Final exam of ECE 457 Applied Artificial Intelligence for the Fall term 2007.

Don’t panic.

Be sure to write your name and student ID number on every page of the exam.

The only materials allowed are a pen or pencil, an eraser, and a calculator.

No books or notes of any kind are allowed.

The back side of exam pages will NOT be evaluated or considered, unless you explicitly indicate it is part of your answer.

Rough paper for personal notes, calculations, etc., is available upon request.

Beware of the Wumpus.

Good luck!

Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Q10
/6	/6.5	/7	/8.5	/5.5	/4	/6	/2	/10	/3

Question 1 (6 points)

Consider a miniature version of the 8-puzzle, called the **3-puzzle**, with start and goal states:

Start

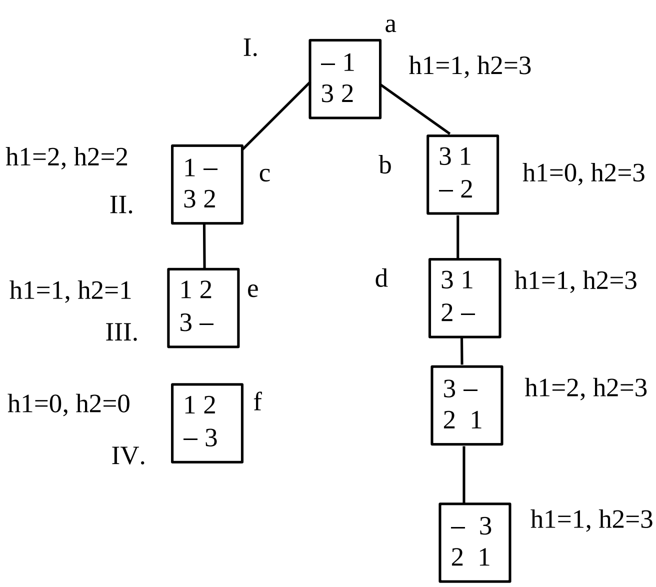
-	1
3	2

Goal

1	2
-	3

Assume that the operations (when allowed) are to move the blank tile *right*, *left*, *up* and *down*. Assume further that we do not generate any duplicate states. You may also assume that all operations have unit cost.

(a) Draw the **state space** for this problem with a depth limit of 4 (to be precise, consider the initial state at depth 0, and stop expanding nodes at depth 4, thus showing all states that are within 4 moves of the start state). ***(1 point)***



(b) Consider the heuristic `h1` = Manhattan distance (the sum of horizontal and vertical distance) of the *blank tile* from its desired position. Label each state in the state space in (a) with `h1` = *number* showing the heuristic cost of the state. ***(1 point)***

(c) Consider the heuristic `h2` = number of misplaced tiles. Label each state in the state space in (a) with `h2` = *number* showing the heuristic cost of the state. Assume that the blank is **not** counted as a tile. ***(1 point)***

(d) Perform **A* search** using `h1`**.** Label the states in the state space in (a) with letters a, b, c, etc. to indicate the order of nodes expanded (also include the goal, if found). ***(1 point)***

(e) Perform **Greedy search** using `h2`**.** Label the states in the state space in (a) with Roman numerals I, II, III, etc. showing the order of nodes expanded (also include the goal, if found). ***(1 point)***

(f) If the blank tile is counted as a tile, would `h2` be admissible? Explain.

No, because when a state is one step away from the goal, h2 will return 2; thus overestimating the true cost. ***(1.5 point)***

Question 2 (6.5 points)
Sudoku is a logic-based number placement puzzle. The objective is to fill a 9x9 grid so that each column, each row, and each of the nine 3x3 boxes contains the digits from 1 to 9. Each digit can only appear once per column, row, and 3x3 box. A sample Sudoku puzzle to use for this question is given in the figure below.

	6							1
			7	9	3			
								5
		9			1	3	2	
		2				7		
	3	5	8			4		
4								
			5	2	6			
1							8	

The squares are labelled $S_{1,1}$ (upper-left corner) to $S_{9,9}$ (lower-right corner). For example, the 5 in the upper-right side of the grid is $S_{3,9}$.

Assume you have access to a function `Y = Box(X)` which takes in a square X on the grid and returns Y, the list of eight squares that belong to the same 3x3 box. For example, `Box(`$S_{2,2}$`) = {`$S_{1,1}$, $S_{1,2}$, $S_{1,3}$, $S_{2,1}$, $S_{2,3}$, $S_{3,1}$, $S_{3,2}$, $S_{3,3}$`}`.

(a) Define the properties of the environment for Sudoku. **(1.5 points)**

Fully Observable, Deterministic, Sequential, Static, Discrete, Single Agent
(0.25 each)

(b) Write Sudoku as a well-defined problem. **(1.5 points)**

Possible Solution 1
Initial state: The board of the figure.
Action: Write a digit 1-9 in a square if it does not appear in the row, column or 3x3 square.
Goal test: All squares are filled.
Cost: 1 per digit written.

Possible Solution 2
Initial state: The board of the figure.
Action: Write a digit 1-9 in a square.
Goal test: All squares are filled and no digit appears twice in a row, column or 3x3 square.
Cost: 1 per digit written.
(0.25 for each item, +0.5 for the condition of checking for duplicates)

(c) Write Sudoku as a constraint satisfaction problem (CSP). **(1.5 point)**

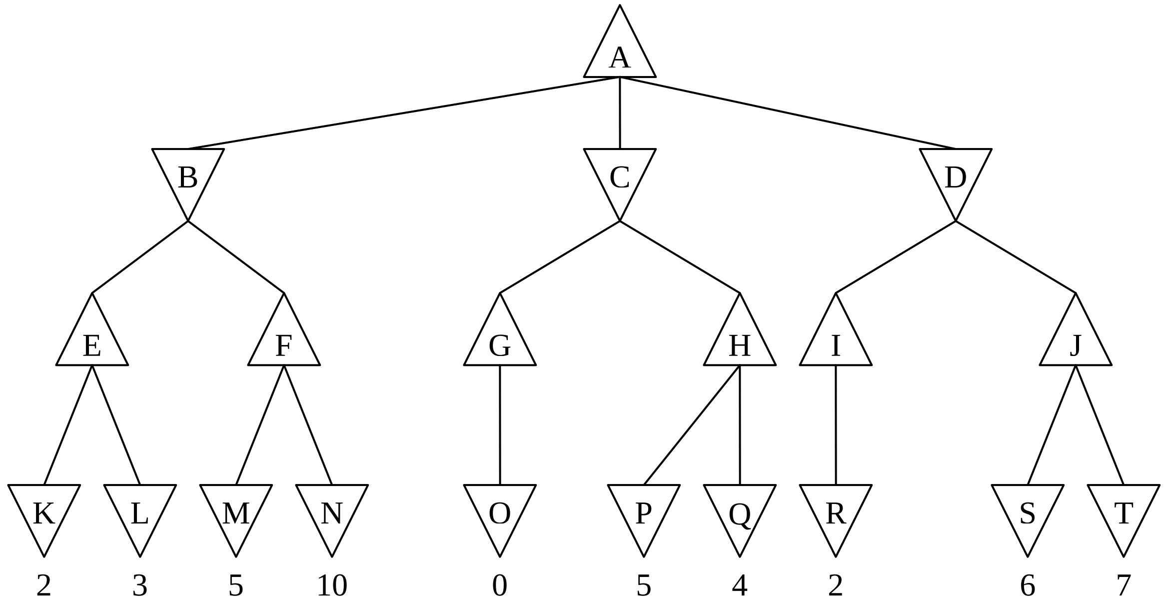
Variables: V = {$S_{1,1}$, …, $S_{9,9}$}
Domain: $V_{i,j} \in D$ = {1, 2, 3, 4, 5, 6, 7, 8, 9}
Constraints: C = { $S_{i,j} \neq S_{i,j+n}$, $S_{i,j} \neq S_{i+n,j}$, $S_{i,j} \notin Box(S_{i,j})$, $S_{1,2} = 6$, $S_{1,9} = 1$, $S_{2,4} = 7$, $S_{2,5} = 7$, $S_{2,6} = 3$, $S_{3,9} = 5$, $S_{4,3} = 9$, $S_{4,6} = 1$, $S_{4,7} = 3$, $S_{4,8} = 2$, $S_{5,3} = 2$, $S_{5,7} = 7$, $S_{6,2} = 3$, $S_{6,3} = 5$, $S_{6,4} = 8$, $S_{6,7} = 4$, , $S_{7,1} = 4$, $S_{8,4} = 5$, $S_{8,5} = 2$, $S_{8,6} = 6$, $S_{9,1} = 1$, $S_{9,8} = 8$} For $n \in [-8, 8]$
(0.25 for the variables and domain, 0.5 for the duplicate constraint, 0.5 for the initial digits)

(d) Propose a good heuristic to search the CSP tree. **(2 points)**

Possible Solution 1
Most constrained variable heuristic: assign a variable that has only one value left.

Possible Solution 2
Most constrained value heuristic: assign to a variable a value that cannot be assigned elsewhere (go elsewhere in the row, column and box).
(2 points if they have a heuristic that basically forces the assignment of unique variable-value pairs, which is the only good heuristic is Sudoku)

Question 3 (7 points)
You have the following game tree. The payoff value of each leaf (nodes K to T) is written under that node.



(a) Apply the Minimax algorithm to obtain the value of each non-leaf node (A to J).

Node:	A	B	C	D	E	F	G	H	I	J
Value:	**3**	**3**	**0**	**2**	**3**	**10**	**0**	**5**	**2**	**7**

(0.1 each, total 1 point)

(b) Apply Alpha-Beta Pruning to the game tree. Find which nodes will be pruned. For each one, identify and explain the value of alpha and beta to show why it is pruned.

N is pruned. *(1 point)* At node F, beta is 3 (from B), and after exploring M alpha is 5. *(1 point)*

H is pruned. *(1 point)* At node C, alpha is 3 (from A), and after exploring G beta is 0. *(1 point)*

J is pruned. *(1 point)* At node D, alpha is 3 (from A), and after exploring I beta is 2. *(1 point)*

Question 4 (10.5 points)
Consider the following text.

Anyone who does not sink and weights the same as a duck is a witch. Everyone who is made of wood weights the same as a duck. Everyone who is a witch is burned. Bob is made of wood, and does not sink.

(a) Represent the text in First-Order Logic. **(2 points)**
NOTE: use the functions and predicates Sink(x), WeightSameAs(x,y), Witch(x), MadeOf(x,y), Burn(x), and the constants Duck, Wood and Bob.

Anyone who does not sink and weights the same as a duck is a witch.
∀x [¬Sink(x) ∧ WeightSameAs(x,Duck) ⇒ Witch(x)] ***(0.5 point)***

Everyone who is made of wood weights the same as a duck.
∀x [MadeOf(x, Wood) ⇒ WeightSameAs(x,Duck)] ***(0.5 point)***

Everyone who is a witch is burned.
∀x [Witch(x) ⇒ Burn(x)] ***(0.5 point)***

Bob is made of wood, and does not sink.
MadeOf(Bob, Wood) ∧ ¬Sink(Bob) ***(0.5 point)***

(b) Convert your FOL sentences to Conjunctive Normal Form. Show all steps of the conversion. **(3.5 points)**

∀x [¬Sink(x) ∧ WeightSameAs(x,Duck) ⇒ Witch(x)]
∀x [¬ {¬Sink(x) ∧ WeightSameAs(x,Duck) } ∨ Witch(x)] ***(0.5 point)***
∀x [Sink(x) ∨ ¬WeightSameAs(x,Duck) ∨ Witch(x)] ***(0.5 point)***
Sink(x) ∨ ¬WeightSameAs(x,Duck) ∨ Witch(x) ***(0.5 point)***

∀x [MadeOf(x, Wood) ⇒ WeightSameAs(x,Duck)]
∀x [¬MadeOf(x, Wood) ∨ WeightSameAs(x,Duck)] ***(0.5 point)***
¬MadeOf(x, Wood) ∨ WeightSameAs(x,Duck) ***(0.5 point)***

∀x [Witch(x) ⇒ Burn(x)]
∀x [¬Witch(x) ∨ Burn(x)] ***(0.5 point)***
¬Witch(x) ∨ Burn(x) ***(0.5 point)***

MadeOf(Bob, Wood) ∧ ¬Sink(Bob)

(c) Are those Horn Clauses? Explain. **(1 point)**
The first one is not a Horn Clause, as it has more than one positive term. The next two are Horn Clauses, and the last one is just facts. ***(1 point if they demonstrate understanding of what a horn clause is)***

(d) Answer the following query using Resolution: Will we burn Bob? **(2 points)**

Negation of query: ¬Burn(Bob)

KB (with query):
[Sink(x1) ∨ ¬WeightSameAs(x1,Duck) ∨ Witch(x1)] ∧
[¬MadeOf(x2, Wood) ∨ WeightSameAs(x2,Duck)] ∧
[¬Witch(x3) ∨ Burn(x3)] ∧
MadeOf(Bob, Wood) ∧ ¬Sink(Bob) ∧ ¬Burn(Bob)

Possible Resolution 1:
{x3/Bob} ***(0.5 point)***
¬Witch(x3)
{x1/Bob} ***(0.5 point)***
Sink(Bob) ∨ ¬WeightSameAs(Bob,Duck)
{x2/Bob} ***(0.5 point)***
Sink(Bob) ∨ ¬MadeOf(Bob, Wood)
(No substitution) ***(0.5 point)***
Contradiction!

Possible Resolution 2:
{x2/Bob} ***(0.5 point)***
WeightSameAs(Bob,Duck)
{x1/Bob} ***(0.5 point)***
Witch(Bob)
{x3/Bob} ***(0.5 point)***
Burn(Bob)
(No substitution) ***(0.5 point)***
Contradiction!

Conclusion:
Adding the negation of the query leads to a contradiction, therefore the KB entails that we burn Bob.

Question 5 (5.5 points)

Natural Language Processing is the branch of AI that studies ways of making agents that can automatically handle and understand human languages such as English. One of the many challenges encountered in this field is how to handle word polysemy, or words with multiple different meanings. As the meaning of such words can only be understood in context, it becomes necessary to analyse the entire sentences to pick out lexical clues.

Take for example the noun "date". It can take several different meanings:

- Meaning 1: The oblong edible fruit of a palm (as in "I ate a date"). This meaning is used 25% of the time.
- Meaning 2: The time at which an event occurs (as in "The date of his birth"). This meaning is used 40% of the time.
- Meaning 3: A romantic meeting (as in "Bob went on a date last night."). This meaning is used 35% of the time.

Given the word "date" by itself, it is impossible for an agent to know which meaning the author of a text intended. But picking out keywords elsewhere in the sentence can help clarify things. For example:

- The word "fruit" is used in 80% of sentences where the word "date" is used in the first meaning, but only 5% of sentences where it has the second meaning and 15% of sentences where it has the third meaning.
- The word "calendar" occurs in 75% of sentences where the word "date" has the second meaning, but only 10% of sentences where it has the first meaning and 20% of sentences where it has the third meaning.
- The word "restaurant" is present in 85% of sentences where the word "date" takes the third meaning, 30% of sentences where it takes the first meaning, and 45% of sentences where it has the second meaning.

With statistical information such as this, one way to solve the problem of word polysemy is to design a Naïve Bayes Classifier which classifies the word "date" to its most probable meaning given the keywords found in the sentence.

(a) Design this classifier. You must start with Bayes' Theorem, and show each step of the development to get the Naïve Bayes Classifier. You must also specify the value of each variable in the final equation. **(4 points)**

M_i is the meaning of the word "date"
W_1, W_2, W_3 are "fruit", "calendar", "restaurant" respectively

Bayes Theorem
$P(M_i \mid W_1, W_2, W_3) = P(W_1, W_2, W_3 \mid M_i) * P(M_i) / P(W_1, W_2, W_3)$ ***(0.5 point)***
Chain rule:
$P(W_1, W_2, W_3 \mid M_i) * P(M_i) = P(M_i, W_1, W_2, W_3)$ ***(0.5 point)***
Conditional independence assumption:
$P(M_i, W_1, W_2, W_3) = P(M_i)*P(W_1|M_i)*P(W_2|M_i)*P(W_3|M_i)$ ***(0.5 point)***
$P(W_1, W_2, W_3)$ is independent of class, so it can be ignored ***(0.5 point)***
Final classifier:
$P(M_i \mid W_1, W_2, W_3) = P(M_i)*P(W_1|M_i)*P(W_2|M_i)*P(W_3|M_i)$ ***(1 point)***

With the following values: ***(1 point)***

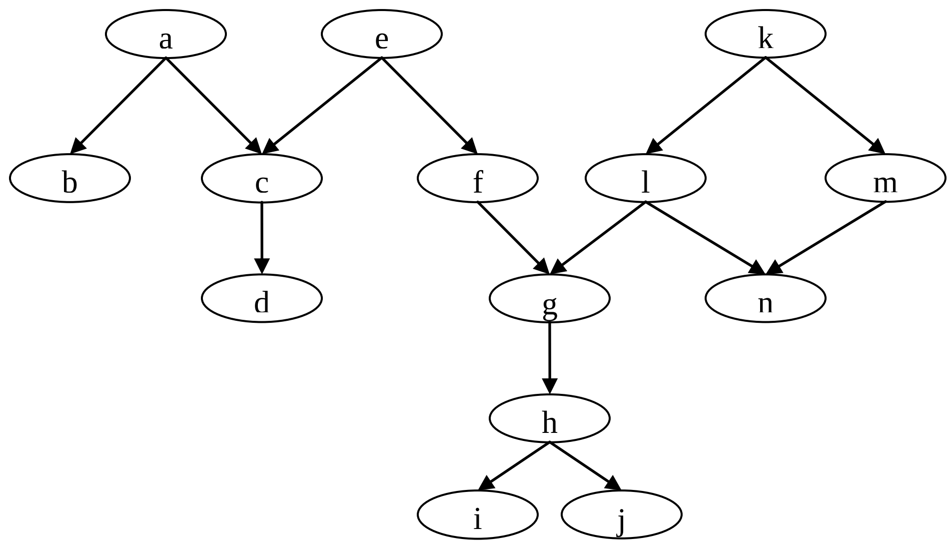
	$P(M_i)$	$P(W_1\|M_i)$	$P(W_2\|M_i)$	$P(W_3\|M_i)$
M_1	0.25	0.8	0.1	0.3
M_2	0.4	0.05	0.75	0.45
M_3	0.35	0.15	0.2	0.85

(b) Your agent encounters the following sentence: "Bob found his favourite fruit, a date, in his food at the restaurant last night." Which meaning will your agent assign to the word "date" in this case? **(1.5 point)**

$P(M_1 \mid W_1, \neg W_2, W_3) = 0.25 * 0.8 * 0.9 * 0.3 = 0.054$
$P(M_2 \mid W_1, \neg W_2, W_3) = 0.4 * 0.05 * 0.25 * 0.45 = 0.0028$
$P(M_3 \mid W_1, \neg W_2, W_3) = 0.35 * 0.15 * 0.8 * 0.85 = 0.0357$

The agent classifies it as "Meaning 1", a fruit.
(1 point for knowing they're computing $P(M_i \mid W_1, \neg W_2, W_3)$)
(.5 for getting the right result)

Question 6 (4 points)
Consider the following Bayesian Network, and use d-separation to answer the questions:



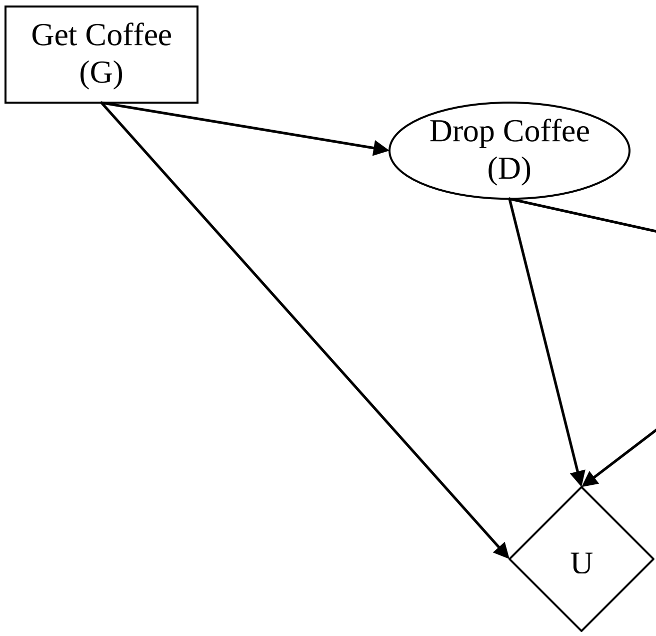
(a) If we observe a value for node **e**, what other nodes are updated?
Nodes c, d, f, g, h, i, j ***(1 point)***

(b) If we observe a value for nodes **a** and **i**, are nodes **d** and **n** independent (i.e. d-separate)?
No ***(1 point)***

(c) If we observe a value for nodes **f**, **h** and **k**, what other nodes are updated?
Nodes c, d, e, g, i, j, l, m, n ***(1 point)***

(d) If we observe a value for node **k**, are nodes **g** and **m** independent (i.e. d-separate)?
Yes ***(1 point)***

Question7 (6 points)
Consider the following decision network. The network is for a robot that can decide to go get coffee for its owner. The robot is clumsy, and there's a probability that it will drop the coffee. The robot has sensors to know when it has dropped the coffee. The robot also has the option to clean up the house.



G	P(D)
F	0
T	0.4

G	D	C	U
F	F	F	0
F	F	T	20
F	T	F	-200
F	T	T	-150
T	F	F	100
T	F	T	-50
T	T	F	-100
T	T	T	-50

Compute the optimal policy for this network. For full marks, be sure to show all necessary equations.

EU(C|D,G) = -50
EU(~C|D,G) = -100
Optimal policy, if it gets coffee and drops it, is to clean up ***(1 point)***

EU(C|~D,G) = -50
EU(~C|~D,G) = 100
Optimal policy, if it gets coffee and doesn't drop it, is to not clean up ***(1 point)***

EU(C|~D,~G) = 20
EU(~C|~D,~G) = 0
Optimal policy, if it doesn't get coffee, is to clean up ***(1 point)***

EU(G) $= \Sigma_d$ **P(d|G) U(G,d,C)**
= P(D|G) U(G,D,C) + P(~D|G) U(G,~D,~C) ***(1 point)***
= 0.4 * -50 + 0.6 * 100 = 40
EU(~G) $= \Sigma_d$ **P(d|~G) U(~G,d,C)**
= P(~D|~G) U(~G,~D,C) ***(1 point)***
= 1 * 20 = 20
Optimal policy is to get coffee ***(1 point)***

Question 8 (2 points)
Explain what a stochastic search technique is, and why it is useful.

A regular search technique explores the search space, moving from state to state by picking the state most likely to lead to a solution from among the next possible states.
A stochastic search technique uses randomness when picking the next state. ***(0.5 point for using randomness in selecting next state)***
It can therefore go to a state that is not the best one right now. ***(0.5 point for going to non-optimal states)***
This allows it to escape local optima and go to the global optimum. ***(1 point)***

Question 9 (10 points)
Short-answer questions. Give a brief answer to each of these questions.

(a) According to the Turin Test, we would have a true AI if: **(1 point)**

An observer cannot tell the difference between our agent and another human being.

(b) Imagine a backtracking depth-first search. It searches the tree in a depth-first manner, but only generates one child for each node. When it reaches a dead end, it backtracks to the last node that has unchecked children (deleting all lower nodes), generates the next child, and goes down that branch. The tree it is searching has a branching factor of b, a maximum depth of m, and the first goal node reached is at depth d. What is the space complexity of the backtracking depth-first search? **(1 point)**

O(m)

(c) Your colleague Bob just built a binary classifier (i.e. a classifier that classifies objects as belonging or not belonging to a class). He evaluated it by computing the number of objects belonging to the target class that were correctly classified as such plus the number of objects not belonging to the target class that were correctly classified as such, divided by the total number of objects. Is this a good measure? Why or why not? **(1 point)**

Counting TN would skew the statistics and favour a system that classifies everything as negatives

(d) Why are greedy search algorithms prone to ending in local optimums rather than the global optimum of the search space? **(1 point)**

They do not allow bad moves. So they head from best move to best move until they can't do any better. But if the series of best moves they are following leads to a local optimum, they cannot escape it.

(e) We never test the same attribute twice along one path in a binary decision tree. Why? **(1 point)**

Any example reaching the second test has already been tested and has a known value for that attribute. The second test is therefore redundant.

(f) Why does the average fitness of a genetic algorithm's population increase over time? **(1 point)**

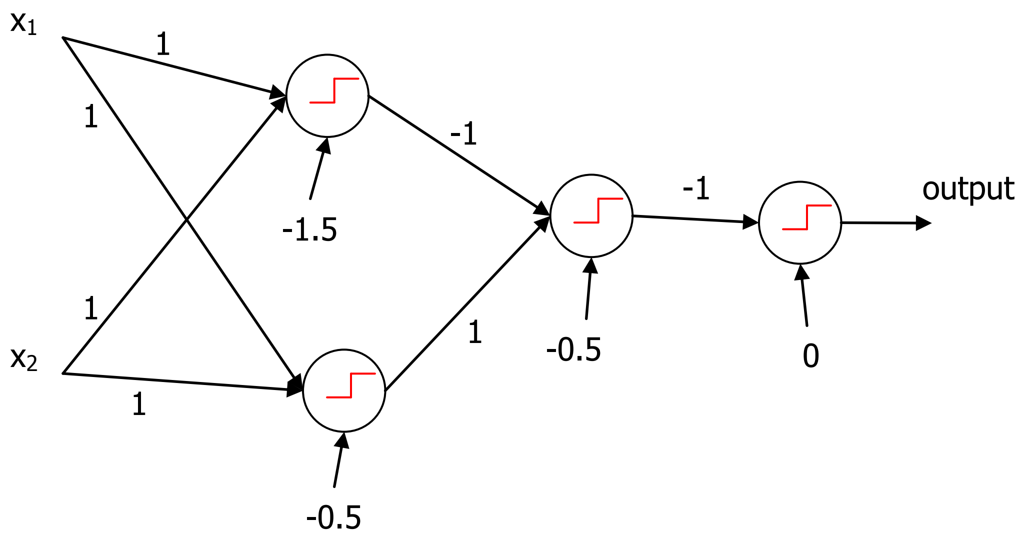
New individuals are generated and added to the population at each generation. Then, to maintain zero-population growth, an equal number of individuals are

killed off. Those are the least-fit individuals. So new, more fit individuals are replacing less-fit individual, and on average the population improves.

(g) What is the difference between a state's reward and its Q-value? **(1 point)**

The reward is the value of the state itself, regardless of the action we do in it. The Q-Value is the value of a state-action pair, or the value of doing a specific action in a specific state.

(h) What function does the following neural network represent? **(1 point)**



Inverted XOR.

(i) Is there a problem with using the shortest-path heuristic to solve inheritance conflicts in an ontology, if you know that the ontology has no redundant links? Explain. **(1 point)**

Yes. Path length is not the same as relevance. One path might simply have more detailed distinctions between the different levels than the other, making the path longer.

(j) State Bayes' Theorem

P(A|B) = P(B|A) * P(A) / P(B)

Question 10 (4 points)
We can make classifiers using either neural networks or fuzzy logic. How would we pick which one to use for a given problem? What are the conditions, the properties of the classification, that make one preferable over the other?

Neural networks give a crisp classification. They classify something as belonging to the class corresponding to the output neuron most activated, and not belonging to the other classes. It is therefore useful for classification problems where the classes are crisply defined and non-overlapping, and an item fully belongs or does not belong to the class. ***(2 points)***

Fuzzy sets allow us to measure the membership degree of an item in a set. It therefore allows us to measure how much something belongs to a class. It is therefore useful for classification problems where the classes are vaguely defined and overlap, and an item can belong partially to several classes. ***(2 points)***