Last Name: First Name

Matriculation Number: Birth Date:

Seat:

Exam
Artificial Intelligence 1

Feb 14, 2022

	To be used for grading, do not write here													
prob.	1.1	2.1	2.2	3.1	4.1	4.2	5.1	5.2	5.3	6.1	7.1	7.2	Sum	grade
total	10	8	7	6	6	7	6	8	8	9	12	8	95	
reached														

Exam Grade: Bonus Points: Final Grade:

The "solutions" to the exam/assignment problems in this document are supplied to give students a starting point for answering questions. While we are striving for helpful "solutions", they can be incomplete and can even contain errors even after our best efforts.
In any case, grading student's answers is not a process of simply "comparing with the reference solution", therefore errors in the "solutions" are not a problem in this case.
If you find "solutions" you do not understand or you find incorrect, discuss this on the course forum and/or with your TA and/notify the instructors. We will – if needed – correct them ASAP.

In the course Artificial Intelligence I/II we award 5 bonus points for the first student who reports a factual error (please report spelling/formatting errors as well) in an assignment or old exam and 10 bonus points for an alternative solution (formatted in LaTeX) that is usefully different from the existing ones.

1 Prolog

Problem 1.1 (Prolog) 10 pt

Consider the following Prolog code:

```
isNat(zero).
isNat(succ(N)) :- isNat(N).

isTree(tree(N,Ts)) :- isNat(N), isTrees(Ts).

isTrees([]).
isTrees([H|T]) :- isTree(H), isTrees(T).

evaluateNat(zero,0).
evaluateNat(succ(N),X) :- evaluateNat(N,Y), X is Y+1.

% If T is a tree, treeSum(T,S) holds if
% S is the sum of the labels of the nodes (evaluated into Prolog integers) .
treeSum(                    ) :-




treeListSum([],S) :-


treeListSum([H|T],S) :-

```

1. What do the following queries return: 3 pt

   (a) `isTree(tree(succ(zero),[tree(zero,[]), tree(succ(zero),[])]))`

   (b) `isTree(X)`

   If a query returns multiple results, only give the first two.

2. Complete the implementation of the predicate `treeSum`. 5 pt
   Hint: Use `treeListSum` as an auxiliary predicate.

3. Assume we change the `isTree` predicate to 2 pt

   ```
   isTree(tree(N,Ts)) :- write(N), isNat(N), isTrees(Ts).
   ```

   Why and how does Prolog's search behavior matter now?

Solution:

1. (a) True

   (b) X = tree(zero, []), X = tree(zero, [tree(zero, [])])

2. ```
   isNat(zero).
   isNat(succ(N)) :- isNat(N).

   isTree(tree(N,Ts)) :- isNat(N), isTrees(Ts).

   isTrees([]).
   isTrees([H|T]) :- isTree(H), isTrees(T).

   evaluateNat(zero,0).
   evaluateNat(succ(N),X) :- evaluateNat(N,Y), X is Y+1.

   % If T is a tree, treeSum(T,S) holds if
   % S is the sum of the labels of the nodes (evaluated into Prolog integers).
   treeSum(tree(N,Ts),S) :- evaluateNat(N,E), treeListSum(Ts,F), S is E+F.

   % If Ts is a list of tree, treesSum(Ts,S) if S is the sum of nodeSum of all trees.
   treeListSum([],S) :- S is 0.
   treeListSum([H|T],S) :- treeSum(H,E), treeListSum(T,F), S is E+F.
   ```

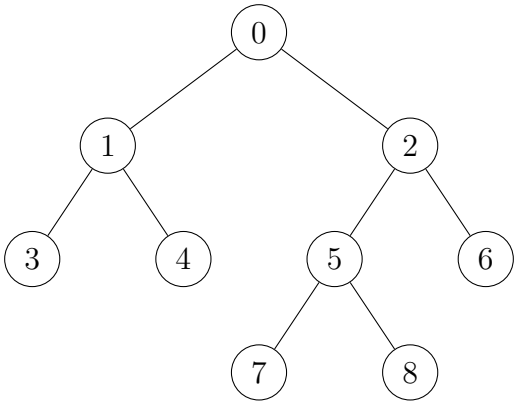
3. The predicate has a side-effect. So the search order matters. Prolog searches depth-first, so each call to isTree in the predicate isTrees is processed before moving to the next tree in the list. Effectively, the node labels are printed in depth-first order.

---

2 Search

Problem 2.1 (Search Algorithms) 8 pt

Consider the following tree with root 0:


   ```

Every edge from parent node i to child node j has path cost i.
As a heuristic for estimating the distance from node n to a goal node, we use $h(n) = 8 - n$.

Assume you have already expanded the root node. List the **next** 4 nodes that will be expanded using

1. depth-first search 1 pt
2. breadth-first search 1 pt
3. uniform-cost search 2 pt
4. greedy search 2 pt
5. A^*-search 2 pt

If there is a tie, first expand the node with the smaller number.

Solution:

1. $1, 3, 4, 2$
2. $1, 2, 3, 4$
3. $1, 2, 3, 4$
4. $2, 6, 5, 8$
5. $2, 6, 5, 1$

Problem 2.2 (Search Problems) 7 pt

Consider the search problem (S, A, T, I, G) where

- $S = \mathbb{Z}$
- $A = \{-6, -4, 0, 5\}$
- $T(a, s) = \{a + s\}$
- $I = \{0\}$
- $G = \{7\}$

1. Give the state resulting from applying the action sequence $-6, -4, 5$ to the initial state. 1 pt
2. Give a solution to the problem. 3 pt
3. Is DFS a good choice for this problem and why (not)? 2 pt

4. If we change the set A to $\{-6, -4, 0, 2\}$, the problem changes substantially. In what way? 1 pt

---

Solution:

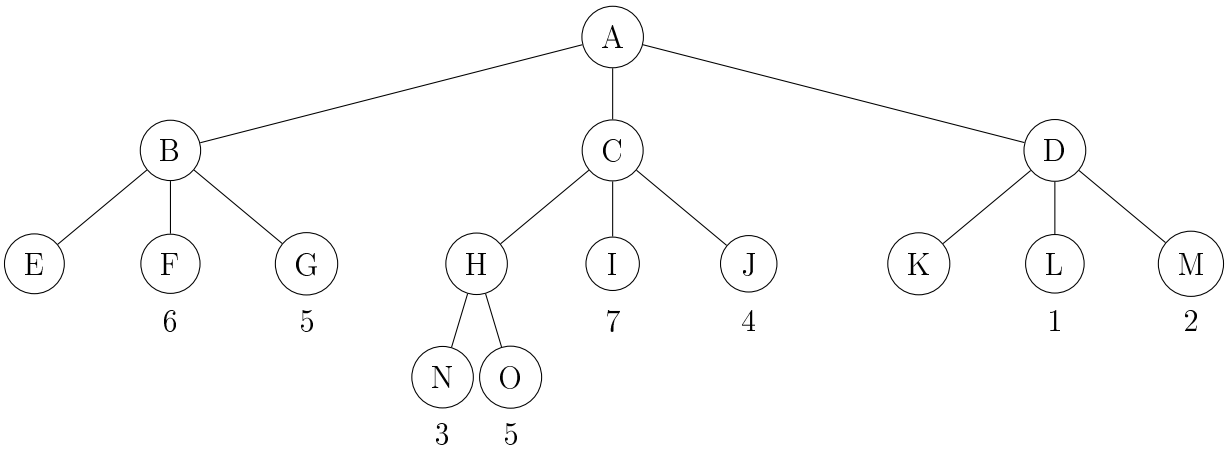
1. -5
2. $5, 5, 5, -4, -4$ (any sequence of actions that add up to 7)
3. No. It will not find a solution: whichever action we try first, can be applied infinitely often.
4. There is no solution.

---

3 Adversarial Search

Problem 3.1 (Minimax) 6 pt

Consider the following minimax game tree (**without** alpha-beta pruning) for the **maximizing player**'s turn. The values at the leafs are the static evaluation function values of those states; some of those values are currently missing.



1. Label the nodes H and C with their minimax values. 2 pt
2. Label the node E with an evaluation function value that results in the player choosing move B. 2 pt
3. Now assume E is labeled 1.
   Label the node K with an evaluation function value that results in α-β-search pruning (i.e., not visiting) L and M.
   Assume children of a node are visited in alphabetical order. 2 pt

---

Solution:

1. H= 5, C= 4

2. Anything > 4.

3. Anything ≤ 4.

---

4 Constraint Satisfaction/Propagation

Problem 4.1 (Modeling) 6 pt

You are designing an exam consisting of 4 questions, and you need to assign a positive integer point value to each question.
The total points of the exam should be 20.
Questions 1 and 2 are hard and should contribute at most 10 points together.
Questions 2 and 3 cover essential topics and should contribute at least 12 points together.
Question 4 is long and should contribute more points than **any** of the others.

Model this problem as a (not necessarily binary) constraint satisfaction problem (V, D, C). Briefly explain the meaning of the variables.

Note: Make sure you give a formally exact definition, i.e., explicitly define the sets V and all sets D_v. You can describe each constraint as a set of tuples or as a formula.

Solution: $V = p_1, p_2, p_3, p_4$
$D_{p_i} = \{1, 2, \dots, \}$ for all i
The sets $\{1, \dots, n\}$ for $17 \leq n$ were also accepted for the domains. Constraints in C:

- $p_1 + p_2 + p_3 + p_4 = 20$
- $p_1 + p_2 \leq 10$
- $p_2 + p_3 \geq 12$
- $p_4 > p_i$ for $i = 1, 2, 3$

p_i is the number of points of question i.

Problem 4.2 (Solving) 7 pt

Consider the following binary CSP:

- $V = \{a, b, c, d\}$
- $D_a = \texttt{bool}$, $D_b = D_c = \{0, 1, 2, 3\}$, $D_d = \{0, 1, 2, 3, 4, 5, 6\}$
- Constraints:
  - if a, then $b \leq 2$
  - if $c < 2$, then a
  - $b + c < 4$
  - $b > d$

- $d = 2c$

1. Give all pairs (v, w) of variables such that v is **not** arc-consistent relative to w. 3 pt
2. Give a solution. 1 pt
3. Give an inconsistent total assignment to the variables. 1 pt
4. Assume we assign a to be true, and apply forward-checking. Give the resulting domains D_b and D_c. 2 pt

---

Solution:

- (b, d), (d, b), (d, c)
- There are 2 solutions: a true, $b \in \{1, 2\}$, $c = 0$, $d = 0$
- Any total assignment that is not a solution, e.g., a false, $b = 0$, $c = 0$, $d = 0$
- $D_b = \{0, 1, 2\}$, $D_c = \{0, 1, 2, 3\}$

---

5 Logic

Problem 5.1 (Propositional Logic) 6 pt

We use the propositional variables P, Q, R. Consider the formula A given by

$$(P \vee Q) \Rightarrow \neg(Q \wedge (R \Rightarrow Q))$$

1. Using the assignment $\varphi(P) = T$, $\varphi(Q) = F$, and $\varphi(R) = T$, give the value $I_\varphi(A)$. 2 pt
2. Argue whether A is valid (i.e., give a proof or a counter-example). 4 pt

---

Solution:

1. T
2. It is not valid. Any assignment with $\varphi(Q) = T$ falsifies it and is thus a counter-example.

Problem 5.2 (Definitions) 8 pt

Consider the following signature of first-order logic:

- unary function symbol f
- unary predicate symbol P

Give counter-examples for the following statements:

1. If a formula A is not a theorem, then $\neg A$ is a theorem. 2 pt

2. Every model that satisfies $\forall x.P(f(x))$ also satisfies $\forall x.P(x)$. 3 pt

3. In the model given by universe $\mathbb{N}$, $I(P) = \{n \in \mathbb{N} | n > 5\}$, and $I(f)(n) = n + 1$, we have $I_\varphi\Big(\big(P(f(x)) \wedge \neg P(y)\big) \Rightarrow P(f(y))\Big) = T$ for all assignments φ. 3 pt

---

Solution:

1. Any non-theorem non-contradiction, e.g., $F = \forall x.P(x)$
2. Any model with $\texttt{image}\, I(f) \subseteq I(P) \subset U$, e.g., $U = \mathbb{N}$, $I(P) = \{0\}$, $I(f)(u) = 0$
3. Any assignment with $\varphi(x) \geq 5$ and $\varphi(y) \leq 4$

Problem 5.3 (Proving in Natural Deduction) 8 pt

Complete the following sequent-style natural deduction proof by filling in all boxes:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\Gamma \vdash \forall x.P(x) \Rightarrow \neg Q(x)}Ax}{\Gamma \vdash \square}\forall E \quad \dfrac{\dfrac{}{\Gamma \vdash \exists y.P(y)}Ax}{\Gamma \vdash \square}\exists E}{\Gamma \vdash \square}\Rightarrow E \quad \dfrac{\dfrac{}{\Gamma \vdash \forall z.Q(z)}Ax}{\Gamma \vdash \square}\forall E}{\Gamma \vdash \square}falseI}{\Gamma \vdash \square}\neg I}{\square}\Rightarrow I}{\square}\Rightarrow I$$

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

where we abbreviate

$$\Gamma = \square$$

Solution:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\Gamma \vdash \forall x.P(x) \Rightarrow \neg Q(x)}Ax}{\Gamma \vdash P(c) \Rightarrow \neg Q(c)}\forall E \quad \dfrac{\dfrac{}{\Gamma \vdash \exists y.P(y)}Ax}{\Gamma \vdash P(c)}\exists E}{\Gamma \vdash \neg Q(c)}\Rightarrow E \quad \dfrac{\dfrac{}{\Gamma \vdash \forall z.Q(z)}Ax}{\Gamma \vdash Q(c)}\forall E}{\Gamma \vdash false}falseI}{\forall x.P(x) \Rightarrow \neg Q(x),\ \exists y.P(y) \vdash \neg\forall z.Q(z)}\neg I}{\forall x.P(x) \Rightarrow \neg Q(x) \vdash (\exists y.P(y)) \Rightarrow \neg\forall z.Q(z)}\Rightarrow I}{\vdash (\forall x.P(x) \Rightarrow \neg Q(x)) \Rightarrow \big((\exists y.P(y)) \Rightarrow \neg\forall z.Q(z)\big)}\Rightarrow I$$

where we abbreviate

$$\Gamma = \forall x.P(x) \Rightarrow \neg Q(x),\ \exists y.P(y),\ \forall z.Q(z)$$

---

6 Knowledge Representation

Problem 6.1 (Specifying Properties in ALC) 9 pt

Consider the following ALC setting:

- concepts: `human`, `child`, `grownup`, `animal`
- relations: `isChildOf`, `owns`

We abbreviate every concept/relation by its first letter.

1. Give an ALC ABox that is not consistent with the axiom $h \sqcap \exists o.a = \bot$. 2 pt
2. Give an ALC TBox that formalizes the following properties 3 pt
   - Children and grownups are humans, and humans are children or grownups. No one is both a child and a grownup.
   - Humans cannot be owned.
3. Give an ALC formalization for the concept of children who have a parent who owns an animal. 2 pt
4. Give the translation to first-order logic of the ALC statement $(\forall i.h) \sqsubseteq (\exists i.g)$. 2 pt

---

Solution:

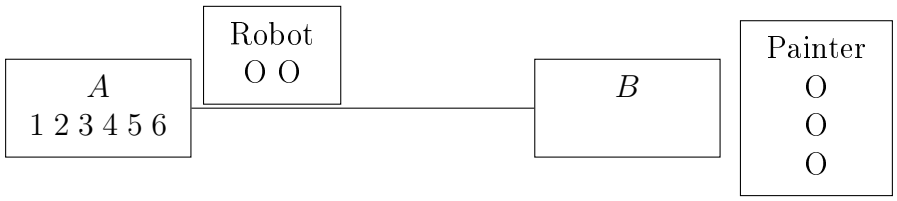
1. $x : h,\ y : a,\ x\ o\ y$
2. We use the axioms $h = c \sqcup g$, $c \sqcap g = \bot$, and $\exists o.h = \bot$
3. $c \sqcap \exists i.\exists o.a$
4. $\forall x.\big(\forall y.i(x,y) \Rightarrow h(y)\big) \Rightarrow \big(\exists y.i(x,y) \wedge g(y)\big)$

---

7 Planning

Problem 7.1 (STRIPS) 12 pt

Consider a set of objects $Obj = \{1, 2, 3, 4, 5, 6\}$ that can be at location A or B. Currently all objects are at location A and **unpainted**. Eventually all objects are needed in location A and **painted**. At location B, a painting station is available that can paint up to 3 objects at a time. A robot is available (currently at location A) that can move up to 2 objects at a time from one location to another.



We formalize this problem as a STRIPS task (P, A, I, G) where the set P of facts contains

- $\texttt{at}(l, o)$ for $l \in \{A, B\}$ and $o \in Obj \cup \{Robot\}$
- $\texttt{painted}(o)$ for $o \in Obj$

and the set A of actions contains

- $\texttt{move}(l, m, O)$ for $l, m \in \{A, B\}$, $O \subseteq Obj$, $|O| \leq 2$ given by
  - precondition: $\texttt{at}(l, o)$ for all $o \in O \cup \{Robot\}$
  - add list: $\texttt{at}(m, o)$ for all $o \in O \cup \{Robot\}$
  - delete list: same as precondition
- $\texttt{paint}(O)$ for $O \subseteq Obj$, $|O| \leq 3$ given by
  - precondition: $\texttt{at}(B, o)$ for all $o \in O$
  - add list: $\texttt{painted}(o)$ for all $o \in O$
  - delete list: nothing

1. Give the initial state I and the goal G. 2 pt
2. After applying $\texttt{move}(A, B, \{1, 2\})$ in I, multiple actions are applicable. Give two of them. 3 pt
3. Give the value $h^*(I)$. 2 pt
4. Give the value $h^+(I)$. 2 pt
5. Let $U_s(l)$ and $P_s(l)$ be the numbers of unpainted and painted objects at location l in state s. For each of the following heuristics $h(s)$, say if it is admissible. 3 pt

(a) $2 \cdot U_s(A) + U_s(B)$

(b) 0

(c) $U_s(A) + \texttt{roundDown}((U_s(A) + U_s(B))/3) + \texttt{roundDown}((P_s(B) + U_s(B))/2)$

Solution:

1. initial state: $\texttt{at}(A, o)$ for all $o \in Obj \cup \{Robot\}$, goal: $\texttt{at}(A, o)$, $\texttt{painted}(o)$ for all $o \in Obj$
2. The applicable actions are $\texttt{move}(B, A, O)$ and $\texttt{paint}(O)$ for any $O \subseteq \{1, 2\}$. Note: Among those, $\texttt{move}(A, B, \emptyset)$ and $\texttt{paint}(\{1, 2\})$ are the not-obviously-suboptimal ones and pondering those helps with the next subquestion.
3. 9. Note: An optimal plan is $\texttt{move}(A, B, O)$, $\texttt{paint}(O)$, $\texttt{move}(B, A, O)$, repeated 3 times for disjoint sets O. $\texttt{move}(A, B, \{1, 2\})$, $\texttt{move}(B, A, \emptyset)$, $\texttt{move}(A, B, \{3, 4\})$, $\texttt{paint}(\{1, 2, 3\})$, $\texttt{move}(B, A, \{1, 2\})$, $\texttt{move}(A, B, \{5, 6\})$, $\texttt{paint}(\{4, 5, 6\})$, $\texttt{move}(B, A, \{3, 4\})$, $\texttt{move}(A, B, \emptyset)$, $\texttt{move}(B, A, \{5, 6\})$ takes 10 steps and is not optimal, but induces an optimal relaxed plan, in which some moves actions can be dropped.
4. 5. Note: An optimal relaxed plan moves 2 objects 3 times, paints twice.
5. 2 and 3. Note:
   (a) too pessimistic, e.g., two unpainted objects in location A need 3 steps, heuristics yields 4
   (b) trivially admissible but useless
   (c) a good heuristic: unpainted objects in location A (resp. any objects in location B) need to be moved at least twice (resp. once) in groups of at most 2; unpainted objects must be painted in groups of at most 3.

Problem 7.2 (Partial Order Planning) 8 pt

Consider the planning task (P, A, I, G) where

- facts $P = \{p, q, r, s\}$
- actions $A = \{X, Y, Z\}$ where the preconditions (above the box) and effects (below the box) of the actions are given by

p	q	p
X	Y	Z
q	$\neg p, r$	$\neg p, s$

- initial state $I = \{p\}$
- goal $G = \{r, s\}$

Our goal is to build a partially ordered plan. Recall that the steps consist of the actions plus the start and finish step; and that the effect of an action consists of the added facts and the negations of the deleted facts.

1. Give the start step and finish step. 2 pt
2. Give all causal links between the steps. 2 pt
3. Give an example of a step that clobbers a link. 2 pt
4. Give the temporal ordering that yields a partially ordered plan that solves the task. 2 pt

---

Solution:

1. $\underset{p}{\boxed{\text{Start}}} \quad \overset{r,s}{\boxed{\text{Finish}}}$
2. $Start \xrightarrow{p} X$, $Start \xrightarrow{p} Z$, $X \xrightarrow{q} Y$, $Y \xrightarrow{r} Finish$, $Z \xrightarrow{s} Finish$
3. Both steps Y and Z clobber both of the links $\xrightarrow{p}$.
4. $X \prec Z$ and $Z \prec Y$

When reusing this question, note that (while not relevant in this case) in general the same action can occur multiple times as different steps.