square there must be a Wumpus in this square or in one of the adjacent squares?

(c) Suppose the agent has traversed the path $(1,1) \rightarrow (1,2) \rightarrow (2,2) \rightarrow (1,2)$. How can the agent deduce that the Wumpus is in square [1,3] using the laws of inference of first order logic?

Answer

(a) Similar to previous practical for propositional case but now stated using first order.

(b) i. $\forall x[\neg Smell(x) \supset \forall y((Adjacent(x,y) \vee x = y) \supset \neg WumpusAt(y))]$.

ii. $\forall x[Smell(x) \supset \exists y((Adjacent(x,y) \vee x = y) \wedge WumpusAt(y))]$.

(c) It follows from (ii) above and the fact that the knowledge base includes or entails:

$$\begin{array}{l} Adjacent((1,3),(1,2)) \\ Smell((1,2)) \\ \neg WumpusAt((1,1)) \\ \neg Smell((2,2)) \text{(or directly } \neg WumpusAt((2,2))) \end{array}$$

10. **Resolution Proofs.** Consider the following sentences:

(a) Marcus was a man.

(b) Marcus was a Roman.

(c) All men are people.

(d) Caesar was a ruler.

(e) All Romans were either loyal to Caesar or hated him (or both).

(f) Everyone is loyal to someone.

(g) People only try to assassinate rulers they are not loyal to.

(h) Marcus tried to assassinate Caesar.

Carry out the following tasks (refer to book and/or slides from Lecture 5):

(a) Convert each of these sentences into first-order logic and then convert each formula into clausal form. Indicate any Skolem functions or constants used in the conversion.

(b) Convert the negation of the question "Who hated Caesar?" into causal form (with an answer literal)

(c) Derive the answer to this question using resolution. Give the answer in English. In the proof use the notation developed in class.

Answer

First-order sentences:

(a) $Man(marcus)$

(b) $Roman(marcus)$

(c) $\forall x. Man(x) \supset Person(x)$.

(d) $Ruler(caesar)$.

(e) $\forall x. Roman(x) \supset Loyal(x, caesar) \vee Hate(x, caesar)$.

(f) $\forall x. \exists y. Loyal(x, y)$

(g) $TryKill(x, y) \supset Ruler(y) \wedge \neg Loyal(x, y)$

(h) $TryKill(marcus, caesar)$.

Conversions to CNF:

(a) $Man(marcus)$

(b) $Roman(marcus)$

(c) $\neg Man(x) \vee Person(x)$.

(d) $Ruler(caesar)$.

(e) $\neg Roman(x) \vee Loyal(x, caesar) \vee Hate(x, caesar)$.

(f) $Loyal(x, f(x))$ ($f(x)$ is a Skolem function)

(g) $(\neg TryKill(x, y) \vee Ruler(y))$ and $(\neg TryKill(x, y) \vee \neg Loyal(x, y))$.

(h) $TryKill(marcus, caesar)$.

Query: $\exists z. Hate(z, caesar)$.
Query with answer predicate: $\exists z.[Hate(z, caesar) \wedge \neg A(z)]$.
Negation of query with answer predicate: $\neg Hate(z, caesar) \vee A(z)$

The resolution follows easily until we get a clause $A(marcus)$: Marcus hated Caesar.

11. Determine whether the following sentence is valid (i.e., a tautology) using FOL Resolution:

$$\exists x \forall y \forall z ((P(y) \Rightarrow Q(z)) \Rightarrow (P(x) \Rightarrow Q(x))).$$

Answer

You do it! Convert this first to CNF to get a set of clauses (with no existential quantification), then do resolution...

12. You say more examples and exercises on resolution? Check here:

https://www.cs.utexas.edu/users/novak/reso.html

Sample Solutions for COSC1125/1127 Tutorial Week 7

The block world (fig. 7.18) may be represented by the following set of predicates:

STATE 1:
ontable(a) ∧ ontable(c) ∧ ontable(d) ∧ on(b,a) ∧ on(e,d) ∧ gripping() ∧ clear(b) ∧ clear(c) ∧ clear(e)

There are then a number of rules (or operators) to operate on states and produce new states. These operators typically include pickup, putdown, stack, and unstack. For example the unstack operator can be described as:

Rule1:
(∀X)(∀Y)(unstack(X,Y) → ((clear(Y) ∧ gripping(X)) ← (on(X,Y) ∧ (clear(X) ∧ gripping())).

If we apply unstack(e,d), a new state will be:

STATE 2:
ontable(a) ∧ ontable(c) ∧ ontable(d) ∧ on(b,a) ∧ **gripping(e)** ∧ clear(b) ∧ clear(c) ∧ clear(e) ∧ **clear(d)**

In the above we can see as a result of applying the operator, new predicates have been added to the new state, whereas other predicates such as on(e,d) ∧ gripping() have been deleted from the new state because they are no longer valid. Operators such as this can be also described by using STRIPS.

You will notice that there are also predicates such as "ontable(a) ∧ ontable(c) ∧ ontable(d)" continue to remain true in the new state. The question here is how we can ensure this is indeed the case as we generate more and more new states. This is where frame axioms can be used.

Frame axioms are rules to tell what predicates describing a state are not changed by rule applications and are thus carried over intact to help describe the new state of the world. An example of frame axioms is:

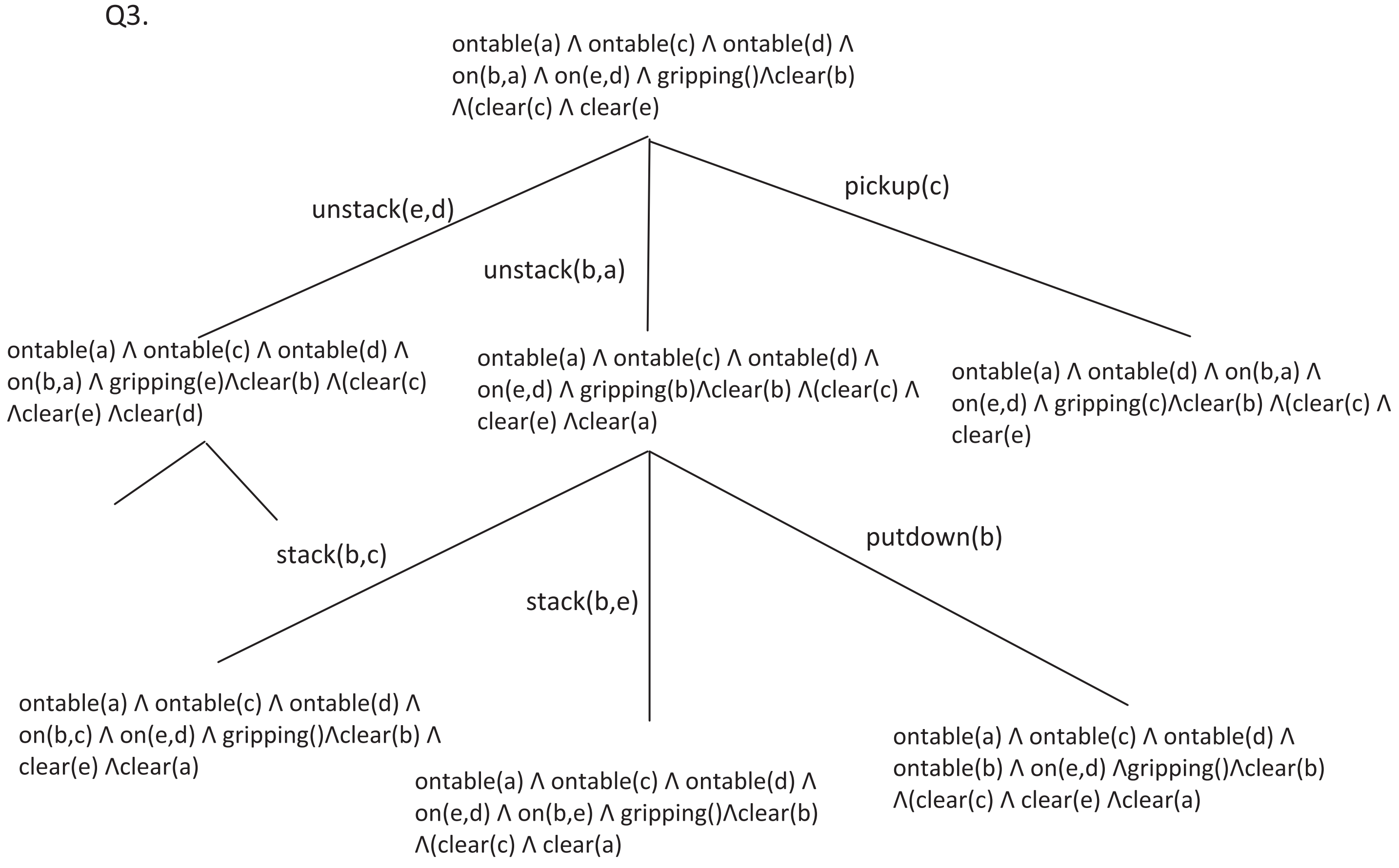
Rule 2:
(∀X)(∀Y) (∀Z)(unstack(Y,Z) → (ontable(X) ←ontable(X))).

The above rule says *ontable* is not affected by the *unstack* operator. By using this frame rule and the operator *unstack*, we can tell that when applying *unstack(e,d)* to STATE 1, we should continue to have "ontable(a) ∧ ontable(c) ∧ ontable(d)" in STATE 2, and update it with new predicates such as **gripping(e) ∧ clear(d)**. Even for a simple block world example like this, we need a number of other frame axioms. For example,

(∀X)(∀Y) (∀Z)(stack(Y,Z) → (ontable(X) ←ontable(X)))
(∀X)(∀Y) (∀Z)(stack(Y,Z) → (clear(X) ←clear(X)))
(∀X) (∀Y)(∀Z)(pickup(X) → (on(Y, Z) ← on(Y,Z)))
(∀X) (∀Y)(pickup(X) → (ontable(Y) ← ontable(Y)))
(∀X) (∀Y)(∀Z)(putdown(X) → (on(Y, Z) ← on(Y,Z)))
(∀X) (∀Y)(putdown(X) → (ontable(Y) ← ontable(Y)))
…
etc.

After creating these axioms (which are straight forward and similar to those already created), we should question the complexity cost of adding this number of support axioms to a system in order to maintain a sound inference scheme. This issue can be addressed by using STRIPS, which maintains a *add* and *delete* list, to keep track of new predicates that should be added to the new states and as well as old predicates that should be deleted from the state.

Q3.



ontable(a) ∧ ontable(c) ∧ ontable(d) ∧ on(b,a) ∧ on(e,d) ∧ gripping()∧clear(b) ∧(clear(c) ∧ clear(e)
unstack(e,d)
unstack(b,a)
pickup(c)
ontable(a) ∧ ontable(c) ∧ ontable(d) ∧ on(b,a) ∧ gripping(e)∧clear(b) ∧(clear(c) ∧clear(e) ∧clear(d)
ontable(a) ∧ ontable(c) ∧ ontable(d) ∧ on(e,d) ∧ gripping(b)∧clear(b) ∧(clear(c) ∧ clear(e) ∧clear(a)
ontable(a) ∧ ontable(d) ∧ on(b,a) ∧ on(e,d) ∧ gripping(c)∧clear(b) ∧(clear(c) ∧ clear(e)
stack(b,c)
stack(b,e)
putdown(b)
ontable(a) ∧ ontable(c) ∧ ontable(d) ∧ on(b,c) ∧ on(e,d) ∧ gripping()∧clear(b) ∧ clear(e) ∧clear(a)
ontable(a) ∧ ontable(c) ∧ ontable(d) ∧ on(e,d) ∧ on(b,e) ∧ gripping()∧clear(b) ∧(clear(c) ∧ clear(a)
ontable(a) ∧ ontable(c) ∧ ontable(d) ∧ ontable(b) ∧ on(e,d) ∧gripping()∧clear(b) ∧(clear(c) ∧ clear(e) ∧clear(a)

The STRIPS representation of the operators:

	P: gripping() ∧clear(X) ∧ontable(X)
pickup(X):	A: gripping(X)
	D: ontable(X) ∧ gripping()
	P: gripping(X)
putdown(X):	A: ontable(X) ∧ gripping()∧ clear(X)
	D: gripping(X)
	P: clear(Y) ∧gripping(X)
stack(X,Y):	A: on(X,Y) ∧ gripping() ∧ clear(X)
	D: clear(Y) ∧ gripping(X)
	P: clear(X) ∧gripping()∧ on(X,Y)
unstack(X,Y):	A: gripping(X) ∧ clear(Y)
	D: gripping()∧ on(X,Y)

Q4.
a)
In a goal driven search, we know the goal state to start with. We can apply different operators to the goal state, and see what new states it can generate. In this case since the goal state is only two steps away from the initial state, you can easily identify that first unstack(e,c) and then secondly stack(e,d) will form a path to the initial state, therefore the reversed path would be the plan to achieve the required task. The plan is unstack(e,d) and stack(e,c).

In a data driven search, since we know the initial state for start, we can always try different operators, leading to different new states. From each new state, we can try these operators again, eventually we can identify the goal state. Since this is a rather small search space, a Breath-First-Search would be sufficient to find the path to the goal easily.

b)
Similar to the above, however the search space is larger than the previous example (it takes at least 8 steps to get to the goal state). It would be more appropriate to use heuristic search (e.g., measuring the distance of the current state from the goal state, and also checking if it is a repeated state), in order to guide the search towards the goal state.

Note for b) it can get really complicated in order to find the correct plan. For example we need to check if the preconditions of an operator are met or not, and the problem of having incompatible subgoals can also emerge. This goes beyond the scope of this subject.

Q5. Nothing really, because the property of color is not required for any preconditions of the operators.

Q6. similar to Q4. A plan such as the following can be generated using the goal-driven search:

unstack(C, A)
putdown(C)
pickup(B)
stack(B,C)
pickup(A)
stack(A,B)

You may use the STRIPS planner to see how the plan is generated.

COSC1125/1127 Artificial Intelligence
School of Science (Computer Science)
RMIT University
Semester 1, 2018
A/Prof. Sebastian Sardina

Tutorial Sheet 7
Markov Decision Processes (MDPs)

1. Recall a Markov Decision Process (MDP) represents a scenario where actions are non-deterministic. Consider the following minor gridworld example:

	1	2	3
1	X		X
2			
3	X		

Cells are indexed by (row, column). Cells $(1, 1)$, $(3, 1)$ and $(1, 3)$ are terminal cells (marked with X), and have rewards of 20, -20 and 5, respectively. For all other cells, there is an reward of -1. The agent starts at cell $(3, 3)$. The agent can move in north, east, south and west directions or stay where it is. The actions are stochastic and have the following outcomes:

- If go north, 70% of the time the agent will go north, 30% will go west.
- If go east, 60% of the time the agent will go east, 40% of the time will go south.
- If go south, 80% of the time the agent will go south, 20% of the time will go east.
- If go west, 70% of the time the agent will go west, 30% of the time will go north.
- If stay in current cell, 100% of the time will stay.

If an action causes the agent to travel into a wall, the agent will stay in their current cell.

Construct a MDP for this instance of gridworld. Recall a MDP consists of S (set of states), s_0 (starting state), possible terminal states, A (set of actions), T (transition model/function) and R (reward function), so all of these have to be specified. To help, consider constructing the MDP in steps:

a) What are the states in this gridworld?

b) What are the starting and terminal states?

c) What are the set of actions?

d) With cell $(3, 3)$ as the current state s, construct the transition model/function for all actions and subsequent states, i.e., $T(s, a, s')$, where a are all possible actions for $(3, 3)$ and s' are all possible successor states from $(3, 3)$.

e) What is the reward function?

Answer

- Each grid corresponds to a state.
- Starting state is $(3,3)$, terminal states are $(1,1)$, $(3,1)$ and $(1,3)$.
- For each state, actions are $\{north, east, south, west, stay\}$:
  - $T((3,3), north, (2,3)) = 0.7$ (70%, go north)
  - $T((3,3), north, (3,2)) = 0.3$ (30%, go west)
  - $T((3,3), east, (3,3)) = 0.6$ (60%, go east)
  - $T((3,3), east, (3,3)) = 0.4$ (40%, go south)
  - $T((3,3), south, (3,3)) = 0.8$ (80%, go south)
  - $T((3,3), south, (3,3)) = 0.20$ (20%, go east)
  - $T((3,3), west, (3,2)) = 0.7$ (70%, go west)
  - $T((3,3), west, (2,3)) = 0.3$ (30%, go north)
  - $T((3,3), stay, (3,3)) = 1.0$ (100%, stay)
- Reward:
  - $R((1,1)) = +20$
  - $R((3,1)) = -20$
  - $R((1,3)) = +5$
  - $R(\text{other states}) = -1$

2. Using the MDP built for question 1, consider the following sequences of states that our agent progressed through. For each sequence, compute the *additive reward* and *discounted reward* (with a decay factor γ of 0.5). Take note of the differences between the two approaches.

   a) $(3,3)$, $(2,3)$, $(2,2)$, $(2,1)$, $(1,1)$
   b) $(3,3)$, $(3,2)$, $(2,2)$, $(2,3)$, $\ldots$ (repeat)

Answer

(a) Additive: $(-1) + (-1) + (-1) + (-1) + (+20) = 16$

Discount: $(-1) + 0.5(-1) + 0.5^2(-1) + 0.5^3(-1) + 0.5^4(+20) = -0.625$ $(10/16)$

(b) <u>Additive:</u> $(-1) + (-1) + (-1) + (-1) + \ldots = -\infty$

<u>Discount:</u> $(-1)+0.5(-1)+0.5^2(-1)+0.5^3(-1)+0.5^4(-1)+\ldots = \sum_{t=0}^{\infty} 0.5^t * (-1) = \frac{-1}{1-0.5} = -2$

3. Consider the MDP from question 1, with a decay factor γ of 1.

a) Using value iteration (check algo here), compute the utilities for each state after two iterations. For this question take (observe we are using the simplest reward scheme $R(s)$ rather than $R(s, a, s')$)):

$$V_{k+1}(s) = R(s) + \max_a \{\Sigma_{s' \in S} T(s, a, s') V_k(s')\}.$$

Initially, we initialize $V_0(s) = 0$ for all states $s \in S$. Remember terminal states do not get more rewards once reached.

b) Repeat and determine the values after three iterations.

Answer

Check this link for the detailed workings on cell $(1, 2)$. Note that, in the case of $(1, 2)$, it is clear that "moving west" is the best action (i.e., maximizes the second term in formula above), but in general one would need to calculate the term for every action and then keep the the max one (that is the best action to do!).

a)

	1	2	3
1	20	12.7	5
2	12.7	−2	2.2
3	−20	−2	−2

b)

	1	2	3
1	20	16.81	5
2	16.81	11.7	1.9
3	−20	−3	−0.059

4. Consider the utilities after two iterations of value iteration from question 3 (we typically only perform this when values have converged, but for this question, do this after 2 iterations). Using the Maximum Expected Utility principle, find the best action (i.e., policy) for each state.

Answer

For the cell $(1, 2)$ we can derive (as explained here) that the best action to do there after 2 iteration is west, that is, $\pi((1, 2)) = $ west.

5. Consider the following policy for the MDP of question 1, with a decay factor γ of 1:

- $\pi((2, 1)) = $ west.
- $\pi((2, 2)) = $ west.
- $\pi((2, 3)) = $ north.
- $\pi((3, 2)) = $ north.
- $\pi((3, 3)) = $ east.
- $\pi((1, 2)) = $ south.

Evaluate this policy, i.e., compute the utility for each state, for one and two iterations (we typically can solve it as a system of simultaneous equations but for this question we use an iterative approach).

Answer

Here we are given the policy (it is fixed), so the task is to calculate its value at every state, that is, $V_\pi(s)$ or $U_\pi(s)$. We can do that as follows:

$$V^\pi_{k+1}(s) = R(s) + \Sigma_{s' \in S} T(s, \pi(s), s') V^\pi_k(s').$$

Observe we do not take the $\max$ of actions because we know, at every state s, what we should do, namely, $\pi(s)$!

BE CAREFUL, THIS HAS NOT BEEN RE-CHECKED CAREFULLY!

	1	2	3
1	20	-0.8	5
2	4.3	-2	-2.2
3	-20	-2	-2

6. Will the stay action be used for an optimal policy? Hint: consider the reward function for non-terminal cells.

Answer

If the reward function is positive valued and large in magnitude for non terminating states, and γ is small, then it is possible for an agent to choose to stay in a state.

COSC1125/1127 Artificial Intelligence

School of Science (Computer Science)
RMIT University
Semester 1, 2018
A/Prof. Sebastian Sardina

Tutorial Sheet 9
Reinforcement Learning

1. In the passive Adaptive Dynamic Programming (ADP) agent, it estimates the transition model T using tables of frequencies. Consider the 2 by 3 gridworld with the policy depicted as arrows and the terminal states are illustrated with X's.

	1	2
1	X	←
2	↑	←
3	X	↑

Compute the transition model after the passive ADP agent sees the following state transition observations:

Trial 1: $(3,2) \to (2,2) \to (2,1) \to (1,1)$
Trial 2: $(3,2) \to (2,2) \to (1,2) \to (1,1)$
Trial 3: $(3,2) \to (2,2) \to (2,1) \to (3,1)$
Trial 4: $(3,2) \to (2,2) \to (2,1) \to (2,2) \to (2,1) \to (1,1)$

With rewards perceived as follows:

State	**Reward**
$(3,2)$	-1
$(2,2)$	-2
$(2,1)$	-2
$(1,2)$	-1
$(3,1)$	-10
$(1,1)$	$+16$

Answer

We go through each trial, and count the frequency of each pair (state-action) and triple (state, action, next state) across all the trials.

Table of state-action frequencies/counts, $N(s, a)$:

state (s)	action (a)	Frequency
$(3, 2)$	N	4
$(2, 2)$	W	5
$(2, 1)$	N	4
$(1, 2)$	W	1

Table of state-action-state frequencies/counts, $N(s, a, s')$:

state (s)	action (a)	next state (s')	Frequency
$(3, 2)$	N	(2,2)	4
$(2, 2)$	W	(2,1)	4
$(2, 2)$	W	(1,2)	1
$(2, 1)$	N	(1,1)	2
$(2, 1)$	N	(3,1)	1
$(2, 1)$	N	(2,2)	1
$(1, 2)$	W	(1,1)	1

From these frequency tables, we can obtain an estimate of T, the transition model.

state (s)	action (a)	next state (s')	T(s,a,s')
$(3, 2)$	N	$(2, 2)$	$4/4 = 1.0$
$(2, 2)$	W	$(2, 1)$	$4/5 = 0.8$
$(2, 2)$	W	$(1, 2)$	$1/5 = 0.2$
$(2, 1)$	N	$(1, 1)$	$2/4 = 0.5$
$(2, 1)$	N	$(3, 1)$	$1/4 = 0.25$
$(2, 1)$	N	$(2, 2)$	$1/4 = 0.25$
$(1, 2)$	W	$(1, 1)$	$1/1 = 1.0$

2. Using the environment model and observations from question 1, now consider how a passive Temporal Difference agent will estimate the utility of the states. Using the first two trials, update the utilities as each observation comes in, using the temporal difference learning rule given in lectures. Use $\alpha = 0.5$ (recall α is the learning rate) and $\gamma = 1$, and for this question, assume α is a constant, i.e., α doesn't change as we visit a state more and more.

Answer

Trial 1:

- Starting state is $(3,2)$, $R[(3,2)] = -1$, $U[(3,2)] = -1$.
- First observation is $(3,2) \rightarrow (2,2)$, reward of $(2,2)$ is -2, hence $U[(2,2)] = -2$.

$$\begin{aligned} U[(3,2)] &= U[(3,2)] + \alpha(R[(3,2)] + \gamma U[(2,2)] - U[(3,2)]) \\ &= -1 + 0.5(-1-2+1) = -2 \end{aligned}$$

- Second observation is $(2,2) \rightarrow (2,1)$, reward of $(2,1)$ is -2, hence $U[(2,1)] = -2$:

$$\begin{aligned} U[(2,2)] &= U[(2,2)] + \alpha(R[(2,2)] + \gamma U[(2,1)] - U[(2,2)]) \\ &= -2 + 0.5(-2-2+2) = -3 \end{aligned}$$

- Third observation is $(2,1) \rightarrow (1,1)$, reward of $(1,1)$ is $+16$, hence $U[(1,1)] = +16$:

$$\begin{aligned} U[(2,1)] &= U[(2,1)] + \alpha(R[(2,1)] + \gamma U[(1,1)] - U[(2,1)]) \\ &= -2 + 0.5(-2+16+2) = 6 \end{aligned}$$

Taking a similar process to trial 2, we obtain the following:

- First observation is $(3,2) \rightarrow (2,2)$. As we visited $(2,2)$ already, no need to update utility of $(2,2)$; Update $U[(3,2)]$:

$$\begin{aligned} U[(3,2)] &= U[(3,2)] + \alpha(R[(3,2)] + \gamma U[(2,2)] - U[(3,2)]) \\ &= -2 + 0.5(-1-3+2) = -3 \end{aligned}$$

- Second observation is $(2,2) \rightarrow (1,2)$, reward of $(1,2)$ is -1, hence $U[(1,2)] = -1$; Update $U[(2,2)]$:

$$\begin{aligned} U[(2,2)] &= U[(2,2)] + \alpha(R[(2,2)] + \gamma U[(1,2)] - U[(2,2)]) \\ &= -3 + 0.5(-2-1+3) = -3 \end{aligned}$$

- Third observation is $(1,2) \rightarrow (1,1)$. As we visited $(2,2)$ already, no need to update utility of $(2,2)$; Update $U[(1,2)]$:

$$\begin{aligned} U[(1,2)] &= U[(1,2)] + \alpha(R[(1,2)] + \gamma U[(1,1)] - U[(1,2)]) \\ &= -1 + 0.5(-1+16+1) = 7 \end{aligned}$$

3. Consider the "occasionally-random" and exploration function methods to strike a balance between exploitation and exploration. Recall in the "occasionally-random" approach, $\frac{1}{t}$ of the time the agent selects a random action, otherwise follow the greedy policy. What would a high t value mean? What about a low value t?

   Contrast this with the exploration function concept. As an example, consider the one taught in lectures and given in the textbook,

$$f(u,n) = \begin{cases} R^+ \text{ , n } < N_e \\ u \text{ , otherwise} \end{cases}$$

What does high/low settings for R^+ and N_e result in?

Answer

For the occasionally-random scheme, a high t value means we do not often select a random action to take, resulting in less exploration. It means the agent is more likely to (greedily) select what it considers is the best policy, i.e., more exploitation, from its current estimation of the environment/world. Conversely a low t value means more focus on exploration over exploitation.

Using the exploration function given in the textbook and lectures, the first case/condition encourages exploration, while second case/condition encourages exploitation. Hence a high value of R^+ means the agent has an optimistic view about unknown or rarely states - the higher it is, the more optimistic. Unvisited states will be set this high value for its utility/Q-values. However, if this view is far from reality, the update rules will bring it back towards the true value, but will take longer. Similarly a small value will mean the agent has a pessimistic view about unknown or rarely visited states. N_e determines for how long we keep in exploration mode. The higher N_e is, the more willing the agent is to explore a state and its associated actions.

4. Consider the grid world from question 1, but now there are no policy specified.

   Apply Q-learning taught in class and textbook to learn Q-Values of all state-action pairs for two trials/episodes (one sequence from starting state to a terminal state). Initialise all Q-values to 0.

   The algorithm in the textbook (and which we follow in this course) does not update the Q-Values of terminal states. Instead, when the next state s' is a terminal state, rather then just setting all previous states, actions and rewards to null, we will additional set all Q-values for that terminal state to its reward. E.g., for state $(1,1)$, its reward is $+16$, and the first time we visit $(1,1)$ we will set $Q(North,(1,1)) = Q(South,(1,1)) = Q(East,(1,1)) = Q(West,(1,1)) = +16$.

   To simplify calculations, for this question assume actions are deterministic, i.e., if agent goes west, it will go west.

   Similar to the TD learning question, use $\alpha = 0.5$ and $\gamma = 1$. For the exploration function, use the one described in question 5, with $R^+ = +10$ and $N_e = 1$. In reality the starting state can vary, but for this question, assume we always start at $(3,2)$. Note that there can be several answers possible, depending on the action selected when there are tie breakers.

Answer

First trial/episode:

- Start at state $(3,2)$, $R[(3,2)] = -1$.

  No previous state, so we don't need to update any Q-values. Instead, given all actions and next states have the same Q-value and we haven't visited any of them yet, we randomly select an action - say west (W) and arrive at $(3,1)$.

- We move to $(3,1)$, so current state is $(3,1)$ and previous state is $(3,2)$. Recall for terminal states, we will update the Q-values for all actions to its reward, but only after updating the Q-value of previous state. Update $Q[W,(3,2)]$:

$$\begin{aligned} Q[W,(3,2)] &= Q[W,(3,2)] + \alpha(R[(3,2)] + \gamma \max_{a'} Q[a',(3,1)] - Q[W,(3,2)]) \\ &= 0 + 0.5(-1+0-0) = -0.5 \end{aligned}$$

  We update $Q[N,(3,1)] = Q[S,(3,1)] = Q[E,(3,1)] = Q[W,(3,1)] = -10$.

Second trial/episode:

- Start at state $(3,2)$, $R[(3,2)] = -1$.

  No previous state, so we don't need to update any Q-values.

  Instead, given all actions and next states have the same Q-value but we haven't gone north from $(3,2)$, we go north (N), and go to $(2,2)$.

- We move to $(2,2)$, so current state is $(2,2)$ and previous state is $(3,2)$. Non-terminal state. Update $Q[N,(3,2)]$:

$$\begin{aligned} Q[N,(3,2)] &= Q[N,(3,2)] + \alpha(R[(3,2)] + \gamma \max_{a'} Q[a',(2,2)] - Q[N,(3,2)]) \\ &= 0 + 0.5(-1+0-0) = -0.5 \end{aligned}$$

  From $(2,2)$, all actions of $Q[*,(2,2)]$ are equally good according to exploration function, hence we random select one. Say we selected north (N), and go to $(1,2)$.

- We move to $(1,2)$, so current state is $(1,2)$ and previous state is $(2,2)$. Non-terminal state. Update $Q[N,(2,2)]$:

$$\begin{aligned} Q[N,(2,2)] &= Q[N,(2,2)] + \alpha(R[(2,2)] + \gamma \max_{a'} Q[a',(1,2)] - Q[N,(2,2)]) \\ &= 0 + 0.5(-2+0-0) = -1 \end{aligned}$$

  From $(1,2)$, all actions of $Q[*,(1,2)]$ are equally good according to exploration function, hence we random select one. Say we selected west (W), and go to $(1,1)$.

- We move to (1,1), so current state is (1,1) and previous state is (1,2). (1,1) is a terminal state, so we will update the Q-values for all actions to its reward, but only after updating the Q-value of previous state. Update Q[W, (1,2)]:

$$\begin{aligned} Q[W,(1,2)] &= Q[W,(1,2)] + \alpha(R[(1,2)] + \gamma \max_{a'} Q[a',(1,1)] - Q[W,(1,2)]) \\ &= 0 + 0.5(-1+0-0) = -0.5 \end{aligned}$$

  Afterwards, we also update $Q[N,(1,1)] = Q[S,(1,1)] = Q[E,(1,1)] = Q[W,(1,1)] = +16$.

5. For the following scenarios, briefly describe how we can model them as reinforcement learning problems (states, actions, type of rewards etc):

   a) Learning to play chess.

   b) Mary is about to graduate, and she decides to plan her finances for the next 40 years (until retirement). Consider what a reinforcement model might look like for financial planning.

Answer

a) States could be all the possible piece configurations. Actions are the moves of each individual piece. Non-terminal state rewards could be the value of a piece taken/lost, and terminal state (win or lose) can have a large positive or negative value.

b) This is an interesting problem, because there is time involved. Investing at 20 years old will likely have smaller returns (rewards) then investing at 40 years old, making the assumption a person will have more money to invest the older they get. To reflect this, the states can be Mary at different ages, actions are different types of investment, and rewards are tied with a state (age) and action (investment). There is a variant of Q-learning that associates rewards to a state-action pair, and it would be appropriate to use such a one.

6. Consider chess. If we wanted to approximate the utility of the states using a sum of weighted features, discuss the type of features that might be useful for playing chess.

Answer

Remember the utility is a indication of the "usefulness" of a state - in the case of chess, it is evaluating whether a state can lead to a winning or losing strategy. In this context, some features, not exhaustive, could be:

- Number of pieces agent has;
- Number of pieces opponent has;
- Total value of pieces (Queen = 9, Rook = 5 etc) agent has (similarly for opponent);
- Number of squares the agent's King can move (if not many, King might not have much opportunity to break a check;
- Number of moves for pawn closest to been promoted;
- Many more!

COSC1125/1127 Artificial Intelligence

School of Computer Science and IT
RMIT University
Semester 1, 2019

Tutorial Sheet 10
Probability Reasoning

We use $P(\cdot)$ (sometimes also written $\Pr(\cdot)$) to refer to a probability function or probability distribution. When using an upper letter (e.g., X or $Cavity$), we refer to the random variable; when using lowercases we refer to a specific value of the corresponding random variable. So, $P(Cavity)$ is a probability distribution over variable $Cavity$; whereas $P(cavity)$ is a shorthand for $P(Cavity = true)$ and $P(\neg cavity)$ is a shorthand for $P(Cavity = false)$.

1. Prove, formally, that $P(A \mid B \wedge A) = 1$.

Answer

You need to use the definition of conditional probability, $P(X \mid Y) = P(X \wedge Y) \mid P(Y)$, and the definitions of the logical connectives. It is not enough to say that if $B \wedge A$ is "given", then A must be true. From the definition of conditional probability and the fact that $A \wedge A \iff A$ and that conjunction is commutative and associative we have,

$$P(A \mid B \wedge A) = \frac{P(A \wedge (B \wedge A))}{P(B \wedge A)} = \frac{P(B \wedge A)}{P(B \wedge A)} = 1$$

2. Consider the domain of dealing 5-card poker hands from a standard deck of 52 cards, under the assumption that the dealer is fair.

   (a) How many atomic events are there in the joint probability distribution (i.e., how many 5-card hands are there)?

   (b) What is the probability of each atomic event?

   (c) What is the probability of being dealt a royal straight flush (the ace, king, queen, jack and ten of the same suit)?

   (d) What is the probability of being dealt four-of-a-kind (i.e., four cards of different suit but same face value)?

   (e) You are told that the probability drawing two cards from a deck of 52 and them both being the same face value is $\frac{1}{221}$. You take the deck and draw the first card — it is the ace of spades! What is the probability that the second card you draw will also be an ace? (even though this is easy to work out using binomials, use conditional probabilty for this question)

Answer

This is a classic combinatorics question. The point here is to refer to the relevant axioms of probability, principally, the following axioms:

(1) All probabilities are between 0 and 1. $0 \leq P(A) \leq 1$

(2) $P(True) = 1$ and $P(False) = 0$

(3) The probability of a disjuction is given by $P(A \vee B) = P(A) + P(B) - P(A \wedge B)$

The question also helps students to grasp the concept of joint probability distribution over all possible states of the world.

(a) It is important to note here that the hand $\{\clubsuit 2, \diamondsuit 3, \heartsuit 4, \diamondsuit 5, \spadesuit 6\}$, is identical to the hand $\{\spadesuit 6, \diamondsuit 5, \heartsuit 4, \diamondsuit 3, \clubsuit 2\}$. If the order of the cards dealt mattered, then the number of hands would simply be $\frac{52!}{47!}$, or $52 \times 51 \times 50 \times 49 \times 48$, because there are 52 choices for the first card, 51 for the second, 50 for the third, and so on.

Generally though, when dealing a hand of cards, it doesn't matter the order in which they are dealt. So, you need to divide this number by the number of possible permutations for a hand of 5 cards, which is $5 \times 4 \times 3 \times 2 \times 1 = 5!$. This means that the total number of 5 card hands that are possible in a 52 card deck is $\frac{52!}{47! \times 5!} = 2,598,960$.

In combinatorics this is written as the binomial coefficient $\binom{52}{5}$, and means: "out of 52 cards, choose 5". In general, $\binom{n}{k} = \frac{n!}{k!(n-k)!}$.

(b) By the fair-dealing assumption, each of these is equally likely. Each hand therefore occurs with probability $\frac{1}{2{,}598{,}960}$

(c) There are four hands that are royal straight flushes (one in each suit). By axiom 3, since the events are mutually exclusive, the probability of a royal straight flush is just the sum of the probabilities of the atomic events, i.e., $\frac{4}{2{,}598{,}960} = \frac{1}{649{,}740}$.

(d) Again, we examine the atomic events that are four-of-a-kind events. There are 13 possible kinds and for each, the fifth card can be one of 48 possible other cards. The total probability is therefore $\frac{13 \times 48}{2{,}598{,}960} = \frac{1}{4{,}165}$.

(e) Even though we are given the suit of the first ace, that information is immaterial as the probability you are given does not specify suit, so we can ignore it. If we let $P(A)$ be the probability of drawing the first ace, its value is simply $\frac{4}{52}$. Let $P(A \wedge B)$ be the probability of drawing two aces in a row, which is $\frac{1}{221}$. We can work out the probability of drawing a second ace, *given* we have already drawn an ace, $P(B \mid A)$, using the definition of conditional probability.

$$P(B \mid A) = \frac{P(A \wedge B)}{P(A)} = \frac{\frac{1}{221}}{\frac{4}{52}} = \frac{52}{4 \times 221} = \frac{3}{51}$$

3. Given the full joint distribution shown in the table below, calculate the following:

	toothache		*¬toothache*	
	catch	*¬catch*	*catch*	*¬catch*
cavity	.108	.012	.072	.008
¬cavity	.016	.064	.144	.576

(a) $P(toothache)$

(b) $P(Cavity)$

(c) $P(Toothache \mid cavity)$

(d) $P(Cavity \mid toothache \vee catch)$.

Answer

Note that $P(Cavity)$ denotes a vector of values for the probabilities of each individual state of Cavity. Also note here we use the uppercase (e.g., $Cavity$) to denote a variable, whereas lowercase (e.g., $cavity$) to denote a constant.

(a) $P(toothache) = 0.108 + 0.012 + 0.016 + 0.064 = 0.2$

(b) $P(Cavity) = \langle 0.108 + 0.012 + 0.072 + 0.008, 0.016 + 0.064 + 0.144 + 0.576\rangle = \langle 0.2, 0.8\rangle$

(c) $P(Toothache \mid cavity) = \langle \frac{0.108+0.012}{0.108+0.012+0.072+0.008}, \frac{0.072+0.008}{0.108+0.012+0.072+0.008}\rangle = \langle 0.6, 0.4\rangle$

(d) $P(Cavity \mid toothache \vee catch)$
$= \langle \frac{0.108+0.012+0.072}{0.108+0.012+0.072+0.016+0.064+0.144}, \frac{0.016+0.064+0.144}{0.108+0.012+0.072+0.016+0.064+0.144}\rangle$
$\approx \langle 0.462, 0.538\rangle$

4. After your yearly checkup, the doctor has bad news and good news. The bad news is that you tested positive for serious disease and that the test is 99% accurate (i.e., the probability of testing positive when you do have the disease is 0.99, as is the probability of testing negative when you don't have the disease). The good news is that this is a rare disease, striking only 1 in 10,000 people of your age. Why is it good news that the disease is rare? What are the chances that you actually have the disease?

Answer

We are given the following information:

$$\begin{aligned}
&P(test \mid disease) = 0.99\\
&P(\neg test \mid \neg disease) = 0.99\\
&P(disease) = 0.0001\\
&test
\end{aligned}$$

where $test$ means that the test is positive. What the patient is concerned about is $P(disease \mid test)$. Roughly speaking, the reason it is a good thing that the disease is rare is that $P(disease \mid test)$ is proportional to $P(disease)$, so a lower prior probability for $Disease$ will mean a lower value for $P(disease \mid test)$. By and large, if 10,000 people take the test, we expect 1 to actually have the disease, and most likely test positive, while the rest do not have the disease, but 1% of them (about 100 people) will test positive anyway, so $P(disease \mid test)$ will be about 1 in 100. More precisely, using the following:

$$\begin{aligned}
P(disease \mid test) &= \frac{P(test \mid disease)P(disease)}{P(test)}\\
&= \frac{P(test \mid disease)P(disease)}{P(test \mid disease)P(disease) + P(test \mid \neg disease)P(\neg disease)}\\
&= \frac{0.99 \times 0.0001}{0.99 \times 0.0001 + 0.01 \times 0.9999}\\
&= 0.009804
\end{aligned}$$

Note that in the above, to calculate $P(test)$, we need to sum over all other hidden variables, but in this case there is only one, that is $Disease$:

$$P(test) = P(test \wedge disease) + P(test \wedge \neg disease)$$

by the product rule, we have:

$$\begin{aligned}
P(test \wedge disease) &= P(test \mid disease)P(disease)\\
P(test \wedge \neg disease) &= P(test \mid \neg disease)P(\neg disease)
\end{aligned}$$

The moral is that when the disease is much rarer than the test accuracy, a positive result does not mean the disease is likely. A false positive reading remains much more likely.

5. Prove, formally, that $P(A \wedge B \wedge C) = P(A \mid B \wedge C) \times P(B \mid C) \times P(C)$.

Answer

Rearranging the definition of conditional probability,

$$P(X \mid Y) = \frac{P(X \wedge Y)}{P(Y)}$$

gives the product rule:

$$P(X \wedge Y) = P(X \mid Y) \times P(Y)$$

Using the product rule and substituting X/A and $Y/[B \wedge C]$ into $P(A \wedge B \wedge C)$, we have,

$$P(A \wedge B \wedge C) = P(A \mid B \wedge C) \times P(B \wedge C)$$

Finally, applying the product rule again to $P(B \wedge C)$ and substituting X/B and Y/C gives:

$$P(A \wedge B \wedge C) = P(A \mid B \wedge C) \times P(B \mid C) \times P(C)$$

6. Prove Bayes' Theorem: $P(A \mid B) = \dfrac{P(B \mid A) \times P(A)}{P(B)}$.

Answer

The probability of two events A and B happening, $P(A \wedge B)$, is the probablity of A, $P(A)$, times the probability of B given that B has occurred, $P(B \mid A)$.

$$P(A \wedge B) = P(A) \times P(B \mid A)$$

On the other hand, the probability of A and B is also equal to the probability of B times the probability of A given B.

$$P(A \wedge B) = P(B) \times P(A \mid B)$$

Equating the two yields:

$$P(B) \times P(A \mid B) = P(A) \times P(B \mid A)$$

and thus,

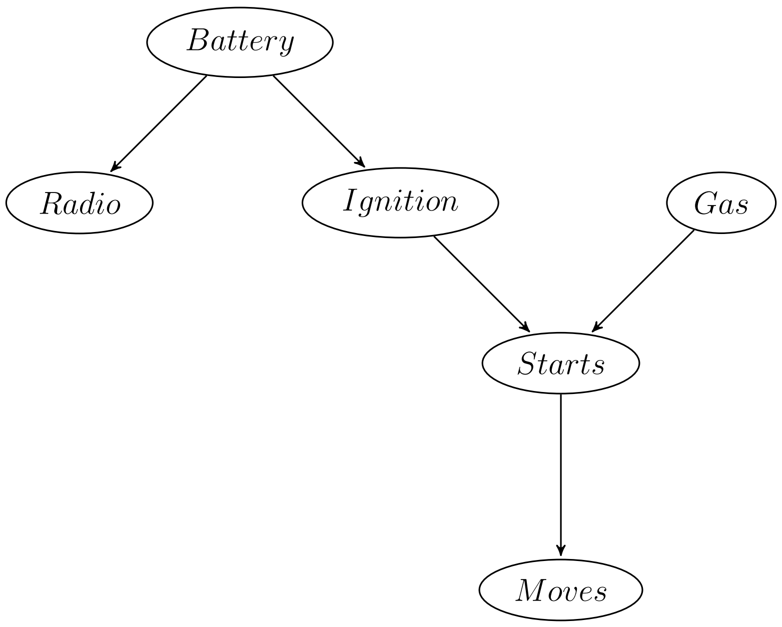
$$P(A \mid B) = P(A) \times \frac{P(B \mid A)}{P(B)}$$

COSC1125/1127 Artificial Intelligence

School of Computer Science and IT
RMIT University
Semester 1, 2019

Tutorial Sheet 11
Bayesian Networks

1. Consider the network for car diagnosis shown in the figure below:

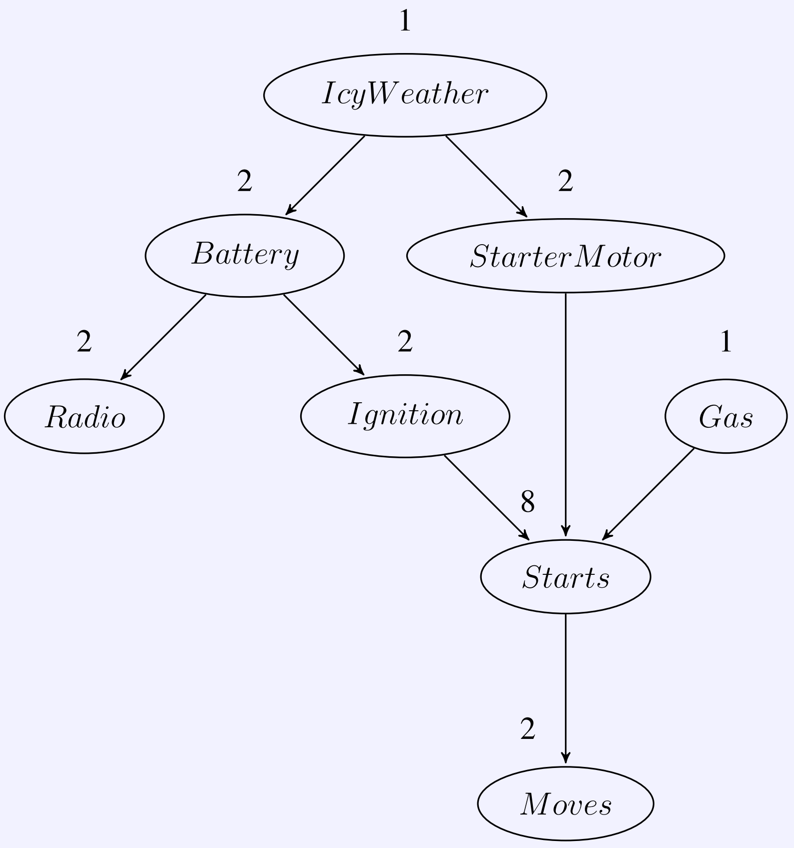

$$\begin{tikzcd}
 & Battery \arrow[dl] \arrow[dr] & & \\
Radio & & Ignition \arrow[dr] & & Gas \arrow[dl] \\
 & & & Starts \arrow[d] & \\
 & & & Moves &
\end{tikzcd}$$

   (a) Extend the network with the Boolean variables *IcyWeather* and *StarterMotor*, by reasoning how components might affect each other. Being a design/modelling process, there may be many correct answers, just state your assumptions.

Answer

Adding variables to an existing net can be done in two ways. Formally speaking, one should insert the variable ordering and rerun the network construction process from the point where the first new variable appears. Informally speaking, one never really builds a network by a strict ordering. Instead, one asks what variables are direct causes or influences on what other ones, and builds local parent/child graphs that way. It is usually very easy to identify where in such a structure the new variable goes, but one must be very careful to check for possible induced dependencies downstream.

$IcyWeather$ is not caused by any of the car-related variables, so needs no parents. It directly affects the battery and starter moter. $StarterMotor$ is an additional precondition for $Starts$. The new network is shown below:



(b) Give reasonable conditional probability tables for all the nodes.

Answer

i. A reasonable prior for $IcyWeather$ might be 0.05 (perhaps, depending on the location and season).

ii. $P(Battery|IcyWeather) = 0.95$,
$P(Battery|\neg IcyWeather) = 0.997$

iii. $P(StarterMotor|IcyWeather) = 0.98$,
$P(StarterMotor|\neg IcyWeather) = 0.999$

iv. $P(Radio|Battery) = 0.9999$,
$P(Radio|\neg Battery) = 0.05$

v. $P(Ignition|Battery) = 0.998$,
$P(Ignition|\neg Battery) = 0.01$

vi. $P(Gas) = 0.995$

vii. $P(Starts|Ignition, StarterMotor, Gas) = 0.9999$

viii. $P(Moves|Starts) = 0.998$

(c) How many independent values are contained in the joint probability distribution for eight Boolean nodes, assuming that no conditional independence relations are known to hold among them?

Answer

With 8 Boolean variables, the joint has $2^8 - 1 = 255$ independent entries. In general there are 2^n entries in a joint probability distribution; however, only $2^n - 1$ *independent* entries. If the probabilities of all of the independent entries are summed up, the probability of the final entry must be $1 - \sum P(independent)$, making its value *dependent* on the rest.

(d) How many independent probability values do your network tables contain?

Answer

Given the topology shown in the figure from question 1a, the total number of independent CPT entries is $1 + 2 + 2 + 2 + 2 + 1 + 8 + 2 = 20$

(e) Reconstruct, showing your workings, five instances of the full joint probability distribution (e.g., reconstruct $P(\neg iW, b, r, i, g, s, sM, \neg m)$).

Answer

Here is one example. The probabilities are taken from question 1(b).

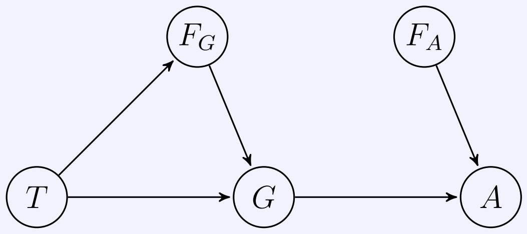
$$\begin{aligned}
&P(\neg iW, b, r, i, g, s, sM, \neg m)\\
&= P(\neg iW) \cdot P(b \mid \neg iW) \cdot P(r|b) \cdot P(i \mid b) \cdot P(g) \cdot P(s \mid i, sM, g) \cdot P(\neg m \mid s)\\
&= (1 - P(iW)) \cdot P(b \mid \neg iW) \cdot P(r|b) \cdot P(i \mid b) \cdot P(g) \cdot P(s \mid i, sM, g) \cdot (1 - P(m \mid s))\\
&= 0.95 \cdot 0.997 \cdot 0.9999 \cdot 0.998 \cdot 0.995 \cdot 0.9999 \cdot 0.002\\
&= 0.00188
\end{aligned}$$

2. In your local nuclear power station, there is an alarm that senses when a temperature gauge exceeds a given threshold. The gauge measures the temperature of the core. Consider the Boolean variable A (alarm sounds), F_A (alarm is faulty), and F_G (gauge is faulty) and the multivalued nodes G (gauge reading) and T (actual core temperature).

   (a) Draw a Bayesian network for this domain, given that the gauge is more likely to fail when the core temperature gets too high.

Answer

A suitable network is shown in the figure below. The key aspects are: the failure nodes are parents of the sensor nodes, and the temperature node is a parent of both the gauge and the gauge failure node. It is exactly this kind of correlation that makes it difficult for humans to understand what is happening in complex systems with unreliable sensors.



   (b) Is your network a polytree?

Answer

No, it is not a polytree, since if we replace the directed edges with undirected edges, we obtain an undirected graph which is cyclic. Therefore, no matter which way the network is drawn, it should not be a polytree because of the fact that the temperature influences the gauge in two ways.

   (c) Suppose there are just two possible actual and measured temperatures, normal and high; the probability that the gauge gives the correct temperature is x when it is working, but y when it is faulty. Give the conditional probability table associated with G.

Answer

The CPT for G is shown below. The wording of the question is a little tricky because x and y are defined in terms of "incorrect" rather than "correct".

	$T = Normal$		$T = High$	
	F_G	$\neg F_G$	F_G	$\neg F_G$
$G = High$	$1 - y$	$1 - x$	y	x
$G = Normal$	y	x	$1 - y$	$1 - x$

(d) Suppose the alarm works correctly unless it is faulty, in which case it never sounds. Give the conditional probability table associated with A.

Answer

The CPT for A is as follows:

	$G = Normal$		$G = High$	
	F_A	$\neg F_A$	F_A	$\neg F_A$
A	0	0	0	1
$\neg A$	1	1	1	0

COSC1125/1127 Artificial Intelligence
School of Computer Science and IT
RMIT University
Semester 1, 2019

Tutorial Sheet 12
Intelligent Agents

1. Define in your own words the following terms (Problem 2.1 from Russell and Norvig's book):

   (a) Agent

   Answer

   An entity that perceives and acts; or, one that can be viewed as perceiving and acting. Essentially any object qualifies; the key point is the way the object implements an agent function. (Note: some authors restrict the term to programs that operate on behalf of a human, or to programs that can cause some or all of their code to run on other machines on a network, as in mobile agents.)

   (b) Agent function

   Answer

   A function that specifies the agents action in response to every possible percept sequence.

   (c) Agent program

   Answer

   A program which, combined with a machine architecture, implements an agent function. In our simple designs, the program takes a new percept on each invocation and returns an action.

   (d) Rationality

   Answer

   A property of agents that choose actions that maximize their expected utility, given the percepts to date.

   (e) Autonomy

   Answer

   A property of agents whose behavior is determined by their own experience rather than solely by their initial programming.

   (f) Reflex agent

   Answer

   An agent whose action depends only on the current percept.

(g) Model-based agent

Answer

An agent whose action is derived directly from an internal model of the current world state that is updated over time.

(h) Goal-based agent

Answer

An agent that selects actions that it believes will achieve explicitly represented goals.

(i) Utility-based agent

Answer

An agent that selects actions that it believes will maximize the expected utility of the outcopme state.

(j) Learning agent

Answer

An agent whose behavior improves over time based on its experience.

2. Explain the concept of *performance measure* in intelligent agents.

Answer

Performance measure is the criterion for ’success’ and is objective: it is used by an outside observer to evaluate how successful an agent is.

3. Explain the concept of a *utility function* in intelligent agents.

Answer

A utility function is used by an agent to evaluate how desirable states or histories are.

4. What is the difference between the performance measure and utility function?

Answer

In our framework, the utility function may not be the same as the performance measure; furthermore, an agent may have no explicit utility function at all, whereas there is always a performance measure.

5. What is practical reasoning and how is it different from theoretical reasoning? Explain in no more than 4 sentences.

Answer

Practical reasoning is the reasoning towards action: what to do next. Theoretical reasoning is that one to understand how the world is, what is believed true.