Last Name: First Name:

Matriculation Number:

Seat:

Exam
Artificial Intelligence 1

Feb 10., 2020

	To be used for grading, do not write here					
prob.	1	2	3	Sum	grade	
total	18	5	3	92		
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 (Girls are witches) 6 pt

6 min

Consider the following logic argument (after Monty Python):

- A witch is a female who burns.
- Things burn - because they're made of wood.
- Wood floats.
- What else floats on water? A duck.
- If something has the same weight as a duck it must float.
- A duck and scales are fetched. A girl and the duck balance perfectly.

1. Write the given statements as a ProLog knowledge base.
2. How would you check whether – according to this knowledge base – a particular girl Mary is a witch?
3. Justify whether Mary is a witch – according to your knowledge base.

Solution:

1.
```
witch(X):-burns(X),female(X).
burns(X):-wooden(X).
floats(X):-wooden(X).
floats(duck).
floats(X):-sameweight(duck,X).
female(girl).
sameweight(duck,girl).
```

2. Using the ProLog query `?- witch(girl).`
3. Of course Mary is not a witch, ProLog cannot derive that $Mary$ can burn, which is needed for being a witch.

Problem 2 (Prolog Fibbonaci) 6 pt

6 min

Write a ProLog program which computes the n-th *Fibonacci number*.

Solution:

```
/* Top down */
fibonacci(0, 0).
fibonacci(1, 1).
fibonacci(N, X) :- N>1, N1 is N - 1, N2 is N - 2, fibonacci(N1, Y), fibonacci(N2, Z), X is Y

/* A more efficient, bottom-up approach. */
```

```
fibo(N,F):-fibo1(0,0,1,N,F).
fibo1(N,F,_,N,F).
fibo1(N1,F1,F2,N,F):- N1<N, N2 is N1+1, F3 is F1+F2, fibo1(N2,F2,F3,N,F).
```

2 Search

Problem 1 2 pt

Does a finite state space always lead to a finite search tree? How about a finite space state that is a tree? Justify your answers. 2 min

Solution: No (there can be cycles). Yes if it's a tree (no cycles).

Problem 2 (Relaxed Problem) 8 pt

5 min

The relaxed version of a search problem P is a problem P' with the same states as P, such that any solution of P is also a solution of P'. More precisely, if s' is a successor of s in P, it is also a successor in P' with the same cost. Prove or refute that for any state s, the cost $c'(s)$ of the optimal path between s and the goal in P' is an admissible A^* heuristic for P.

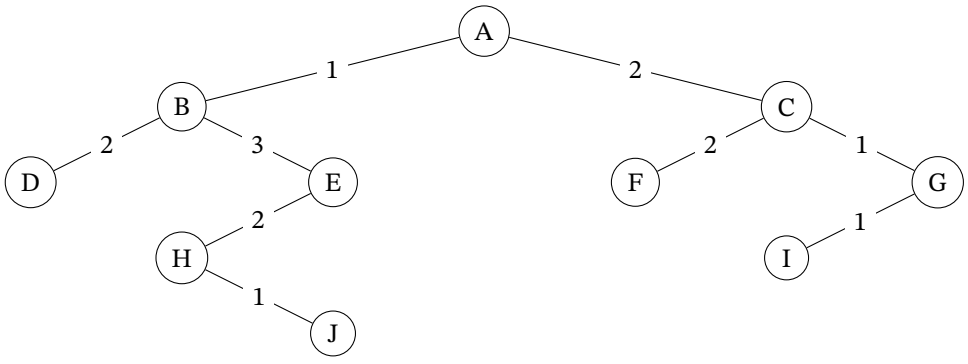
Hint: Think about the graphical representation of the problems.

Solution: Graphically, P' has all the arcs from P, plus maybe more. This means that the optimal path in P' is the same as the optimal path in P, or better through the possibility of using the additional arcs. Therefore $c'(s) \leq c(s)$, which is the admissibility criterion for a heuristic.

Problem 3 (Uninformed search) 4 pt

4 min

Apply uniform cost search and iterative deepening on the following tree exhaustively (the goal is not node G!).



List the nodes in the order they are expanded for uniform cost search and iterative deepening. For iterative deepening, write out each iteration in a new line. For uniform cost, assume that when nodes have the same cost, they get expanded left to right.

1. Iterative deepening:
2. Uniform cost:

Solution:

Uniform cost:

A, B, C, D, G, E, F, I, H, J

Iterative deepening:

A

A, B, C

A, B, D, E, C, F, G

A, B, D, E, H, C, F, G, I

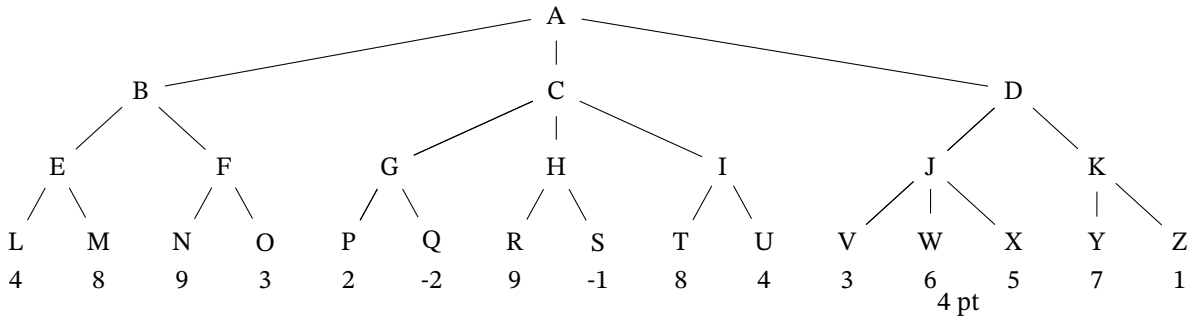
A, B, D, E, H, J, C, F, G, I

---

3 Adversarial Search

Problem 1 (Game Tree) 10 pt

Consider the following game tree. Assume it is the maximizing player's turn to move. The values at the leaves are the static evaluation function values of the states at each of those nodes. 10 min



4 pt

1. Label each non-leaf node with its minimax value. 1 pt
2. Which move would be selected by Max? 3 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. 2 pt
4. In general (i.e., not just for the tree shown above), if we traverse a game tree by visiting children in right-to-left order instead of left-to-right, can this result in a change to

(a) the minimax value computed at the root?

(b) The number of nodes pruned by the alpha-beta algorithm?

Solution:

1. A:8, B:8, C:2, D:6, E:8, F:9, G:2, H:9, I:8, J:6, K:7
2. B
3. OHRSITUKYZ
4. (a) no, (b) yes

---

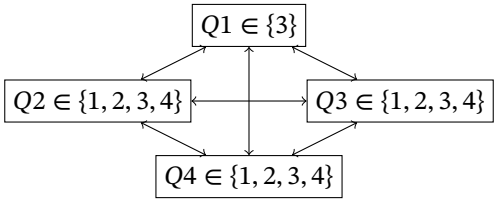
4 Constraint Satisfaction Problems & Inference

Problem 1 16 pt

16 min

Consider solving the 4-queens problem as a constraint satisfaction problem. That is, place 4 queens on a 4×4 board such that no queen is in the same row, column or diagonal as any other queen.

One way to formulate this problem is to have a variable for each queen, and binary constraints between each pair of queens indicating that they cannot be in the same row, column or diagonal. Assuming the i-th queen is put somewhere in the i-th column, then the possible values in the domain for each variable are the row numbers in which it could be placed. Say we initially assign queen $Q1$ the unique value 3, meaning $Q1$ is placed in column 1 and row 3. This results in an initial constraint graph given by the set of candidate values of each variable is shown inside the node:



4 pt

1. Apply forward checking and give the remaining candidate values for the variables $Q2$, $Q3$ and $Q4$.

   4 pt

2. Define the concept of *arc consistency*.

   6 pt

3. Fill in the table below with the candidate values of each queen after each of the following steps of applying the arc consistency algorithm to the figure.

	Q1	Q2	Q3	Q4
Initial domain	3	1, 2, 3, 4	1, 2, 3, 4	1, 2, 3, 4
After $Q2 \to Q1$	3			
After $Q3 \to Q1$	3			
After $Q2 \to Q3$	3			
After $Q3 \to Q2$	3			

2 pt

4. In general, when will the arc consistency algorithm halt?

Solution:

1. $Q_2 = \{1\}$, $Q_3 = \{2, 4\}$, $Q_4 = \{1, 2, 4\}$
2. v is arc-consistent wrt w if for every value of v there is a value for w such that C_{vw} is satisfied.
3. $Q2 \to Q1$: remove $2, 3, 4$ from D_2; $Q3 \to Q1$: remove $1, 3$ from D_3; $Q2 \to Q3$: no change; $Q3 \to Q2$: remove 2 from Q_3
4. When no constraint changes the domains anymore.

---

5 Logic

Problem 1 (Natural Deduction) 7 pt
Prove (or disprove) the validity of the following formulae in Natural Deduction: 7 min

1. $((P \Rightarrow Q) \wedge P) \Rightarrow Q$ 2 pt
2. $(\neg Q \Rightarrow \neg P) \Rightarrow (P \Rightarrow \neg\neg Q)$ 5 pt

---

Solution:

1.

(1)	1	$(P \Rightarrow Q) \wedge P$	Assumption
(2)	1	$P \Rightarrow Q$	$\wedge E_\ell$ (on 1)
(3)	1	P	$\wedge E_r$ (on 1)
(4)	1	Q	$\Rightarrow E$ (on 2 and 3)
(5)		$((P \Rightarrow Q) \wedge P) \Rightarrow Q$	$\Rightarrow I$ (on 1 and 4)

2.

(1)	1	$\neg Q \Rightarrow \neg P$	Assumption
(2)	1, 2	P	Assumption
(3)	1, 2	$\neg Q$	Assumption
(4)	1, 2, 3	$\neg P$	$\Rightarrow E$ (on 1 and 3)
(5)	1, 2, 3	F	FI (on 2 and 4)
(6)	1, 2	$\neg\neg Q$	$\neg I$ (on 3, 5)
(7)	1	$P \Rightarrow \neg\neg Q$	$\Rightarrow I$ (on 2 and 6)
(8)		$(\neg Q \Rightarrow \neg P) \Rightarrow (P \Rightarrow \neg\neg Q)$	$\Rightarrow I$ (on 1 and 7)

Problem 2 (First-Order Tableau) 5 pt

5 min

Prove or refute the following formula using the first-order free variable tableaux calculus. We have $P, R \in \Sigma_1^p$ and $f \in \Sigma_1^f$.

$$(\forall X.P(X) \Rightarrow R(f(X))) \Rightarrow ((\exists X.P(X)) \Rightarrow (\exists.R(Y)))$$

Solution:

$$\begin{array}{c} (((\forall X.P(X) \Rightarrow R(f(X))) \Rightarrow (\exists.P(X))) \Rightarrow (\exists.R(Y)))^{\mathsf{F}} \\ \forall X.P(X) \Rightarrow R(f(X))^{\mathsf{T}} \\ \exists X.P(X) \Rightarrow \exists Y.R(Y)^{\mathsf{F}} \\ \exists X.P(X)^{\mathsf{T}} \\ \exists Y.R(Y)^{\mathsf{F}} \\ P(Z) \Rightarrow R(f(Z))^{\mathsf{T}} \\ P(s)^{\mathsf{T}} \\ R(W)^{\mathsf{F}} \\ \begin{array}{c|c} P(Z)^{\mathsf{F}} & R(f(s))^{\mathsf{T}} \\ \bot[s/Z] & \bot[f(s)/W] \end{array} \end{array}$$

Problem 3 (Unification) 3 pt

3 min

Give a most general unifier of the terms $\mathbf{A} = f(X, g(Y, X))$ and $\mathbf{B} = f(Y, Z)$. Give one more unifier that is NOT most general and justify why it is not.

Solution:

- $[Y/X], [g(Y,Y)/Z] = \sigma$
- $[a/X], [a/Y], [g(a,a)/Z] = [a/Y] \circ \sigma$

---

6 Knowledge Representation

Problem 1 (Tableau-Calculus for ALC) 7 pt

7 min

Prove that the following concept is inconsistent using $\mathcal{T}_{\mathcal{ALC}}$:

$$(\forall \mathsf{takes}.\mathsf{GLOIN} \sqcup \forall \mathsf{takes}.\mathsf{AI1}) \sqcap \exists \mathsf{takes}.(\overline{\mathsf{AI1}} \sqcap \overline{\mathsf{GLOIN}})$$

Remember that in ALC quantifiers bind tightly, so $\forall \mathsf{takes}.\mathsf{GLOIN} \sqcup \forall \mathsf{takes}.\mathsf{AI1}$ means $(\forall \mathsf{takes}.\mathsf{GLOIN}) \sqcup (\forall \mathsf{takes}.\mathsf{AI1})$.

Hint: Use the judgment $x : C$ in $\mathcal{T}_{\mathcal{ALC}}$, where C is the concept above.

Solution: We use the $\mathcal{ALC}$ tableau calculus $\mathcal{T}_{\mathcal{ALC}}$ for consistency:

$$\begin{array}{c}
x : (\forall \mathsf{takes}.\mathsf{GLOIN} \sqcup \forall \mathsf{takes}.\mathsf{AI1}) \sqcap \exists \mathsf{takes}.(\overline{\mathsf{AI1}} \sqcap \overline{\mathsf{GLOIN}}) \\
x : (\forall \mathsf{takes}.\mathsf{GLOIN} \sqcup \forall \mathsf{takes}.\mathsf{AI1}) \\
x : \exists \mathsf{takes}.(\overline{\mathsf{AI1}} \sqcap \overline{\mathsf{GLOIN}}) \\
x \ \mathsf{takes} \ y \\
y : \overline{\mathsf{AI1}} \sqcap \overline{\mathsf{GLOIN}} \\
y : \overline{\mathsf{AI1}} \\
y : \overline{\mathsf{GLOIN}} \\
\begin{array}{c|c}
x : \forall \mathsf{takes}.\mathsf{GLOIN} & x : \forall \mathsf{takes}.\mathsf{AI1} \\
y : \mathsf{GLOIN} & y : \mathsf{AI1} \\
\bot & \bot
\end{array}
\end{array}$$

7 Planning

Problem 1 18 pt

18 min

Consider the following problem. A fused bulb hangs out of reach from the ceiling. A robot needs to repair the bulb. The room also contains a box. Pushing that box into the correct position, and climbing onto the box, will bring the bulb into reach for the robot.

The exercise is to model this problem as a STRIPS planning task. In doing so, assume the following framework. The robot is currently at position "A", the fused bulb is at position "B", and the box is at position "C". The robot and box are at the same height "Low", the fused bulb is at height "High". By climbing onto the box, the robot changes from "Low" to "High"; vice versa when climbing off the box. The actions available for the robot are "*Go*" from one place to another (only possible if the robot is at "Low"), "*Push*" an object from one place to another (only possible if the robot and object are at "Low"), "$ClimbUp$" onto or "*ClimbDown*" from an object, and "*Repair*" to fix an object. The robot needs to be at the same place and height as an object in order to repair it.

Note that the robot can only push an object or climb onto an object if both of them are at the same location. Furthermore, in case of pushing an object the robot changes to the destination location as well.

(a) Write a STRIPS formalization of the initial state and goal descriptions.

(b) Write a STRIPS formalization of the five actions: Go, $Push$, $ClimbUp$, $ClimbDown$, and $Repair$. In doing so, please do make use of "object variables", i.e., write the actions up in a parameterized way. State, for each parameter, by which objects it can be instantiated.

In both (a) and (b), make use of the following predicates: (do not use any other predicates)

- $At(x, y)$: To indicate that object $x \in \{Box, Bulb, Robot\}$ is at position $y \in \{A, B, C\}$.
- $Height(x, y)$: To indicate that object $x \in \{Box, Bulb, Robot\}$ is at height $y \in \{Low, High\}$.
- $Pushable(x)$: To indicate that object $x \in \{Box, Bulb, Robot\}$ can be pushed.
- $Climbable(x)$: To indicate that the robot can climb on object $x \in \{Box, Bulb, Robot\}$.
- $Repaired(x)$: To indicate that object $x \in \{Box, Bulb, Robot\}$ is repaired.

Solution:

(a) Initial state description:

$$\begin{aligned} I \quad = \quad & \{At(Robot, A), At(Bulb, B), At(Box, C), \\ & Height(Robot, Low), Height(Box, Low), Height(Bulb, High), \\ & Pushable(Box), Climbable(Box)\} \end{aligned}$$

Goal description: $G = \{Repaired(Bulb)\}$

(b) Action descriptions:

- $Go(x, y) =$
  $$\begin{aligned} pre &: \{At(Robot, x), Height(Robot, Low)\} \\ add &: \{At(Robot, y)\} \\ del &: \{At(Robot, x)\} \end{aligned}$$
  for all $x, y \in \{A, B, C\}$.
- $Push(x, y, z) =$
  $$\begin{aligned} pre &: \{At(Robot, y), Pushable(x), At(x, y), Height(Robot, Low), Height(x, Low)\} \\ add &: \{At(x, z), At(Robot, z)\} \\ del &: \{At(x, y), At(Robot, y)\} \end{aligned}$$
  for all $x \in \{Box, Bulb, Robot\}$, $y, z \in \{A, B, C\}$.

- $ClimbUp(x, y) =$

  $pre : \{At(Robot, y), At(x, y), Climbable(x), Height(Robot, Low), Height(x, Low)\}$
  $add : \{Height(Robot, High)\}$
  $del : \{Height(Robot, Low)\}$

  for all $x \in \{Box, Bulb, Robot\}$, $y \in \{A, B, C\}$.
- $ClimbDown() =$

  $pre : \{Height(Robot, High)\}$
  $add : \{Height(Robot, Low)\}$
  $del : \{Height(Robot, High)\}$

- $Repair(x, y, z) =$

  $pre : \{At(Robot, y), At(x, y), Height(Robot, z), Height(x, z)\}$
  $add : \{Repaired(x)\}$
  $del : \{\}$

  for all $x \in \{Box, Bulb, Robot\}$, $y \in \{A, B, C\}$, $z \in \{High, Low\}$.
