Name:

Birth Date:

Matriculation Number:

Exam
Artificial Intelligence 1

July 19, 2021

	To be used for grading, do not write here												
prob.	1.1	2.1	2.2	2.3	3.1	3.2	4.1	4.2	5.1	5.2	6.1	Sum	grade
total	15	8	5	8	7	8	6	9	9	8	12	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.
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.
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.

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 (Reading and Writing Prolog) 15 pt

Note: The negation-normal-form of a formula F is a formula equivalent to F in which negations only occur immediately in front of propositional variables. For example, the negation-normal form of $\neg(P \wedge Q)$ is $\neg P \vee \neg Q$.

Consider the following partial Prolog program for computing the negation-normal form:

```
contains([H|_],H).
contains([_|T],X) :- contains(T,X).

isForm(pv(A))     :- contains(["p", "q"], A).
isForm(conj(F,G)) :- isForm(F), isForm(G).
isForm(disj(F,G)) :- isForm(F), isForm(G).
isForm(neg(F))    :- isForm(F).


nnf(pv(A),H)           :- _____________________________________.

nnf(conj(F,G), H)      :- _____________________________________.

nnf(disj(F,G), H)      :- _____________________________________.

nnf(neg(pv(A)),H)      :- _____________________________________.

nnf(neg(neg(F)), H)    :- _____________________________________.

nnf(neg(conj(F,G)), H) :- _____________________________________.

nnf(neg(_______________), H) :- _____________________________________.
```

4 pt

1. Give the first three results in order that are returned by the query `isForm(F)`? 3 pt
2. Complete the following query such that it can be used as a test case for the program:

   ```
   nnf(neg(conj(pv("p"),neg(pv("p")))), _____________________________________)
   ```

   8 pt
3. Complete the program such that `nnf(F,H)` computes the negation-normal-form of F.

Solution:

1. `pv("p"), pv("q"), conj(pv("p"),pv("p"))`

2. `nnf(neg(conj(pv("p"),neg(pv("p")))),  disj(neg(pv("p")), pv("p"))  )`

3. ```
   contains([H|_],H).
   contains([_|T],X) :- contains(T,X).

   isForm(pv(A)) :- contains(["p", "q"], A).
   isForm(conj(F,G)) :- isForm(F), isForm(G).
   isForm(disj(F,G)) :- isForm(F), isForm(G).
   isForm(neg(F)) :- isForm(F).

   nnf(pv(A),H) :- H = pv(A).
   nnf(conj(F,G), H) :- nnf(F,F2), nnf(G,G2), H = conj(F2,G2).
   nnf(disj(F,G), H) :- nnf(F,F2), nnf(G,G2), H = disj(F2,G2).

   nnf(neg(pv(A)),H) :- H = neg(pv(A)).
   nnf(neg(neg(F)), H) :- nnf(F,H).
   nnf(neg(conj(F,G)), H) :- nnf(disj(neg(F),neg(G)), H).
   nnf(neg(disj(F,G)), H) :- nnf(conj(neg(F),neg(G)), H).
   ```

Grading:

1. 1 point per result and 1 point for the order
2. deductions for mistakes
3. 1 point per blank; -1 for various uniform errors like using `nnf` as unary, assuming `nnf` negates, or using `isForm` as a return statement

---

2 Search

Problem 2.1 (DFS and BFS Concretely) 8 pt

Consider the **infinite** tree whose nodes are the natural numbers with root 0. For every node, the children and their order are as follows:

- children of 0: 1, 7, 11
- children of 1: 2, 5
- children of 2: 3
- children of 3: 4
- children of 5: 6
- children of 7: 8, 10
- children of 8: 9
- children of 11: 12
   ```

- children of n for $n \geq 12$: n+1
- other nodes have no children

1. List the nodes in the order of expansion during
   (a) depth-first search 3 pt
   (b) breadth-first search 3 pt
2. Assuming the goal state is 7, how does it matter whether we use depth-first or breadth-first search? 2 pt

---

Solution:

1. (a) $0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, \ldots$
   (b) $0, 1, 7, 11, 2, 5, 8, 10, 12, 3, 6, 9, 13, 4, 14, 15, \ldots$
2. Both algorithms will find the goal state. DFS takes longer because it traverses all descendants of 1 whereas BFS finds 7 right away.

Grading:

1. If generally correct, we deducted 0.5 or 1 for each mistake.
2. We gave 2 points for the correct answer, and 0 points otherwise.

Problem 2.2 (Heuristics) 5 pt

Consider A^* search with path cost function g and heuristic function h.

1. What is the effect of using the heuristic $h(n) = 0$? 3 pt
2. What is the effect of using the path cost as the heuristic, i.e., if we put $h(n) = g(n)$? 2 pt

---

Solution:

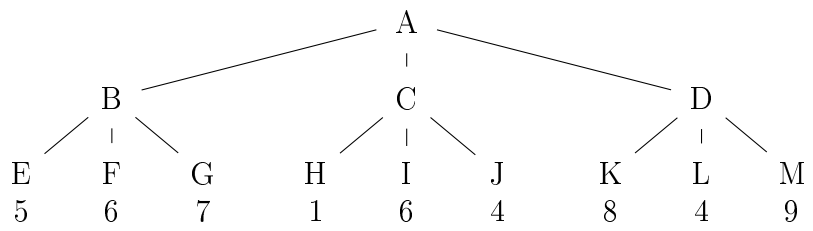
1. The algorithm will behave like uninformed search.
2. The algorithm will behave like uninformed search (because $2g(n)$ leads to the same expansion decisions as $g(n)$). Bonus point: the heuristic is not even admissible because we may have $g(n) > h^*(n)$.

---

1. deductions for mistakes
2. deductions for mistakes; 1 bonus point for the heuristics not being admissible (needed because the intended solution was incorrect)

Problem 2.3 (Adversarial Search) 8 pt

Consider the following minimax game tree for the **minimizing player**'s turn. The values at the leaves are the static evaluation function values of those states.



1. Label each non-leaf node with its minimax value. 4 pt
2. Which move would be selected by the player? 1 pt
3. List the nodes that the alpha-beta algorithm would prune (i.e., not visit). Assume children of a node are visited left-to-right. 3 pt

---

Solution:

1. A=6, B=7, C=6, D=9
2. Move C
3. L, M

Grading:

1. 1 point for each correct node
2. 1 point for the correct answer
3. 1 point for each correct node and 1 point for not having extraneous nodes

---

3 Constraint Satisfaction/Propagation

Problem 3.1 (4 Rooks on a Small Board) 7 pt

Consider the following problem: We want to place 4 rooks on a 3×3 chess-board such that no two rooks threaten each other. (Rooks move like queens except not diagonally.)

Model the problem as a constraint satisfaction problem (V, D, C).
Use your model to argue briefly but rigorously why this problem is unsatisfiable.

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: $V = a_1, a_2, b_1, b_2, c_1, c_2, d_1, d_2$
$D_x = \{1, 2, 3\}$ for all $x \in V$ Constraints in C:

- $v_1 \neq w_1$ for all $(v, w) \in V^2$ with $v \neq w$
- $v_2 \neq w_2$ for all $(v, w) \in V^2$ with $v \neq w$

The constraints require that all of a_1, b_1, c_1, d_1 are all different. But their joint domain only has 3 different values.

Grading: 2 points each for the variables, domains, and constraints, and 1 point for the argument. Deductions for mistakes.

Problem 3.2 (CSP Formalization) 8 pt

Consider the following binary CSP:

- $V = \{x, y, z\}$
- $D_x = \{0, 1, 2\}$, $D_y = \{1, 2\}$, $D_z = \{0, 1\}$
- Constraints: $x < y$, $y \neq z$, $x > z$

1. Give all pairs (v, w) of variables such that v is arc-consistent relative to w. 3 pt
2. Give all solutions. 2 pt
3. What is special about the constraint $y \neq z$? 1 pt
4. Assume we assign $y = 1$ and apply forward-checking. Give the resulting domains D_x, D_y, D_z. 2 pt

---

Solution:

- (y, x), (z, x), (y, z), (z, y)
- Solutions: $(x, y, z) = (1, 2, 0)$
- The constraint can be dropped because it is satisfied automatically if the other two constraints are.
- $D_x = \{0\}$, $D_y = \{1\}$, $D_z = \{0\}$

Grading:

1. 0.5 points for the correctly classification of each pair.
2. 2 points the correct set of solutions.
3. no partial credit
4. 1 point each for the two non-trivial domains

---

4 Logic

Problem 4.1 (Satisfiability and Validity) 6 pt

Consider propositional logic with propositional variables $\{P, Q, R\}$. For each of the following statements, give a counter-example that refutes it:

1. The formula $\big((P \wedge Q) \vee (Q \wedge R)\big) \Rightarrow (\neg P \vee \neg R)$ is satisfied by all assignments. 2 pt
2. If a formula F cannot be proved in the natural deduction calculus, then $\neg F$ is valid. 2 pt
3. If, for two formulas F, G, all assignments satisfy $F \Rightarrow G$ and no assignment satisfies F, then no assignment satisfies G. 2 pt

---

Solution:

1. A falsifying assignment is $P = Q = R = 1$
2. P cannot be proved but is satisfiable.
3. Let $F = P \wedge \neg P$ and $G = P$. Then every assignments satisfies $F \Rightarrow G$, none satisfies F, and G is satisfiable.

Grading: deductions for mistakes

Problem 4.2 (Proving in Natural Deduction) 9 pt

Prove the following formula using natural deduction:

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

Solution:

$$\dfrac{\dfrac{\dfrac{\dfrac{}{\forall x.P(x), \forall y.P(y) \Rightarrow Q(y) \vdash \forall x.P(x)}Ax}{\forall x.P(x), \forall y.P(y) \Rightarrow Q(y) \vdash P(z)}\forall E(z) \quad \dfrac{\dfrac{}{\forall x.P(x), \forall y.P(y) \Rightarrow Q(y) \vdash \forall y.P(y) \Rightarrow Q(y)}Ax}{\forall x.P(x), \forall y.P(y) \Rightarrow Q(y) \vdash P(z) \Rightarrow Q(z)}\forall E(z)}{\forall x.P(x), \forall y.P(y) \Rightarrow Q(y) \vdash Q(z)} \Rightarrow E}{\forall x.P(x), \forall y.P(y) \Rightarrow Q(y) \vdash \forall z.Q(z)} \forall I}{\forall x.P(x) \vdash (\forall y.P(y) \Rightarrow Q(y)) \Rightarrow \forall z.Q(z)} \Rightarrow I$$

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

Grading: $7-9$ points if the structure of implication/forall introduction/elimination was present. $4-6$ points if there was some resemblance to it. $1-3$ points for other attempts.

5 Knowledge Representation

Problem 5.1 (Specifying Properties in ALC) 9 pt

Consider the following ALC setting:

- concepts: `animal`, `plant`
- relations: `eats`

We abbreviate every concept/relation by its first letter.

Note: For the purposes of this question, carnivores are the animals that eat other animals, and herbivores are the animals that only eat plants.

---

1. Give ALC expressions for
   (a) the concept of all herbivores — 3 pt
   (b) the statement that carnivores do not eat each other — 3 pt
2. Assume a domain $\mathcal{D}$ and interpretations $A \subseteq \mathcal{D}$, $P \subseteq \mathcal{D}$, and $E \subseteq \mathcal{D} \times \mathcal{D}$ of a, p, and e, respectively. Give the semantics of the formula $\forall e.\big((\exists e.a) \sqcap (\exists e.p)\big)$. — 3 pt

---

Solution:

1. (a) $a \sqcap \forall e.p$
   (b) $c \sqcap \exists e.c \equiv \bot$ where c abbreviates $a \sqcap \exists e.a$
2. $\{u \in \mathcal{D} \,|\, \text{for all } v \in \mathcal{D} \text{ with } (u,v) \in E, \text{ there are } x, y \in \mathcal{D} \text{ such that } (v,x) \in E, x \in A, (v,y) \in E, y \in P\}$

Grading: In each case, 2 points if mistakes, 1 point if some resemblance to the correct solution.

Problem 5.2 (Extending ALC) 8 pt

Consider ALC concepts as given by the grammar

$$C ::= a \mid \top \mid \bot \mid \overline{C} \mid C \sqcap C \mid C \sqcup C \mid \exists R.C \mid \forall R.C$$

and with the

- semantics that maps every concept C to $[\![C]\!] \subseteq \mathcal{D}$,
- translation to first-order logic with equality that translates every concept C to a formula $\overline{C}^{fo(x)}$ with free variable x,

both defined by induction on concepts.

We want to extend ALC with the following two concept constructors:

- $C - D$ for the C's that are not D's
- $\exists^{!} R.C$ for objects that are connected via role R to *at most* one C

Give the cases that we must add to grammar, semantics, and translation to obtain that extension.

Solution: We extend the

- syntax with productions $C ::= C - C \mid \exists^! R.C$,
- semantics with cases $[\![C - D]\!] = [\![C]\!] \setminus [\![D]\!]$ and $[\![\exists^! R.C]\!] = \{x \in \mathcal{D} \mid$ there is at most one $v \in [\![C]\!]$ with $(u, v) \in [\![R]\!]\}$,
- translation with cases $\overline{C - D}^{fo(x)} = \overline{C}^{fo(x)} \wedge \neg \overline{D}^{fo(x)}$ and $\overline{\exists^! R.C}^{fo(x)} = \forall y, y'.\big(R(x, y) \wedge R(x, y') \wedge \overline{C}^{fo(y)} \wedge \overline{C}^{fo(y')}\big) \Rightarrow y = y'$.

Grading: 4 points for each operator; 2 points for the production, 1 each for the cases

6 Planning

Problem 6.1 (Planning Deliveries in STRIPS) 12 pt

Consider a truck that can carry 4 objects at a time and is supposed to deliver objects $Obj = \{V, W, X, Y, Z\}$ from location A to certain locations $Loc = \{A, B, C, D\}$ along some roads $Roads = \{\{A, B\}, \{B, C\}, \{B, D\}\}$. We use the following STRIPS task:

- facts: $\{\texttt{at}(l, o) \mid l \in Loc, o \in Obj\} \cup \{\texttt{truck}(l) \mid l \in Loc\}$
- actions $\texttt{move}(l, m, O)$ for $\{l, m\} \in Roads$, $O \subseteq Obj$, $|O| \leq 4$ given by
  - precondition: $\texttt{at}(l, o)$ for all $o \in O$, $\texttt{truck}(l)$
  - add list: $\texttt{at}(m, o)$ for all $o \in O$, $\texttt{truck}(m)$
  - delete list: same as precondition
- initial state: $\texttt{truck}(C)$, $\texttt{at}(A, o)$ for $o \in Obj$
- goal state: $\texttt{at}(C, V)$, $\texttt{at}(D, W)$, $\texttt{at}(B, X)$, $\texttt{at}(B, Y)$, $\texttt{at}(A, Z)$

1. Give a sequence of two actions that is applicable in the initial state and give the resulting state. 4 pt
2. Give an optimal plan for the task above. 4 pt
3. Consider the following heuristics: $h(s) = \frac{1}{4} \sum_{o \in Obj} d(s, o)$ where $d(s, o)$ is the number of roads separating the location of o in state s from its location in the goal state. Argue whether this heuristic is admissible or not. 4 pt

Solution:

1. $\texttt{move}(C, B, \emptyset), \texttt{move}(B, Q, \emptyset)$ where $Q \in \{A, C, D\}$ results in $\texttt{truck}(\texttt{Q})$, $\texttt{at}(A, o)$ for $o \in Obj$
2. $\texttt{move}(C, B, \emptyset), \texttt{move}(B, A, \emptyset), \texttt{move}(A, B, \{V, W, X, Y\}), \texttt{move}(B, C, \{V\}), \texttt{move}(C, B, \emptyset), \texttt{move}(B, D, \{W\})$
3. The heuristic is admissible. An optimal must make at least $d^s(o)$ steps to move object o to its goal state, so $\sum_{o \in Obj} d^s(o)$ steps in total. At best, it can carry 4 objects in each step. So $h(s)$ is a smaller than the minimal number of steps in any optimal plan (which is the definition of admissible).

Grading:

1. 2 points for the actions and 2 points for the successor state; deductions for mistakes
2. deductions for mistakes
3. 1 point for the answer, 3 points for the argument
