CSIT5900 Artificial Intelligence **Name:** _____
Fall 2023 Final
19:30 - 22:30 on 7/12/2023 **Stu ID:** _____
Time Limit: 3 hrs

Instructions:

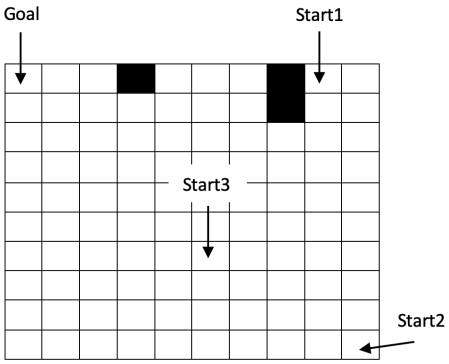
1. This exam contains 14 pages (including this cover page) and 10 questions.
2. This is a closed book exam. There are some copies of lecture notes in the end for your reference.
3. Observe the honor code.

Grade Table (for teacher use only)

Question	Points	Score
Reactive Agents and State Machines	10	
Perceptron Learning and GSCA Rule Learning	15	
Heuristic Search	12	
Alpha-Beta Pruning	5	
Games and RL	10	
MDP	10	
Uncertainty	10	
Propositional Logic	10	
FOL Representation	10	
Game Theory	8	
Total:	100	

Question 1: Reactive Agents and State Machines . . . . . 10 points

Consider the robot with the same specification as our boundary-following robot discussed in class: eight sensors $s_1, ..., s_8$ and four actions ($North$, $South$, $East$, and $West$). Now suppose the environment is a 10x10 grid with an obstacle inside as shown below:



Suppose we want the robot to go to the **top left** corner labeled by "Goal". For each of the three starting positions labled in the figure, determine if this goal can be achieved by a reactive agent:

- If your answer is yes, give a production system for it. Do not call another production system. Write your own rules. When your production system is run with the given starting position, it will move the robot to the goal position, and as soon as it is in the goal position, it will stop (*nil* action).
- If your answer is no, give your reason for it.

Solution:

- Start1 (4 pts): No. The starting position and the goal position have the same sensor readings.
- Start2 (3 pts): yes
- Start3 (3 pts): yes

Question 2: Perceptron Learning and GSCA Rule Learning . . . . . 15 points

Consider the following data set:

ID	x_1	x_2	x_3	x_4	OK
1	1	0	1	0	Yes
2	0	0	1	0	Yes
3	1	0	0	0	No
4	1	1	0	1	No
5	0	1	1	1	No

where x_1, x_2, x_3 and x_4 are some features that should not concern us here.

2.1 (6 pts) Use these five instances to train a single perceptron using the error-correction procedure. Use the learning rate = 1, and the initial weights all equal to 0. Recall that the threshold is considered to be a new input that always have value "1". Please give your answer by filling in the following table, where weight vector (w_1, w_2, w_3, w_4, t) means that w_i is the weight of input x_i, and t is the weight for the new input corresponding to the threshold. Stop when the weight vector converges. If it doesn't converge, explain why not.

ID	Weight vector	Weighted Sum	Actual	Desired
Initial	0, 0, 0, 0, 0			
1				
2				
3				
4				
5				
...	...	...	...	...

2.2 (3 pts) What is the Boolean expression corresponding to your perceptron?

2.3 (6 pts) From the same training set, apply the GSCA algorithm to try to learn a set of rules. Give the set of rules if it succeeds. If it fails to learn a set of rules, explain why it failed.

Solution: 1.

ID	Extended input	Weight vector	Sum	Actual	Desired
Initial		0, 0, 0, 0, 0			
1	1, 0, 1, 0, 1	0, 0, 0, 0, 0	0	1	1
2	0, 0, 1, 0, 1	0, 0, 0, 0, 0	0	1	1
3	1, 0, 0, 0, 1	-1, 0, 0, 0, -1	0	1	0
4	1, 1, 0, 1, 1	-1, 0, 0, 0, -1	-2	0	0
5	0, 1, 1, 1, 1	-1, 0, 0, 0, -1	-1	0	0
1		0, 0, 1, 0, 0	-2	0	1
2		0, 0, 1, 0, 0	1	1	1
3		-1, 0, 1, 0, -1	0	1	0
4		-1, 0, 1, 0, -1	-2	0	0
5		-1, -1, 0, -1, -2	0	1	0
1		0, -1, 1, -1, -1	-3	0	1
2		0, -1, 1, -1, -1	0	1	1
3		0, -1, 1, -1, -1	-1	0	0
4		0, -1, 1, -1, -1	-3	0	0
5		0, -1, 1, -1, -1	-2	0	0
1		0, -1, 1, -1, -1	0	1	1

It converges to give a perceptron with the weight vector $(0, -1, 1, -1)$ and the threshold 1.

2. Boolean expression: $\neg x_2 x_3 \neg x_4$, i.e. $\overline{x_2} x_3 \overline{x_4}$. (Your Boolean expression should correspond to your perceptron, not necessarily the original data.)

3.
We begin with: $true \supset OK$.
Choose the feature that yields the largest value of r_α:
$r_{x_1} = 1/3, r_{x_2} = 0, r_{x_3} = 2/3, r_{x_4} = 0$
. So we choose x_3, this will generate : $x_3 \supset OK$
This rule still covers the negative instance ID5, so we still need to narrow it. This time we have $r_{x_1} = 0.5, r_{x_2} = 0, r_{x_4} = 0$. We can add $\{x_1\}$: $x_1 \wedge x_3 \subset OK$. This rule covers only positive instances, so we have learned one rule. To learn the next one, we eliminate data with ID1 and continue like above.
We begin with: $true \supset OK$.
Choose the feature that yields the largest value of r_α:
$r_{x_1} = 0, r_{x_2} = 0, r_{x_3} = 0.5, r_{x_4} = 0$.
So we choose x_3, this will generate : $x_3 \supset OK$
This rule still covers the negative instance ID5, so we still need to narrow it. This time we have $r_{x_1} = 0$, and x_2=0 and x_4=0. So we add either of x_1, x_2, x_4, but there are no positive examples covered by the new generated rule. So we track back to $x_3 \supset OK$ and replace it by $x_1 \subset OK$ or $x_2 \subset OK$ or $x_4 \subset OK$. All of them cover no positive examples. So it backtrack to the beginning and abort (or loop for ever). So GSCA only learns one rule, for the reason that this training set cannot be learned without making use of negative literals like $\neg x_2$ or $\neg x_4$ in the rules.

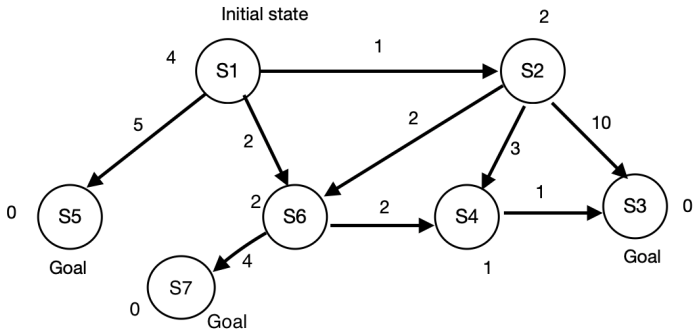
Question 3: Heuristic Search . . . . . 12 points

Consider the following search problem, where the numbers on the edges are costs of the corresponding actions, and the numbers next to the states are their heuristic values.

3.1 (6 pts) Apply A* search by tree on this problem and give the solution returned by it. You can answer this question by drawing a search tree with the sequence of nodes selected for expansion clearly indicated. You can use any tie-breaking rule.

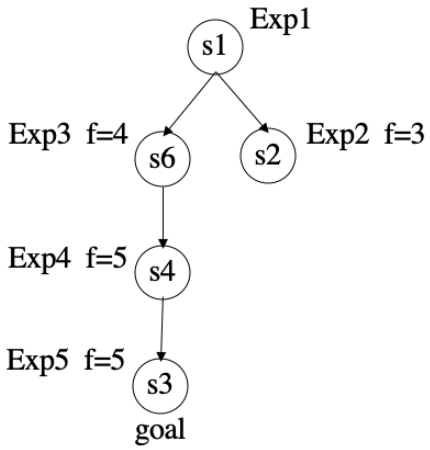
3.2 (3 pts) Can you come up with a new *admissible* heuristic function so that A* algorithm will return the solution $S1 \rightarrow S5$ without using any tie-breaking rule? If you can, provide such a heuristic function. If not, explain why not.

3.3 (3 pts) Can you come up with a heuristic function, admissible or not, so that A* will return the solution $S1 \rightarrow S6 \rightarrow S7$?



Solution:

1. The answer is not unique. For example, use "Prefer newly generated nodes" as tie-breaking rule.



2. No. There are multiple optimal solutions. Any of them can be returned depending on the tie-breaking rule.

3. Yes. Let it be 0 on $S6$ and $S7$ and infinitely large on other states.

Question 4: Alpha-Beta Pruning . . . . . 5 points

Consider the following game tree: Perform a left-to-right alpha-beta pruning. Which nodes are pruned? Notice that Left-to-right means that whenever a node is expanded, it's children are considered in the order from left to right.

Solution: M and E.

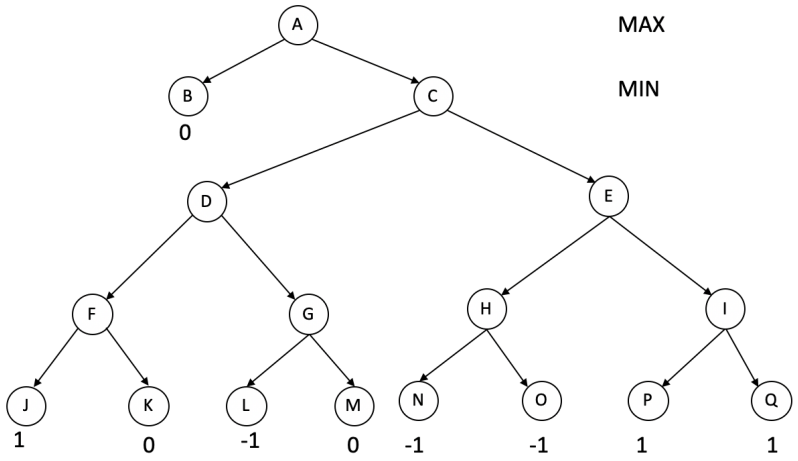


Figure 1: A minimax search tree

Question 5: Games and RL . . . . . 10 points

Consider the Tic-Tac-Toe game. Describe how you can train player X to play the game using reinforcement learning:

- Describe the states;
- Describe the actions that X can play in the states;
- Describe the starting state;
- Describe the end states;
- Describe what the transition probability function means, and whether it needs to be learned;
- Describe what the reward function means, and whether it needs to be learned

There are articles on the internet about using reinforcement learning to play the game. You can read as many of them as you want. In the end, if your answer uses some ideas from them, you have to cite them.

Solution: One possible way:

- States: all positions with equal number of X's and O's on the board.
- Actions: X marks an un-occupied cell.
- Starting state: the empty board position.
- End states: end of the game.

- T-function: $T(s, a, s')$ means that after X makes the move a in s, the probability that O's response will lead to the new state s'. This cannot be known ahead of time so needs to be learned. What will be learned will depend on the opponent you used to train your strategy for X.
- Rewards: One way to define the reward would be to make it 0 if the transition does not lead to an end state, and X's utility if it does. So $R(s, a, s') = 0$ if s' is not an end state. In this case, this does not need to be learned.

Question 6: MDP ..... 10 points

Imagine you are playing the following coin game. You can either quit, which will end the game and reward you \$10, or stay, which will reward you \$5 immediately and then a fair coin will be tossed. If it ends up with the head side up, then the game continues. If it ends up with the tail side up, then the coin will be tossed again. This time if it ends up with the head side up, then you pay \$9 (meaning your net loss is -\$4 in this case), and the game continues; but if it ends up with the tail side up, then the game ends. Answer the following questions by assuming the discount factor of 1.

6.1 (6 pts) Formulate this problem as a MDP.

6.2 (4 pts) Compute the optimal plan of your MDP.

Solution: MDP:

- States: $\{In, Out\}$.
- Starting state: In.
- End states: Out.
- Actions: $\{stay, quit\}$.
- $T(In, quit, Out) = 1$. $T(In, stay, Out) = .25$. $T(In, stay, In) = .75$.
- $R(In, quit, Out) = 10$. $R(In, stay, Out) = 5$, $R(In, stay, In) = [5 * 0.5 + (5 - 9) * 0.25]/.75 = 2$. *(Give partial mark for* $R(In, stay, In) = [5 * 0.5 + (5 - 9) * 0.25]$ *which will make quit better)*
- $\gamma = 1$.

$\pi_1 = stay, \pi_2 = quit$
$V(out) = 0$

$$\begin{aligned} V_{\pi_1}(in) &= T(in, stay, in) * [Reward(in, stay, in) + V(in)] \\ &\quad + T(in, stay, out) * [Reward(in, stay, out) + V(out)] \\ &= 0.75 * [2 + V_{\pi_1}(in)] + 0.25 * (5 + 0) \end{aligned}$$

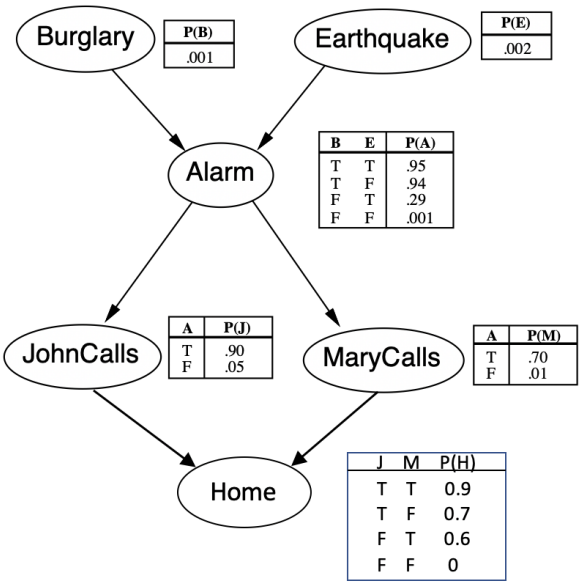
so, $V_{\pi_1}(in) = 11$

$$V_{\pi_2}(in) = T(in, quit, out) * [Reward(in, quit, out) + V(out)]$$
$$= 1 * (10 + 0) = 10$$

so, $V_{\pi_2}(in) = 10$
The optimal policy is to stay at every turn util being forced to end game *(Give full mark for policy if they compute the values correctly using the wrong rewards and/or transition probabilities)*

Question 7: Uncertainty ..... 10 points

Consider the following Bayesian network which adds one more node, *Home* (whether to go back home), to Pearl's example:



7.1 (5 pts) Are J (JohnCalls) and M (MaryCalls) independent given A (Alarm)? Explain your answer using D-separation.

7.2 (5 pts) Compute the probability of H (Home) being true given that A is true: $P(H|A)$. There is no need to perform numerical calculations. As long as your formula is right, you will get the full mark.

Solution: Yes, J and M are independent given A.
Two path from J to M, (1)J-A-M, (2)J-H-M
for (1) A is in E and is not the case that both path arrows lead in to A

for (2) H is not in E, both path arrows lead in to H, and neither H nor any descendants of H are in E

$$
\begin{aligned}
P(H=1|A=1) &= \Sigma_{J,M} P(H=1,J,M|A=1) \\
&= \Sigma_{J,M} P(H=1|J,M,A=1)P(J|A=1)P(M|A=1) \\
&= P(H=1|J=1,M=1,A=1)P(J=1|A=1)P(M=1|A=1) \\
&+ P(H=1|J=1,M=0,A=1)P(J=1|A=1)P(M=0|A=1) \\
&+ P(H=1|J=0,M=1,A=1)P(J=0|A=1)P(M=1|A=1) \\
&+ P(H=1|J=0,M=0,A=1)P(J=0|A=1)P(M=0|A=1)
\end{aligned}
$$

Question 8: Propositional Logic . . . . . 10 points

A_1, A_2 and A_3 are three friends. We know the following facts about them:

- At least one of them is in the car.
- A_3 cannot drive.
- A_1 is in the car only if A_2 is in the car.

Use propositional logic to show that A_2 is in the car:

8.1 (5 pts) Encode the given facts as well as any necessary common sense knowledge as a KB using the following symbols:

- P_i: A_i is in the car, $i = 1, 2, 3$.
- D_i: A_i is the driver, $i = 1, 2, 3$.

8.2 (2 pts) Convert your KB to a set of clauses.

8.3 (3 pts) Use resolution (and proof by refutation) to show that your KB entails P_2.

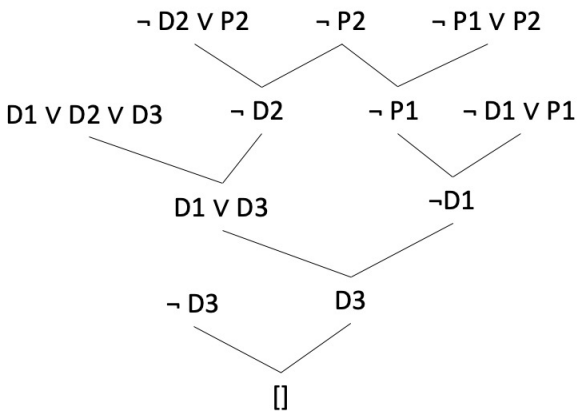
Solution:

$$
\begin{aligned}
&P_1 \vee P_2 \vee P_3, (\text{not necessary})\\
&D_1 \vee D_2 \vee D_3,\\
&D_1 \supset P_1,\\
&D_2 \supset P_2,\\
&D_3 \supset P_3,\\
&\neg D_3,\\
&P_1 \supset P_2
\end{aligned}
$$

Clauses:

$$
\begin{aligned}
&P_1 \vee P_2 \vee P_3,\\
&D_1 \vee D_2 \vee D_3,\\
&\neg D_1 \vee P_1,\\
&\neg D_2 \vee P_2,\\
&\neg D_3 \vee P_3,\\
&\neg D_3,\\
&\neg P_1 \vee P_2
\end{aligned}
$$

Add negation of query $\neg P_2$.



Question 9: FOL Representation . . . . . 10 points

Given a graph, let predicate $e(x, y)$ stand for that nodes x and y have an edge connecting them, and function $c(x)$ for the color of node x. Let R, W and Y be three constants denoting colors red, white and yellow, respectively. Represent the following statements in first-order logic:

1. If two nodes are connected (by an edge), then they cannot have the same color.
2. The graph is fully connected, i.e. there is an edge between every pair of nodes.
3. A red node is conncted only to yellow nodes.
4. There is exactly one red node in the graph.
5. If a red node is connected with a yellow node, then it is also connected with a white node.

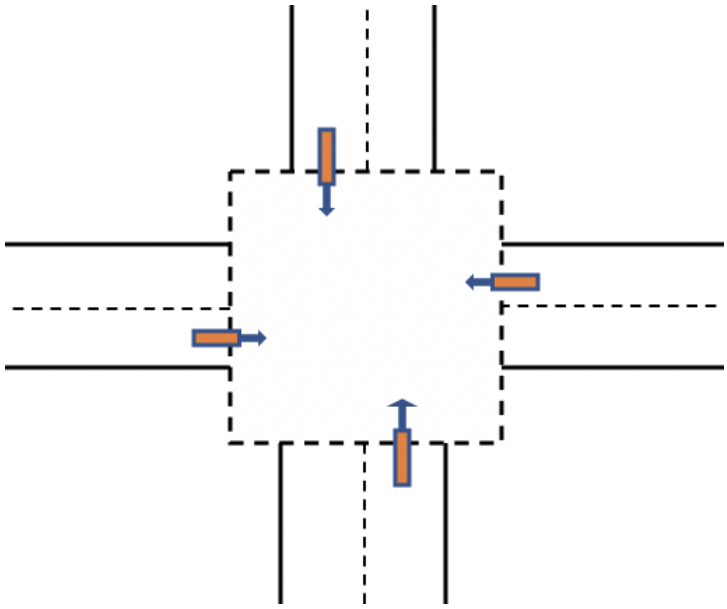
Note that we do not have an explicit node class. For example, when we write $\forall x \exists y.e(x, y)$, the variables x and y are understood to be nodes in the given graph. Note also that we assume the graph is undirected, so you can use either $e(x, y)$ or $e(y, x)$ to say that x and y are connected.

Solution:

1. $\forall x, y.e(x, y) \supset c(x) \neq c(y)$,
2. $\forall x, y.x \neq y \supset e(x, y)$,
   $\forall x, y.e(x, y)$ is acceptable ,
3. $\forall x, y.e(x, y) \wedge c(x) = R \wedge x \neq y \supset c(y) = Y$, okay to not include $x \neq y$,
4. $\exists x.[c(x) = R \wedge \forall y(c(y) = R \supset y = x)]$,
5. $\forall x, y.e(x, y) \wedge c(x) = R \wedge c(y) = Y \supset \exists z(e(x, z) \wedge c(z) = W)$

Question 10: Game Theory ..... 8 points

Imagine 4 cars come to an intersection as illustrated in the following figure.



Each car can either stop or go. If two neighboring cars go at the same time, they will collide. Formulate this problem as a game in normal form and compute all its Nash equilibria.

Solution: There are multiple ways to formulate it. The easiest one is to have two agents: agent 1 consists of the two cars on the south-north road, and agent 2 consists of the two cars on the east-west road. The NEs are then $(go, stop)$ and $(stop, go)$:

	go	stop
go	-1, -1	1,0
stop	0,1	0, 0

Alternatively, you can let each car be an agent: $N = \{1, 2, 3, 4\}$. In this case, the NEs will be $(go, stop, go, stop)$ and $(stop, go, stop, go)$, with utility functions like

1	2	3	4	u_1	u_2	u_3	u_4
go	go	go	go	-1	-1	-1	-1
go	go	go	stop	-1	-1	-1	0
go	go	stop	go	-1	-1	0	-1
go	go	stop	stop	-1	-1	0	0
go	stop	go	go	-1	0	-1	-1
go	stop	go	stop	1	0	1	0
go	stop	stop	go	-1	0	0	-1
go	stop	stop	stop	1	0	0	0
...	...	...	...	...	...	...	...