Last Name: First Name:

Matriculation Number:

Exam
Artificial Intelligence 1

April 4, 2024

Please ignore the QR codes; do not write on them, they are for grading support

	To be used for grading, do not write here														
prob.	1.1	2.1	2.2	2.3	3.1	4.1	4.2	5.1	5.2	5.3	6.1	7.1	7.2	Sum	grade
total	10	8	6	7	6	5	6	6	8	7	7	6	8	90	
reached															

In the course Artificial Intelligence I/II we award bonus points for the first student who reports a factual error in an old exam. (Please report spelling/formatting errors as well.)

1 Prolog

Problem 1.1 (Analyzing a Prolog Program)
Consider the following Prolog program:

```
1 foo([X|_],X).
2 foo([_|L],X) :- foo(L,X).
3
4 good([], _, _).
5 good([X|Xs], Ys, XYs) :- goodX(X, Ys, XYs), good(Xs, Ys, XYs).
6
7 goodX(X, Ys, [pair(X,Y)|_]) :- foo(Ys, Y).
8 goodX(X, Ys, [_|XYs]) :- goodX(X, Ys, XYs).
```

1. Give a value for `xys` such that the query `good([1,2], [1,2], xys)` returns `true`. 3 Points

   Solution: For example, `[pair(1,1),pair(2,1)]`.

2. Which definition from the course does the program `good(Xs,Ys,XYs)` implement? 3 Points

   Solution: It checks if a variable with domain `Xs` is arc-consistent with respect to a variable with domain `Ys` under a constraint given by the list of pairs `XYs`.

For the remaining questions, assume we swap the lines 1 and 2.
Explain (in about 2 sentences each) how this change effects program's of the form `foo(s,t)` regarding

3. ...correctness? 2 Points

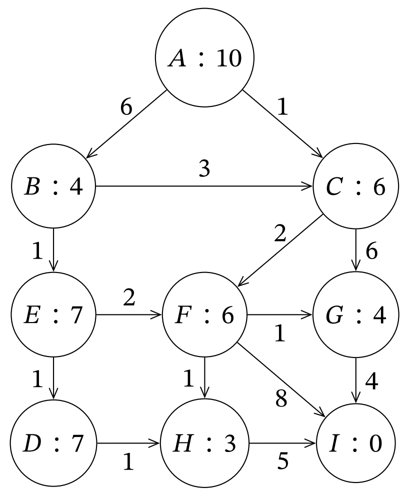
   Solution: It has no effect. `foo` outputs only `true/false` and has no side-effects, so the order of cases does not matter.

4. ...efficiency? 2 Points

   Solution: Due to Prolog's depth-first search behavior, it will first traverse the entire list without doing anything, and then backtrack to check the second case for each element of the list. The is still linear but much slower. It also requires linear space instead of constant space.

2 Search

Problem 2.1 (Search Algorithms)
Consider the following *directed graph*:



Every *node* is *labeled* with $n : h(n)$ where n is the identifier of the *node* and $h(n)$ is the *heuristic* for estimating the *cost* from n to a *goal node*.
Each node's children are ordered **alphabetically**.
Every *edge* is *labeled* with its actual *cost*.

Assume you have already *expanded* the *node* A. List the **next** 4 ***nodes*** **(i.e., excluding** A**)** that will be *expanded* using the respective *algorithm*.
If there is a tie, break it using **alphabetical order**.

1. *Depth-first search* 1 Points

   Solution: B, C, F, G

2. *Breadth-first search* 1 Points

   Solution: B, C, E, F

3. *uniform-cost search* 2 Points

   Solution: C, F, G, H

4. *greedy search* 2 Points

   Solution: B, C, G, I

5. *A*-search* 2 Points

   Solution: C, F, H, G

Problem 2.2 (Search in an Infinite Tree)
Consider an infinite tree defined as follows:

- The root has 1 child.
- Every other node has one more child than its parent.

Explain (in about 2 sentences each) what pitfalls we have to watch for (e.g., correctness, complexity, completeness, ...) when searching in this tree using ...

1. ... depth-first search. 3 Points

   Solution: The search is not complete because it infinitely recurses along the left-most branch without ever searching any other parts.

2. ... breadth-first search. 3 Points

   Solution: Because the number of nodes at depth n is $n!$, the search would be very slow for deep nodes. The queue of unexpanded nodes would run out of memory already at low depths.

Problem 2.3 (Search Problems)

Consider the family of search problems P_n given for $n = 1, 2, ...$ by $\langle S, A, T, I, G\rangle$ where

- $S = \{0, 1, \ldots, n\}$
- $A = \{f, b, s\}$
- T is given by
  - $T(f, x) = \{x \oplus 2\}$
  - $T(b, x) = \{x \ominus 2\}$
  - $T(s, x) = \{x^2\} \cap S$

  where $\oplus$ and $\ominus$ are addition/subtraction modulo n.
- $I = \{0\}$
- $G = \{n - 1\}$

1. For $n = 19$, give the result of applying the action sequence f, f, s in state 0. 2 Points

   Solution: 16

2. Under what circumstances does this problem have a solution? 2 Points

   Solution: If n is odd.

3. For $n = 15$, explain what is special about applying the action sequence f, f, s to state 0. 2 Points

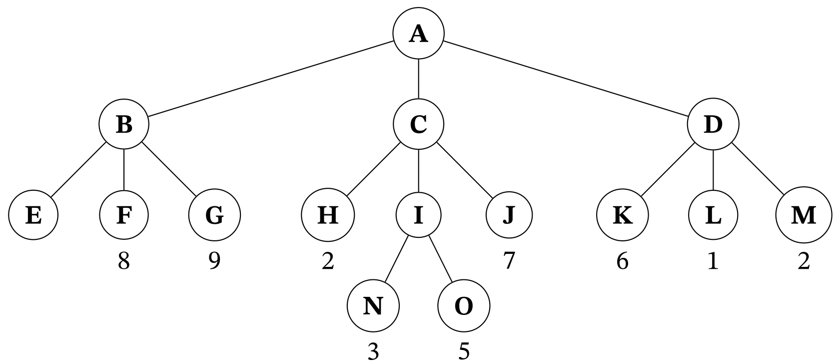
   Solution: The third action in the sequence is not applicable because $T(s, 4) = \emptyset$.

4. Explain (in 1 sentence) why this problem represents a fully observable environment. 1 Points

   Solution: I is a singleton set.

3 Adversarial Search

Problem 3.1 (Minimax)
Consider the following *minimax* game tree for the **maximizing** player's turn. The *values* at the *leaves* are the static *evaluation function values* of those *states*; some of those *values* are currently missing.



1. Label the *nodes* I and C with their *minimax* value. 2 Points

   Solution: I: 5, C: 2

2. If possible, label the *node* **E** with an *evaluation function value* that results in the player definitely choosing move **C** (no matter how ties are broken).
   Otherwise, argue why that is impossible. 2 Points

   Solution: Any label < 2.

3. Now assume **E** is *labeled* with 5, and we use *αβ-pruning*. We expand *child nodes* in alphabetical order. Which nodes would be *pruned*? 2 Points

   Solution: I, N, O, J, M (or I,J,M)

4 Constraint Satisfaction

Problem 4.1 (Assignments and Solutions)
Consider the CSP given by

- Variables: $\{a, b, c, d\}$
- Domains: $D_a = D_b = \{0, 1, 2, 3\}$ and $D_c = D_d = \{0, 1, 2, 3, 4, 5\}$
- Constraints:
  - ☐ $a < b$
  - ☐ $b < d - c$
  - ☐ $d - a > 3$
  - ☐ $d = 2c$

1. Check the constraints that make this CSP non-binary. 1 Points

Solution: Only the second one (because it is ternary).

2. Give the unique *solution* to this *CSP*. 2 Points

Solution: $(a, b, c, d) = (0, 1, 2, 4)$

3. Check the true statements: 2 Points
   - ☐ The *assignment* $a = 0$, $b = 1$ is *consistent*.
   - ☐ The *assignment* $a = 0$, $b = 1$, $c = 2$, $d = 3$ is *consistent* and *total*.
   - ☐ The *assignment* $b = 0$, $c = 2$, $d = 4$ is *consistent* and *total*.

Solution: Only the first statement is true.

Problem 4.2 (Relating CSP and SAT)
Like in the homework problem, we want to relate CSP and SAT.
Assume a binary CSP with variables $X_1, \ldots, X_n$, a finite domain D_i for every X_i, and a constraint $C_{ij} \subseteq D_i \times D_j$ for every $1 \leq i < j \leq n$. A solution of this CSP instance is an assignment α mapping each X_i to a value in D_i.

1. Give an instance of SAT, i.e., a set P of propositional variables and a propositional formula F over P, that is equivalent to this CSP problem. 4 Points

Solution:

$$P = \{P_{iv} : 1 \leq i \leq n,\ v \in D_i\}$$
$$F = \bigwedge_{1 \leq i \leq n} A_i \wedge \bigwedge_{1 \leq i < j \leq n} B_{ij}$$

where

$$A_i = \bigvee_{u \in D_i} P_{iu} \wedge \bigwedge_{u,v \in D_i, u \neq v} \neg(P_{iu} \wedge P_{iv})$$

(expresses that A_i has has exactly one value)

$$B_{ij} = \bigvee_{(u,v) \in C_{ij}} P_{iu} \wedge P_{jv}$$

(expresses that C_{ij} is satisfied)

2. To show the equivalence, for every assignment α for the CSP instance, give an assignment φ for the SAT instance defined in the previous subproblem such that F is satisfied under φ iff all constraints are satisfied under α. 2 Points

Solution: $\varphi(P_{iv}) = T$ iff $v = \alpha(X_i)$

5 Logic

Problem 5.1 (Propositional Logic)
Consider the formula $A = \big((p \vee q) \Rightarrow (p \wedge q)\big) \wedge \big((p \vee \neg q) \Rightarrow (p \vee \neg r)\big)$ using propositional variables p, q, r.

1. Give all satisfying assignments for A. 2 Points

Solution: $\langle \varphi(p), \varphi(q), \varphi(r) \rangle \in \{\langle F, F, F \rangle, \langle T, T, F \rangle, \langle T, T, T \rangle\}$

2. Give a formula in DNF that is equivalent to A. 2 Points

Solution: $(p \wedge q \wedge \neg r) \vee (\neg p \wedge \neg q \wedge \neg r) \vee (p \wedge q \wedge r)$
(This can be read off off the list of satisfying assignments. Reading off a faulty list was also accepted.)

3. Turn A into a theorem by replacing exactly one occurrence of a connective with a different connective. 2 Points

Solution: Change the $\wedge$ in the middle to $\vee$.
(Reasoning: Neither conjunct of A is a theorem, so changing a connective inside one conjunct cannot suffice. So we need to change the $\wedge$. Replacing with $\vee$ or $\Rightarrow$ has the best chance of producing a theorem because they are true more often than $\wedge$. Only $\vee$ works.)

Problem 5.2 (Modeling in First-Order Logic)
Consider the following situation:

- Some individuals are persons, some are animals.
- Persons and animals may like other persons or animals.
- Alice is a person, and she likes the animal Bubbles.
- For every person or animal, we can obtain its mother.

1. Model this situation in first-order logic by giving a signature, i.e., a list of function/predicate symbols with arity. 4 Points

---

Solution:

- nullary function symbols: A (for Alice), B (for Bubbles)
- unary function symbols: m (for mother)
- unary predicate symbols: p (for person), a (for animal)
- binary predicate symbols: l (for like)

---

2. State formulas over your signature that capture the following properties: 2 Points
   1. Every individual is a person or an animal but not both.
   2. Every mother likes their offspring.

Solution:

1. $\forall x.p(x) \vee a(x), \forall x.\neg(p(x) \wedge a(x))$
2. $\forall x.l(m(x), x)$

---

3. Give a model over your signature, i.e., a domain D and an interpretation $I(s)$ for every function/predicate symbol, in which all properties from the above subproblem hold. 2 Points

---

Solution: E.g., $D = \mathbb{N}$, $I(A) = 0$, $I(B) = 1$, $I(m)(n) = n + 2$, $I(p) = even$, $I\ a = odd$, $I(l) = \{(0,1)\} \cup \{(n+2, n) : n \in \mathbb{N}\}$.

Problem 5.3 (Tableaux Calculus)
Consider the following tableau where $A = (p \wedge q) \vee (\neg p \wedge \neg q)$.

$$\begin{array}{c}
A^F \\ \hline
(p \wedge q)^F \\
(\neg p \wedge \neg q)^F \\
\begin{array}{c|c}
\begin{array}{c} p^F \\ \begin{array}{c|c} (\neg p)^F & (\neg q)^F \\ p^T & q^T \end{array} \end{array} &
\begin{array}{c} q^F \\ \begin{array}{c|c} (\neg p)^F & (\neg q)^F \\ p^T & q^T \end{array} \end{array}
\end{array}
\end{array}$$

1. How can you tell that this tableau is saturated? 2 Points

Solution: For every node, all subnodes resulting from applying the respective tableau rule are already on the branches emanating from it.

---

2. Explain how, using the tableau, we can find all satisfying assignments for A. 2 Points

---

Solution: Each branch that cannot be closed induces a falsifying assignment for A (because we started with A^F) by collecting the atomic formulas on it. Counting branches left to right, branch 1 can be closed on p and branch 4 on q. Branch 2 yields $\varphi(p) = F$, $\varphi(q) = T$, and branch 3 yields $\varphi(p) = T$, $\varphi(q) = F$.
The satisfying assignments are the other assignments, i.e., $\varphi(p) = \varphi(q) = T$ and $\varphi(p) = \varphi(q) = F$.

---

3. Give the fully saturated tableau for the root A^T. 3 Points

Solution:

$$\begin{array}{c|c} \multicolumn{2}{c}{A^T} \\ \hline p \wedge q^T & \neg p \wedge \neg q^T \\ p^T & \neg p^T \\ q^T & \neg q^T \\ & p^F \\ & q^F \end{array}$$

6 Knowledge Representation

Problem 6.1 (ALC)

Consider the following description logic signature

- concept symbols: i (for instructor), s (for student), c (for course), p (for program)
- role symbol m (for is-member-of) used for
  - instructors giving a course
  - students taking a course
  - students being enrolled in a program
  - courses being part of a program

We use an extension of ALC, in which there are dual roles: there is a role m^{-1} that captures the relation has-as-member, e.g., $MK\ m\ AI$ iff $AI\ m^{-1}\ MK$.

1. Give an axiom for the above signature that captures that instructors can only be members of courses. 1 Points

Solution: $i \sqsubseteq \forall m.c$

2. Give an axiom for the above signature that captures: courses that are taken by a student, must be given by an instructor. 2 Points

Solution: $c \sqcap \exists m^{-1}.s \sqsubseteq \exists m^{-1}.i$

3. Calculate the translation to first-order logic of $s \sqsubseteq (\forall m.\exists m.p)$. 2 Points

Solution: $\forall x.s(x) \Rightarrow \forall y.m(x,y) \Rightarrow \exists z.m(y,z) \wedge p(z)$

4. Given a model $\langle D, [\![-]\!]\rangle$, define an appropriate case of the interpretation mapping for the formula $\forall r^{-1}.C$. 2 Points

Solution: $[\![\forall r^{-1}.C]\!] = \{u \in D \mid \text{ for } v \in D, \text{ if } (v,u) \in [\![r]\!], \text{ then } v \in [\![C]\!]\}$

7 Planning

Problem 7.1 (Admissible Heuristics in Gripper)
Consider the following situation from the homeworks:

- We have two rooms, A and B, one robot initially located in room A, and n balls that are initially located on the floor in room A.
- The goal is to have all balls on the floor in room B.
- The robot can move between the rooms and has a gripper for picking up and releasing balls. Moving, picking up, and releasing are one action each.

Given state s, we write s_A and s_B for the number of balls on the floor in room A or B, i.e., $n - s_A - s_B$ is the number of balls held by the robot.

For each of the following heuristics, argue (by informal proof or counter-example) whether they are admissible.

1. Assume the gripper can hold up to **1 ball** at a time. 2 Points
   Potential heuristic: $h(s) = n - s_B$.

   Solution: Admissible. $h^*(s) > n - s_B = h(s)$ because every ball not yet in room B requires at least one release action.

2. Assume the gripper can hold up to **1 ball** at a time. 2 Points
   Potential heuristic: $h(s) = 4 \cdot s_A$.

   Solution: Not admissible. Counter-example: $s_A = 1$, $s_B = n - 1$, robot in room A (holding 0 balls). $h^*(s) = 3 < 4 = h(s)$ because the robot must perform 3 actions for the last ball (pick up, move, release).

3. Assume the gripper can hold up to **2 balls** at a time, which are picked up **in one action** and released in one action. 2 Points
   Potential heuristic: $h(s) = n - s_B$.

   Solution: Not admissible. Counter-example: $s_A = 0$, $s_B = n - 2$, robot in room B (holding 2 balls). Then $h^*(s) = 1 < 2 = h(s)$.

Problem 7.2 (Relaxation)
We want to solve a STRIPS planning task.

1. Explain (in about 2 sentences) the purpose of relaxed planning. 2 Points

   Solution: A relaxed problem is easier to solve than the original problem. Then the length of its optimal plan of the relaxed problem can be used as a heuristic for the original problem.

2. Explain (in about 2 sentences) why it is bad to relax too much or too little. 2 Points

Solution: If we relax too little, the relaxed problem is still too difficult to solve. If we relax too much, the relaxed problem is so easy that it does not induce a useful heuristic.

Now consider a concrete task given by a finite set B of size n and

- facts: $inA(b)$, $inB(b)$, $held(b)$ for $b \in B$, $Rfree$, $RinA$, $RinB$
- actions

action	precondition	add list	delete list
$move_A$	$RinB$	$RinA$	$RinB$
$move_B$	$RinA$	$RinB$	$RinA$
$pickup(b)$	$RinA, Rfree, inA(b)$	$held(b)$	$Rfree, inA(b)$
$release(b)$	$RinB, held(b)$	$inB(b), Rfree$	$held(b)$

- initial state I: $inA(b)$ for $b \in B$, $Rfree$, $RinA$
- goal state: $inB(b)$ for $b \in B$

3. Let h^+ be the heuristic obtained from the delete-relaxation. Give the value of $h^+(I)$. 2 Points

Solution: $2n + 1$ ($move_B, pickup(1), release(1), \dots, pickup(n), release(n)$)

4. Let h^+ be the heuristic obtained from the only-adds-relaxation. Give the value of $h^+(I)$. 2 Points

Solution: n ($release(1), \dots, release(n)$)