Last Name: First Name:

Matriculation Number:

Seat:

Exam
Artificial Intelligence 1

Feb 13, 2023

	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	Sum	grade
total	8	8	4	7	6	6	8	6	8	6	8	10	85	
reached														

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 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 (First-Order Terms in Prolog) 8 pt

Consider the following Prolog program:

```
isSignature([]).                                                    1
isSignature([funSym(Name,Arity)|Tail]) :-                           2
  string(Name), integer(Arity), Arity >= 0, isSignature(Tail).      3
                                                                    4
isTerm(_, Vars, var(Name)) :-                                       5
  contains(Vars, Name).                                             6
isTerm(Sig, Vars, applyFunSym(Name,Args)) :-                        7
  contains(Sig,funSym(Name,L)),                                     8
  len(Args,L), isTermList(Sig,Vars,Args).                           9
                                                                    10
isTermList(_, _, []).                                               11
isTermList(Sig, Vars, [T|Ts]) :-                                    12
  isTerm(Sig, Vars, T), isTermList(Sig, Vars, Ts).                  13
                                                                    14
% computes the length of a list                                     15
len([],0).                                                          16
len([_|T],M) :- len(T,L), M is L+1.                                 17
                                                                    18
% checks if a list contains a certain value                         19
contains([H|_],H).                                                  20
contains([_|T],X) :- contains(T,X).                                21
```

Here `string` and `integer` are built-in predicates that succeed if their argument is a string or an integer, respectively.

1. What does the predicate `isTerm(S,V,T)` compute? 3 pt
2. What do the following queries return: 3 pt
   (a) `isSignature([funSym("G",1)])`
   (b) `isTerm([funSym("f",1)], ["X"], X)`

   If a query returns multiple results, only give the first two.
3. Now assume we change line 9 by switching the order of the two predicates, i.e., `isTermList(Sig,Vars,Args), len(Args,L).`
   Under what circumstances and how would Prolog's search behavior change the run time of `isTerm`? 2 pt

Solution:

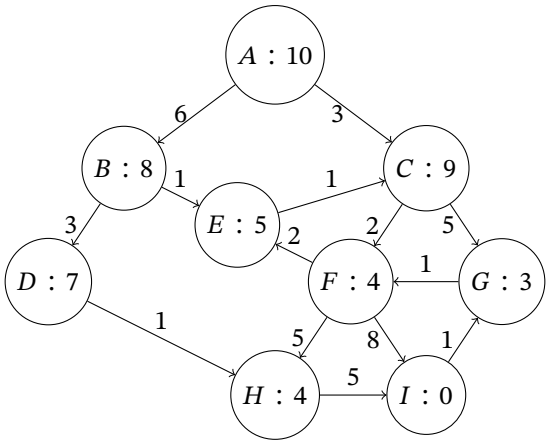
1. It checks if `T` is a term over the signature `S` using free variables from `V`.
2. (a) `true`

   (b) `X = var("X"), X = applyFunSym("f",[var("X")])`
3. The checks are done in left-to-right order, and the second check is only attempted if the first on succeeded. If both checks succeed, the run-time is unaffected. But for ill-formed input, the run-time will be higher or lower.

2 Search

Problem 2.1 (Search Algorithms) 8 pt

Consider the following 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. 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

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, break it using alphabetical order.

Solution:

1. B, D, H, I
2. B, C, D, E
3. C, F, B, E
4. B, E, D, H
5. C, F, G, E

Problem 2.2 (Complexity) 4 pt

Consider trees with maximal branching factor b, maximal depth m, and minimal depth d of a solution.
Give the worst-case complexity classes in b, d, and m (using O-notation) of

1. time complexity of breadth-first search 1 pt
2. space complexity of breadth-first search 1 pt
3. time complexity of depth-first search 1 pt
4. space complexity of depth-first search 1 pt

Solution:

1. $O(b^d)$
2. $O(b^d)$
3. $O(b^m)$
4. $O(bm)$

Problem 2.3 (Search Problems) 7 pt

Consider the search problem $\langle S, A, T, I, G \rangle$ where

- ☐ $S = \mathbb{Z} \times \mathbb{Z}$
- ☐ $A = \{X, Y, M\}$
- ☐ T is given by
  - ☐ $T(X, (x, y)) = \{(-x, y)\}$
  - ☐ $T(Y, (x, y)) = \{(x, -y)\}$

☐ $T(M,(x,y)) = \{(x+1,y),(x,y+1)\}$

☐ $I = \{(0,0)\}$

☐ $G = \{(x,y) \in S \mid x^2 + y = 8\}$

1. Check the box of the row that makes this problem non-deterministic. 1 pt
2. Give the possible states resulting from applying the action sequence M, X, Y to the initial state. 2 pt
3. Give a solution of minimal length to this problem. 3 pt
4. Assume we have added an action D to A, and we want the successor state after applying D to be the one where both coordinates are swapped. Give the necessary extension of the transition model. 1 pt

---

Solution:

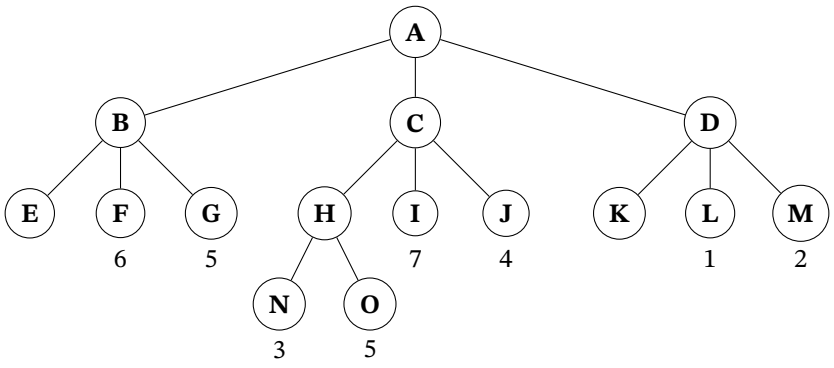
1. The case for $T(M,(x,y))$.
2. $(-1,0),(0,-1)$
3. The minimal solutions have length 5 and must use M, M, M, M, Y in any order that does not start with Y.
4. $T(D,(x,y)) = \{(y,x)\}$

---

3 Adversarial Search

Problem 3.1 (Minimax) 6 pt

Consider the following minimax game tree for the **minimizing** 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. If possible, label the node **K** with an evaluation function value that results in the player definitely choosing move **D** (i.e., no matter how **E** is labeled or how ties are broken). 2 pt
   Otherwise, argue why that is impossible.
3. If possible, label the node **E** with an evaluation function value that results in the player definitely choosing move **D** (i.e., no matter how **K** is labeled or how ties are broken). 2 pt
   Otherwise, argue why that is impossible.

Solution:

1. C= 7, H= 3
2. K$\leq$ 5.
3. This is impossible. We can always label $K > \max\{E, F, G\}$, in which case the player will prefer move B to move D.

---

4 Constraint Satisfaction/Propagation

Problem 4.1 (Modeling) 6 pt

You want to place 4 items into 3 boxes. The sizes of the items are S_1, S_2, S_3, S_4. The capacities of the boxes are C_1, C_2, C_3. The total of the sizes of the items in each box must not exceed its capacity.

Model this problem as a (not necessarily binary) constraint satisfaction problem $\langle V, D, C\rangle$. Briefly explain how the solutions correspond to the placement of the items.

Solution: E.g.,
$V = \{b_1, b_2, b_3, b_4\}$
$D_{b_i} = \{1, 2, 3\}$ for all i
Constraints in C: For every $j \in \{1, 2, 3\}$, one constraint $\Sigma_{i=1,\ldots,4; b_i=j} S_i \leq C_j$

Explanation: If $b_i = j$, then item i is placed in box j.

Problem 4.2 (Solving) 8 pt

Consider the following binary CSP:

- $V = \{a, b, c, d, e, f\}$

- $D_a = D_b = D_c = \{0, 1, 2, 3\}$, $D_d = D_e = D_f = \{0, 1, 2, 3, 4, 5, 6\}$
- Constraints:
  - $a < 2$ or $d > 5$
  - $b \cdot e > 10$
  - $e < f$
  - $d = 2c$
  - $f^2 - b^2 > 17$
  - $a > c$

1. Check the boxes for (v, w) where v is **not** arc-consistent relative to w. 2 pt

   ☐ (b, c) ☐ (c, b) ☐ (b, f) ☐ (f, b) ☐ (c, d) ☐ (d, c)

2. Give an **inconsistent** total assignment to the variables. 1 pt
3. Give **the two** solutions. 3 pt
4. Now assume we add the constraint $f = a + e$.
   (a) What distinguishes this constraint crucially from the other constraints? 1 pt
   (b) How does this difference affect constraint propagation? 1 pt

---

Solution:

1. (d, c), (f, b)
2. Any total assignment that is not a solution, e.g., $a = b = c = d = e = f = 0$.
3. The problem can be decomposed into two independent problems:
   - For (a, c, d), the only solution is $(1, 0, 0)$.
   - For (b, e, f), the only solutions are $(3, 4, 6)$ and $(3, 5, 6)$.

   The overall solutions (a, b, c, d, e, f) are the two combinations $(1, 3, 0, 0, 4, 6)$ and $(1, 3, 0, 0, 5, 6)$.
4. (a) It is a higher-order constraint.
   (b) Techniques for binary CSPs such as forward checking or arc-consistency do not apply (or are more complicated).

---

5 Logic

Problem 5.1 (Propositional Logic) 6 pt

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

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

1. Give the interpretation $\mathcal{I}_\varphi(A)$ for every possible assignment φ, e.g., via a truth table. 2 pt
2. Give some formula in CNF that is equivalent to $\neg A$. 2 pt
3. Assume we have constructed a saturated tableau with root A^F that has a single open branch. What useful information about A can we read off that open branch? 2 pt

---

Solution:

1. There are four assignments:

$\varphi(P)$	$\varphi(Q)$	$\mathcal{I}_\varphi(A)$
0	0	1
0	1	1
1	0	1
1	1	0

   Note that the left side of the implication is always true and A is equivalent to $\neg(P \wedge Q)$.

2. E.g., $P \wedge Q$.
3. The literals on the branch determine an assignment that falsifies A. In particular, A is not a theorem.

Problem 5.2 (Predicate Logic) 8 pt

Consider the following signature of predicate logic:

- unary function symbol f
- binary function symbol g
- binary predicate symbol p

1. Give a formula that uses all three symbols and is a theorem. 2 pt

2. Now consider the formulas A and B given by

$$A \;=\; \forall x.\forall y.p(f(x), g(x,y))$$

$$B \;=\; \forall x.\forall y.p(x,y)$$

Give a model $\langle D, \mathcal{I}\rangle$ in which A is true but B is false. 3 pt

3. Prove or refute the following statement: If the formula A has one free variable x, and there is a model in which A is true under all assignments, then A is a theorem. 3 pt

---

Solution:

1. E.g., $T \vee A$ where A is as above.
2. E.g., $D = \{0,1\}$, $\mathcal{I}(f)(x) = 0$, $\mathcal{I}(g)(x,y) = 0$, $\mathcal{I}(p) = \{(0,0)\}$.
3. False. A counter-example is given by $G \;=\; p(x,x)$ and any model $\langle D, \mathcal{I}\rangle$ in which $\mathcal{I}(p) = D \times D$.

Problem 5.3 (Proving in Tableau Calculus) 6 pt

We use the propositional variables P, Q, and R and define formulas A, B, and C by

$$A \;=\; Q \wedge (Q \Rightarrow R)$$

$$B \;=\; P \Rightarrow A$$

$$C \;=\; P \Rightarrow R$$

Prove the formula $B \Rightarrow C$ using the tableaux calculus.

Solution:

$$\frac{B \Rightarrow C^F}{\begin{array}{c} B^T \\ C^F \\ P^T \\ R^F \\ \hline \begin{array}{c|c} P^F & A^T \\ \text{close on } P & Q^T \\ & Q \Rightarrow R^T \\ & \begin{array}{c|c} Q^F & R^T \\ \text{close on } Q & \text{close on } R \end{array} \end{array} \end{array}}$$

$\perp$ for closing is acceptable.

---

6 Knowledge Representation

Problem 6.1 (Description Logic) 8 pt

Consider the following $\mathcal{ALC}$-setting:

- concepts: `student`, `instructor`, `course`, `person`
- relations: `takes`, `gives`

We abbreviate every concept/relation by its first letter.

1. Give an $\mathcal{ALC}$-ABox using individuals *Alice*, *Bob*, and $Math$ such that *Alice* is a student taking the $Math$-course given by the instructor *Bob*. 2 pt
   You can abbreviate every individual by their first letter.
2. Give an $\mathcal{ALC}$-TBox with two concept axioms expressing the following: 2 pt
   (a) Every person is an instructor or a student.
   (b) Courses are only given by instructors.
3. Consider the model $\langle D, \mathcal{I} \rangle$ given by: 2 pt
   - $D = \{MK, Carol, Donald, AI1, AI2\}$
   - $\mathcal{I}(\mathtt{s}) = \{Carol, Donald\}$
   - $\mathcal{I}(\mathtt{i}) = \{MK\}$
   - $\mathcal{I}(\mathtt{c}) = \{AI1, AI2\}$
   - $\mathcal{I}(\mathtt{p}) = \{MK, Carol, Donald\}$
   - $\mathcal{I}(\mathtt{t}) = \{(Carol, AI1), (Donald, AI2)\}$
   - $\mathcal{I}(\mathtt{g}) = \{(MK, AI1), (MK, AI2)\}$

   We define the concept G by

   $$G \;=\; (\exists \mathtt{t}.\mathtt{c}) \sqcup (\exists \mathtt{g}.\mathtt{c})$$

   Give the interpretation $\mathcal{I}(G)$.
4. Give the result of the first-order translation $\overline{G \sqsubseteq \mathtt{p}}^{fo(x)}$ (where G is as in the previous question). 2 pt

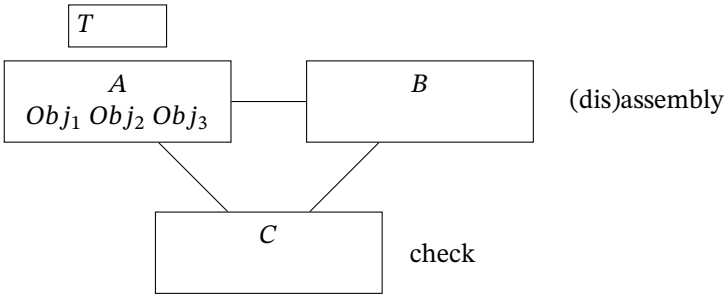
Solution:

1. $A : s, B : i, M : c, B\,g\,M, A\,t\,M$
2. Axioms
   (a) $p \sqsubseteq s \sqcup i$
   (b) $\exists g.c \sqsubseteq i$
3. $\{MK, Carol, Donald\}$
4. $\forall x.\big((\exists y.t(x,y) \wedge c(y)) \vee (\exists y.g(x,y) \wedge c(y))\big) \Rightarrow p(x)$

7 Planning

Problem 7.1 (STRIPS) 10 pt

Consider a machine that processes objects $Obj = \{Obj_1, Obj_2, Obj_3\}$, which can be at location A, B, or C. Currently all objects are at location A and **unchecked** and **assembled**. Eventually all objects are needed in location A and **checked** and **assembled**.
At location B, objects can be assembled or disassembled. At location C, disassembled objects can be checked.
A transporter T is available (currently at location A) that can move **one** object at a time from one location to any other location.



We formalize this problem as a STRIPS task where the **facts** are

- $\texttt{at}(l, o)$ for $l \in \{A, B, C\}$ and $o \in Obj \cup \{T\}$
- $\texttt{isCh}(o)$ and $\texttt{isAss}(o)$ and $\texttt{isDis}(o)$ for $o \in Obj$

and the **actions** are given by table below for any $l, m \in \{A, B, C\}$ and $o \in Obj$

action	precondition	add list	delete list
move(l, m, o)	at(l, o)	at(m, o)	at(l, o)
assemble(o)	at(B, o)	isAss(o)	isDis(o)
disassemble(o)	at(B, o)	isDis(o)	isAss(o)
check(o)			

1. Complete the definition of the action check(o). 2 pt
2. Give the initial state I and the goal G. 2 pt
3. Give the value $h^*(I)$. 2 pt
4. Give the value $h^+(I)$. 2 pt
5. Prove or refute that the following heuristic is admissible: $h(s)$ is 5 times the number of unchecked objects in state s. 2 pt

---

Solution:

1.
   - precondition: at(C, o), isDis(o)
   - add list: isCh(o)
   - delete list: nothing
2. initial state: at(A, o) and isAss(o) for all $o \in Obj$, and at(A, T),
   goal: at(A, o), isCh(o) and isAss(o) for all $o \in Obj$
3. 21. An optimal plan requires 7 actions per object: move(A, B, o), disassemble(o), move(B, C, o), check(o), move(C, B, o), assemble(o), move(B, A, o)
4. 12. With the delete heuristic, objects do not have to re-assembled or moved back. So only 4 actions are needed per object.
5. It is not admissible. A counter-example is the state s with T and one disassembled, unchecked object at location C and all other objects checked and assembled at A. Then $h^*(s) = 4 < 5 = h(s)$ in violation of the admissibility condition.
