



School of Computer Science

Time Constrained Assessment

Module Code	CMP2020M
Module Title	Artificial Intelligence
Module Coordinator	Dr. Bashir Al-Diri
Duration of Assessment	3 Hours
Date	Tuesday 5 th May 2020
Release Time	15:00 British Summer Time
Submission Time	18:00 British Summer Time
Total Wordlimit	Wordlimit (+/- 10%)

General Instructions to Candidates.

1. You **must** submit your answers as a MS Word Document to Turnitin on Blackboard **before** the submission time: failure to do so will be classified as misconduct in examinations. We strongly recommend you submit 15 minutes prior to the deadline.
2. You **must** also send a copy of your work to the socssubmissions@lincoln.ac.uk at the same time. You must place the Module Code and your Student Id in the Subject Field of the Mail.
3. Hand-written notes or diagrams, **must** be photographed and inserted into the Word Document as an image.
4. This assessment is an open resource format: you may use online resources, lecture and seminar notes, text books and journals.
5. All work will be subject to plagiarism and academic integrity checks. In submitting your assessment you are claiming that it is your own original work; if standard checks suggest otherwise, Academic Misconduct Regulations will be applied.
6. The duration of the Time Constrained Assessment will vary for those students with Learning Support Plans. Extensions do not apply, but Extenuating Circumstances can be applied for in the normal way.

Module Specific Instructions to Candidates

QUESTIONS TO ANSWER:	Answer all questions
MARKING SCHEME:	The marks are indicated for each individual question
MATERIALS PROVIDED:	Formulas and truth tables (appendix).
MATERIALS PERMITTED:	non programmable calculator/ graphical calculator/ scientific calculator.

Question 1: Logic Programming [10 marks in total]

Consider the following statements:

- We will go swimming only if it is sunny.
- It is not sunny this afternoon and it is colder than yesterday.
- If we do not go swimming, then we will take a canoe trip.
- If we take a canoe trip, then we will be home by sunset.
- Therefore (conclusion), we will be home by the sunset.

Prove the above argument is valid using inference rules by answering the following questions.

1. Translate the above statements into propositional logic [2 marks],

Write your answer here:

ANSWER:

S: "it is sunny this afternoon"

C: "it is colder than yesterday"

W: "we will go swimming"

T: "we will take a canoe trip."

H: "we will be home by the sunset."

2. Write down the premises and the goal into propositional logic [2 marks],

Write your answer here:

ANSWER:

Premises are:

- It is not sunny this afternoon and it is colder than yesterday. $\neg S \wedge C$
- We will go swimming only if it is sunny. $W \rightarrow S$
- If we do not go swimming, then we will take a canoe trip. $\neg W \rightarrow T$
- If we take a canoe trip, then we will be home by sunset. $T \rightarrow H$

Goal is:

- We will be home by the sunset. H

3. Write a formal proof, a sequence of steps that state premises or apply inference rules to previous steps. Justify each step by citing the rule of inference needed [6 marks].

Write your answer here:

ANSWER:

• Step	• Reason
• 1. $\neg S \wedge C$	• Premise
• 2. $\neg S$	• Simplification
• 3. $W \rightarrow S$	• Premise
• 4. $\neg W$	• modus tollens of 2 and 3
• 5. $\neg W \rightarrow T$	• Premise
• 6. T	• modus ponens of 4 and 5
• 7. $T \rightarrow H$	• Premise
• 8. H	• modus ponens of 6 and 7

Question 2: Artificial Neural Networks (ANNs) [20 marks in total]

- a) Describe the basic model of an artificial neuron. (5 marks)

Write your answer Here:

ANSWER:

Explanation of basic model of ANN : The basic computational element (model neuron) is often called a node or unit. It receives input. Each input has an associated weight. The unit computes some function f of the weighted sum of its inputs. (5 marks)

- b) A simple perceptron has two input units, a unipolar step function, weights $w_1 = 0.2$ and $w_2 = -0.5$, and a threshold $\theta = -0.2$ (θ can therefore be considered as a weight for an extra input which is always set to -1). What is the actual output of this perceptron for the input pattern $\mathbf{x} = [1, 1]^T$? (6 marks)

Write your answer Here:

ANSWER:

Compute the output (y):

$$\begin{aligned}\text{net} &= (x_1 * w_1) + (x_2 * w_2) + (-1 * \theta) \\ &= (1 * 0.2) + (1 * -0.5) + (-1 * -0.2) \\ &= 0.2 - 0.5 + 0.2 \\ &= -0.1 \\ y &= f(-0.1) = 0\end{aligned}$$

(4 marks)

(2 marks)

- c) The previous perceptron is trained using the learning rule $\Delta \mathbf{w} = \eta (d - y) \mathbf{x}$, where \mathbf{x} is the input vector, η is the learning rate, \mathbf{w} is the weight vector, d is the desired output, and y is the actual output. What are the new values of the weights and threshold after one step of training with the input vector $\mathbf{x} = [0, 1]^T$ and desired output 1, using a learning rate $\eta = 0.2$? (9 marks)

Write your answer Here:

ANSWER:

Compute the new weights (w_1, w_2 and θ):

They might use the previous value for y or they might recalculate it which is still correct.

$$\text{net} = (0 * 0.2) + (1 * -0.5) + (-1 * -0.2) = -0.3.$$

$$y = f(-0.3) = 0;$$

$d = 1, y = 0$ (it might be different if it comes from previous question so it might lead to different answers), $\mathbf{x} = [0 \ 1 \ -1]^T$, $\mathbf{w} = [0.2 \ -0.5 \ -0.2]^T$ and $\eta = 0.2$.

$$\Delta w_1 = 0.2 (1 - 0) = 0$$

(1.5 marks)

$$\Delta w_2 = 0.2 (1 - 0) = 0.2$$

(1.5 marks)

$$\Delta w_3 = \Delta \theta = 0.2 (1 - 0) - 1 = -0.2$$

(1.5 marks)

$$w_1 = 0.2 + 0 = 0.2$$

(1.5 marks)

$$w_2 = -0.5 + 0.2 = -0.3$$

(1.5 marks)

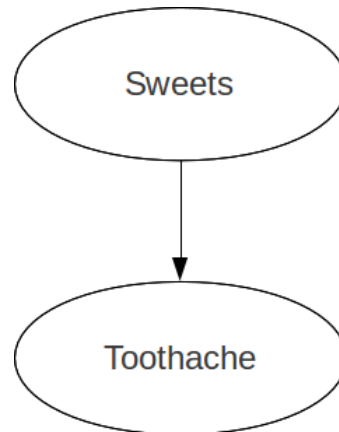
$$\theta = -0.2 - 0.2 = -0.4$$

(1.5 marks)

Question 3: Probabilistic Reasoning [20 marks in total]

Probabilistic Reasoning (medical expert system):

The following diagram shows a Bayesian belief network representing what happens when you eat sweets, i.e, if you eat sweets you may have toothache.



The conditional probabilities are (where S=sweets and T=toothache):

- $P(S)=0.6$ probability of eating sweets
- $P(T|S)=0.8$ indicates the probability of having toothache after eating sweets
- $P(T|\neg S)=0.1$ indicates probability of having toothache without eating sweets.

Please answer the following questions, and make sure you evidence your way of thinking by using the correct formulas:

1. Calculate the prior probability $P(\neg S)$. [3 marks]

Write your answer here:

ANSWER:

$$P(\neg S) = 1 - P(S) = 1.0 - 0.6 = 0.4$$

2. Calculate the conditional probability $p(\neg T|\neg S)$. [5 marks]

Write your answer here:

ANSWER:

$$P(\neg T|\neg S) = 1 - P(T|\neg S) = 1.0 - 0.01 = 0.9$$

3. Calculate $P(T)$, i.e the probability of having toothache [6 marks]

Write your answer here:

ANSWER:

$$P(T) = P(T|S)P(S) + P(T|\neg S)P(\neg S) = 0.8*0.6 + 0.1*0.4 = 0.48 + 0.04 = 0.52$$

2 marks for understanding the general principle (adding the probabilities)

2 marks for picking the right variables

2 marks for the correct calculation

4. Calculate the conditional probability $P(S|T)$, i.e. which is the probability of having eaten sweets if you have toothache [6 marks]

Write your answer here:

ANSWER:

Bayes rule (2 marks)

$p(S|T) = p(T|S)p(S)/p(T)$ (correct numbers entered: 2 marks)

$p(S|T) = 0.8 \cdot 0.6 / 0.52 = 0.92 = 92\%$ (correct result: 2 marks)

Question 4: Planning [20 marks in total]

Look at a simple version of the “blocksworld” problem:

The objective of the “blocksworld” puzzle is to move objects by a robotic manipulator following the following constraints:

- The manipulator can only pick up one object at a time.
- Objects can either be put in one of three empty places on a table or on top of any other object that is free, i.e., has not yet an object on top of it.
- Only objects that have nothing stacked on top of them can ever be picked up.

1. The initial state is depicted below. It shows two objects “b_1” and “b_2” stacked in place 1. Describe this initial state completely in predicate logic (in a syntax similar to PDDL), using the predicate (on ?a ?b) and (free ?a). [5 marks]



Write your answer here:

ANSWER:

(on b_2 b_1)

(on b_1 place_1)

(free b_2)

(free place_2)

(free place_3) [1 mark each]

2. Now look at the goal state below. Again, use predicate logic to describe the *relevant* constraints of this goal state using the same predicates as above:

[3 marks]



Write your answer here:

ANSWER:

(on b_1 b_2)

(on b_2 place_3) [1 mark each, 1 mark for not putting too many constraints]

3. Define an action “(move ?object ?from ?to)” that takes a free object “?object” from either an object or place “?from” and puts it onto either a place or other object “?to” that is also initially free. Describe the precondition and the effect of your action in predicate logic (PDDL-like syntax). [7 marks]

Write your answer here:

ANSWER:

(move ?o ?f ?t):

preconditions: [3 marks]

(and

(free ?o)

(free ?t)

(on ?o ?f)

)

effects: [4 marks]

(and

(not (free ?t))

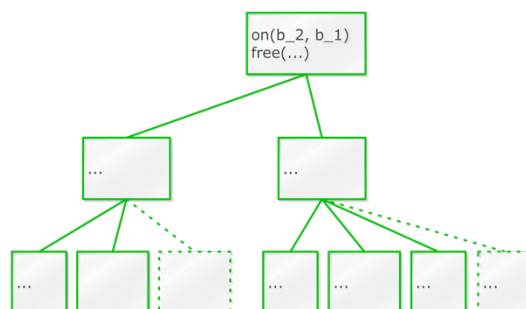
(not (on ?o ?f))

(on ?o ?t)

(free ?f)

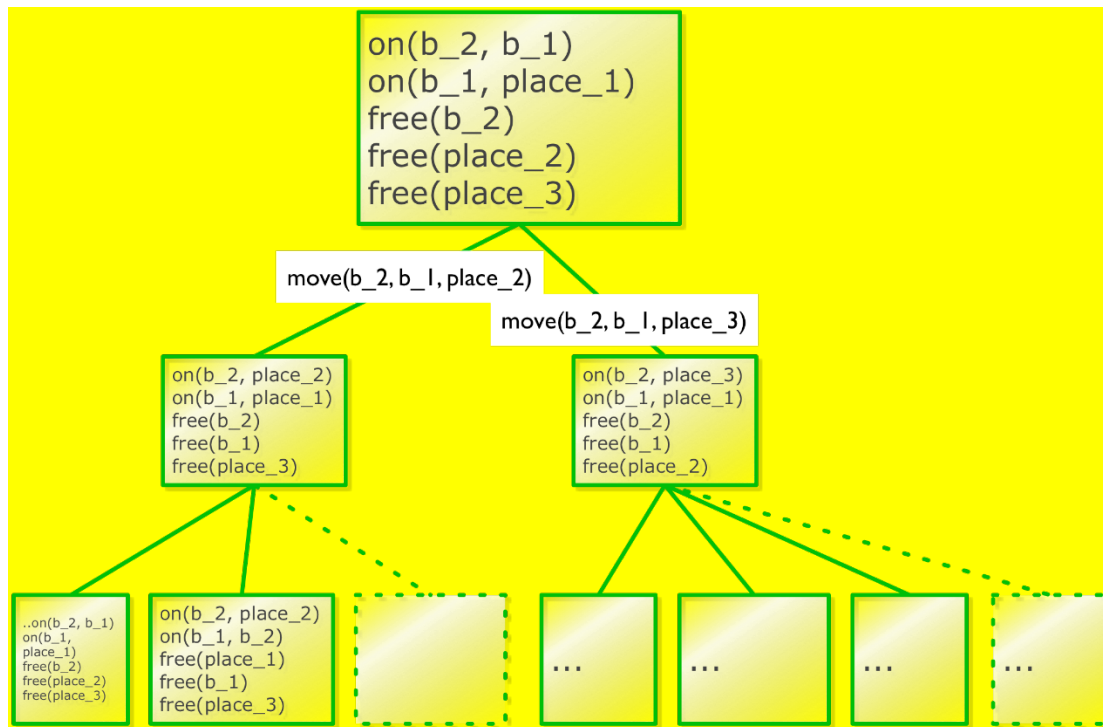
)

4. Draw a search tree (up to depth 3, i.e. two move actions to be executed) using the predicate notation above, i.e., in each node of the search tree indicate the predicates that are true, and on each edge describe the action that is executed. The outcome should look similar to this tree (obviously, you need to complete the nodes and also the number of leaves will be different for you): [5 marks]



Write your answer here:

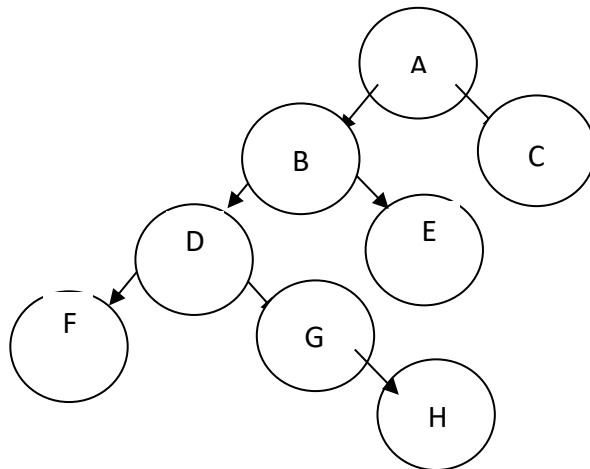
ANSWER:



correct number of branches: 2 marks, correct states: 3 marks

Question 5: Search [20 marks in total]

1. Using the following node tree, explain the difference between depth-first and breadth-first search. What is the node search order in each case? (6 marks)



Write your answer here:

ANSWER:

Depth first starts at the root, and searches to the bottom of each branch, in branch order. [1 or 2 for more detailed explanation, eg implementation/data structures]

Breadth first starts at the root, and visits each neighbour before moving to the next level of the tree. [1 or 2 for more detailed explanation]

Breadth first: A B C D E F G H [1]

Depth first: A B D F G H E C [1]

2. Explain how Dijkstra's algorithm is initialised for a grid-based world, with a known start location. (4 marks)

Write your answer here:

ANSWER:

The total costs for all nodes in the grid are set to infinity [1]

Except for the start node which is zero [1]

All nodes are put on the open list [1]

Node links can be initialised arbitrarily [1]

3. What are the "open" and "closed" lists used for in Dijkstra's algorithm? (2 marks)

Write your answer here:

ANSWER:

The open list contains nodes whose total costs may still to be evaluated and updated [1]

The closed list contains nodes whose costs and links have been finalised [1]

4. In Dijkstra's algorithm, how is a pair-cost used to update the total cost for a node after its neighbour node is closed? (4 marks)

Write your answer here:

ANSWER:

The total cost for the node is re-estimated as the total cost for the closed neighbour + pair cost [1]

The node link is also updated [1]

If the current node total cost is higher than the new estimate [1]

And node isn't already closed [1]

5. Give an example of a specific heuristic distance function for A* that could be used in a grid-based world. (2 marks)

Write your answer here:

ANSWER:

2 marks for either Euclidean, Chebyshev or Manhattan distance (or something else sensible).

6. What does the term "admissible heuristic" mean in relation to the A* algorithm? (2 marks).

Write your answer here:

ANSWER:

It means that the heuristic does not over-estimate the distance cost from any node to the target node [2]

Question 6: Games AI [10 marks in total]

1. State two fundamental requirements of an AI system for a video game (2 marks).

Write your answer here:

ANSWER: Several possible answers including : speed, predictability, low resource requirements, believability. Other sensible suggestions are acceptable. [1 mark for each]

2. Describe an extension to the A* algorithm which uses a 3D grid to plan non-colliding 2D paths for multiple agents moving simultaneously through the same map. (5 marks)

Write your answer here:

ANSWER:

The third dimension represents time [1]

Agents move from layer to layer through the time dimension at each update step [2]

Add a wait function [1]

Process as normal [1]

3. What is path-patching in A*, and when might it be used? (3 marks).

Write your answer here:

ANSWER:

It means recalculating a sub-section of an existing path [2]

It might be used if a path becomes blocked by another object or change to the environment [1]

Appendix: Useful Formulas

Truth Tables

operator \neg
Negation (not)

P	$\neg P$
T	F
F	T

operator \wedge
Conjunction (and)

P	Q	$P \wedge Q$
T	T	T
T	F	F
F	T	F
F	F	F

operator \vee
Disjunction (or)

P	Q	$P \vee Q$
T	T	T
T	F	T
F	T	T
F	F	F

operator \rightarrow
Implication

P	Q	$P \rightarrow Q$
T	T	T
T	F	F
F	T	T
F	F	T

operator \equiv
Equivalence

P	Q	$P \equiv Q$
T	T	T
T	F	F
F	T	F
F	F	T

Inference Rules

- Modus ponens: If P and $P \rightarrow Q$ are true, then infer Q.
- Modus tollens: If $P \rightarrow Q$ and $\neg Q$ are true, then infer $\neg P$.
- And elimination: If $P \wedge Q$ is true, then infer both P and Q are true
- And introduction: If both P and Q are true, then infer $P \wedge Q$
- Disjunctive syllogism (DS): If addition ($P \vee Q$) and $\neg P$ are true, then infer Q

Probability Theory

Mathematically, conditional probability is defined as:

Bayes Theorem (general)

$$P(A | B) = P(A \wedge B) / P(B)$$

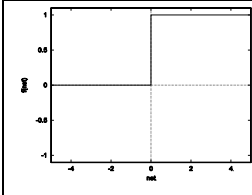
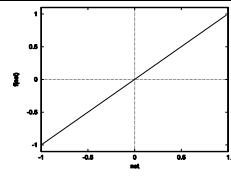
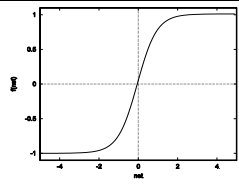
Note that $P(B) \neq 0$.

$$P(H|E) = \frac{P(H)P(E|H)}{P(E)}$$

Artificial Neuron

General Model	$\bar{\mathbf{I}} = \begin{bmatrix} I_1 \\ I_2 \\ \vdots \\ I_n \end{bmatrix} \quad \bar{\mathbf{w}} = \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_n \end{bmatrix} \quad O = f\left(\sum_{j=1}^n w_j I_j - \theta\right) = f(\bar{\mathbf{w}}^T \bar{\mathbf{I}} - \theta) = f(\text{net})$
---------------	---

Activation Functions

		
Step: $f(\text{net}) = \begin{cases} 1, & \text{net} \geq 0 \\ 0, & \text{net} < 0 \end{cases}$	Linear: $f(\text{net}) = \text{net}$	Sigmoid: $f(\text{net}) = \frac{2}{1 + e^{-\lambda \text{net}}} - 1$

Gradient Descent Algorithm

GRADIENT-DESCENT(*training_examples*, η)

Each training example is a pair of the form $\langle \vec{x}, t \rangle$, where \vec{x} is the vector of input values, and t is the target output value. η is the learning rate (e.g., .05).

- Initialize each w_i to some small random value
- Until the termination condition is met, Do
 - Initialize each Δw_i to zero.
 - For each $\langle \vec{x}, t \rangle$ in *training_examples*, Do
 - Input the instance \vec{x} to the unit and compute the output o
 - For each linear unit weight w_i , Do

$$\Delta w_i \leftarrow \Delta w_i + \eta(t - o)x_i$$

- For each linear unit weight w_i , Do

$$w_i \leftarrow w_i + \Delta w_i$$