Last Name: First Name:

Matriculation Number:

Seat:

Retake Exam
Artificial Intelligence 1
—
with Solutions

August 10., 2020

	To be used for grading, do not write here													
prob.	1.1	1.2	2.1	2.2	2.3	3.1	4.1	4.2	5.1	5.2	6.1	7.1	Sum	grade
total	4	4	8	5	5	10	5	10	8	7	7	10	83	
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 (A PROLOG warm-up) 4 pt

Given as a ProLog fragment we have clauses for natural numbers.

```
nat(zero).
nat(s(X)):-nat(X).
```

Write unary predicates `even/1` and `odd/1` with the obvious meanings as well as a binary predicate `leq` for the "less or equal" relation on natural numbers.

Solution:

```
nat(zero).
nat(s(X)) :- nat(X).

even(zero).
even(s(X)) :- odd(X).
odd(s(X)) :- even(X).

leq(zero, X).
leq(s(X), s(Y)) :- leq(X,Y).
```

Problem 1.2 (Query for Ancestry) 4 pt

Write the following facts in ProLog, write a ProLog predicate `ancestor/2`, and query the database to find out whether Helen is Harry's ancestor. To write down the facts, only use the predicates `mother/2` and `father/2`.

- Helen is Saul's mother,
- Saul is James' father,
- James is Harry's father.

Hint: It may be a good idea to write a `parent/2` predicate. Only use it to write the predicate `ancestor/2`.

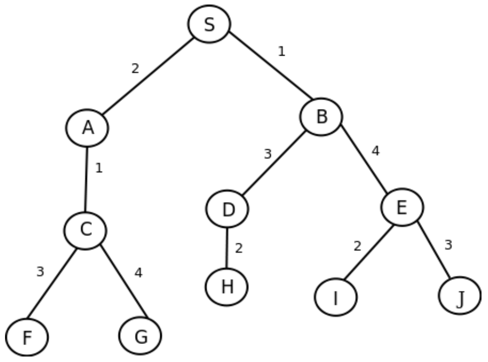
Solution:

```
mother(helen,saul).
father(saul,james).
father(james,harry).

parent(X,_) :- mother(X,_).
parent(X,_) :- father(X,_).

ancestor(X,Y):- parent(X,Y).
ancestor(X,Y):- parent(X,Z), ancestor(Z,Y).

?- ancestor(helen, harry).
```

2 Search

Problem 2.1 (Search in a graph) 8 pt

Look at the following graph and complete the tables below. S is the start state and G is the goal state, the step costs are given as labels on the edges.



In the table below, enter the labels of the nodes in the order they are visited by the respective search strategy. Remember that the search ends once the goal state is visited. If two nodes have equal chances to be visited, take the leftmost one first.

Search method	Sequence of nodes
BFS	
DFS	
Uniform cost	
Iterative deepening (step size 2)	

In the table below, complete the table with **Yes** or **No** for the respective search strategies and properties.

Property	BFS	DFS	Uniform Cost	Iterative Deepening
Optimal				
Complete				

Solution:

Search method	Sequence of nodes
BFS	S A B C D E F G
DFS	S A C F G
Uniform cost	S B A C D E F H G
Iterative deepening (step size 2)	S S A C B D E S A C F G

Property	BFS	DFS	Uniform Cost	Iterative Deepening
Optimal	N	N	Y	N
Complete	Y	N	Y	Y

For the purpose of this question, the requirements regarding completeness were considered reasonably satisfied in practice, yielding a Y. The caveats regarding optimality were considered as sometimes and thus not always satisfied, yielding an N.

Problem 2.2 (Admissibility limits) 5 pt

The condition for a heuristic $h(n)$ to be admissible is that for all nodes n holds that $(0 \leq h(n) \leq h^*(n))$, where $h^*(n)$ is the true cost from n to goal. What happens when for all nodes, $h(n) = 0$ and when $h(n) = h^*(n)$?

Solution: When $h(n) = 0$, the search will behave like an uninformed search, and when $h(n) = h^*(n)$ the search will only expand the nodes on the optimal path to a goal.

Problem 2.3 (Astar vs. Greedy) 5 pt

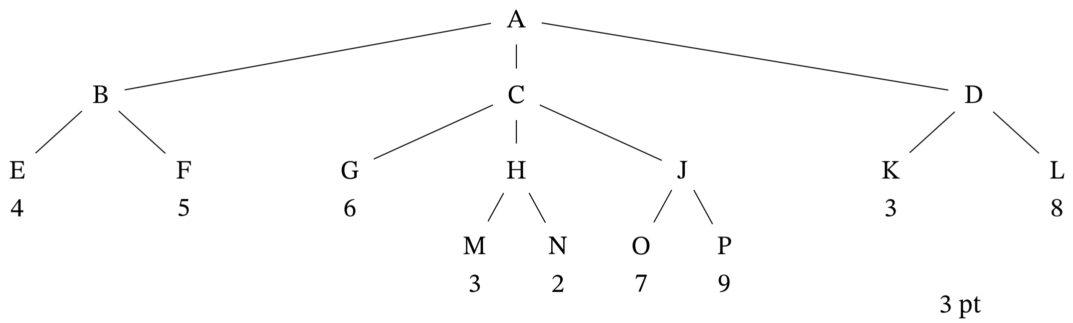
Shortly explain the principle of operation of the A* search. Explain (in few sentences) how it differs from the greedy search?

Solution: A* will expand the nodes in the fringe in an ascending order of the function f(node)=h(node)+g(node), where h(node) is the heuristic of the node and g(node) is the (current) distance from the initial node to this node. Greedy will expand only taking the heuristic into account.

3 Adversarial Search

Problem 3.1 (Game Tree) 10 pt

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



1. What is the minimax value of node A? 3 pt
2. Which move would be selected by Max? 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. 4 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 2 pt
   (a) the minimax value computed at the root?
   (b) The number of nodes pruned by the alpha-beta algorithm?

Solution:

1. A:4
2. B
3. J,O P, L
4. (a) no, (b) yes

---

4 Constraint Satisfaction Problems & Inference

Problem 4.1 (Arc consistency) 5 pt
Define the concept of *arc consistency*.

Solution: A variable u is arc consistent relative to v, if there is either no constraint between u and v, or for every value $d \in D_u$, there is some $d' \in D_v$ such that $(d, d') \in C_{uv}$. A constraint network is arc consistent if all variables are pairwise arc consistent relative to each other.

Problem 4.2 (Scheduling CS Classes) 10 pt

You are in charge of scheduling for computer science classes that meet Mondays, Wednesdays and Fridays. There are 5 classes that meet on these days and 3 professors who will be teaching these classes. You are constrained by the fact that each professor can only teach one class at a time. The classes are:

- Class 1 - *Intro to Artificial Intelligence*: meets 8:30-9:30am,
- Class 2 - *Intro to Programming*: meets 8:00-9:00am,
- Class 3 - *Natural Language Processing*: meets 9:00-10:00am,
- Class 4 - *Machine Learning*: meets 9:30-10:30am,
- Class 5 - *Computer Vision*: meets 9:00-10:00am.

The professors are:

- Professor A, who is available to teach Classes 1, 2, 3, 4, 5.
- Professor B, who is available to teach Classes 3 and 4.
- Professor C, who is available to teach Classes 2, 3, 4, and 5.

1. Formulate this problem as a CSP problem in which there is one variable per class, stating the domains, and constraints. Constraints should be specified formally and precisely, but may be implicit rather than explicit. 3 pt
2. Give the constraint graph associated with your CSP. 2 pt
3. Show the domains of the variables after running arc-consistency on this initial graph (after having already enforced any unary constraints). 3 pt
4. List all optimal cutsets for the constraint graph associated with the CSP. 2 pt

Solution:

1.

Variables	Domains
C_1	A
C_2	A,C
C_3	A,B,C
C_4	A,B,C
C_5	A,C

Constraints: $C_1 \neq C_2, C_1 \neq C_3, C_1 \neq C_5, C_3 \neq C_4, C_3 \neq C_5, C_4 \neq C_5$

2.

$$\begin{tikzcd}
C_2 \arrow[d, dash] & & & \\
C_1 \arrow[rr, dash] \arrow[dr, dash] & & C_3 \arrow[dl, dash] \arrow[dr, dash] & \\
& C_5 \arrow[rr, dash] & & C_4
\end{tikzcd}$$

3.

Variable	Domain
C_1	A
C_2	C
C_3	B
C_4	A
C_5	C

4. The two optimal cutsets are $\{C_3\}$ and $\{C_5\}$.

5 Logic

Problem 5.1 (Natural Deduction) 8 pt

Prove the validity of the following formulae in Natural Deduction. Do not forget to mark your assumptions!

1. $(A \vee B) \Rightarrow (B \vee A)$ 4 pt

   Recall that case distinction can be done with $\vee$-Elimination.

2. $(A \Rightarrow B) \Rightarrow ((B \Rightarrow C) \Rightarrow (A \Rightarrow C))$ 4 pt

Solution:

1.

1(Assumption)^1	$A \vee B$
2(Assumption)^2	A
3 $\vee$ -Introduction	$B \vee A$
4(Assumption)^2	B
5 $\vee$ -Introduction	$B \vee A$
6 $\vee$ -Elimination^2	$B \vee A$
7 $\Rightarrow$ -Introduction^1	$(A \vee B) \Rightarrow (B \vee A)$

2.

1(Assumption)^1	$A \Rightarrow B$
2(Assumption)^2	$B \Rightarrow C$
3(Assumption)^3	A
4 $\Rightarrow$ -Elimination$on1$	B
5 $\Rightarrow$ -Elimination$on2$	C
6 $\Rightarrow$ -introduction^3	$A \Rightarrow C$
7 $\Rightarrow$ -introduction^2	$(B \Rightarrow C) \Rightarrow (A \Rightarrow C)$
8 $\Rightarrow$ -introduction^1	$(A \Rightarrow B) \Rightarrow ((B \Rightarrow C) \Rightarrow (A \Rightarrow C))$

Problem 5.2 (First-Order Tableaux) 7 pt

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

$$\forall X.P(X) \Rightarrow Q(X) \Rightarrow (\forall X.P(X) \Rightarrow \forall X.Q(X))$$

Solution:

$$\begin{array}{c}
\forall X.P(X) \Rightarrow Q(X) \Rightarrow \forall X.P(X) \Rightarrow \forall X.Q(X)^F \\
\forall X.P(X) \Rightarrow Q(X)^T \\
\forall X.P(X) \Rightarrow \forall X.Q(X)^F \\
P(Y) \Rightarrow Q(Y)^T \\
\forall X.P(X)^T \\
\forall X.Q(X)^F \\
P(Z)^T \\
Q(c)^F \\
\begin{array}{c|c}
P(Y)^F & Q(Y)^T \\
\bot[Y/Z] & \bot[c/Y]
\end{array}
\end{array}$$

6 Knowledge Representation

Problem 6.1 (Tableau-Calculus for ALC) 7 pt

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 7.1 (Planning Bike Repair) 10 pt

Consider the following problem. A bicycle has a front wheel and a back wheel installed and both wheels have a flat tire. A robot needs to repair the bicycle. The room also contains a tire pump and a box with all the other equipment needed by the robot to repair a bicycle. The robot can repair a wheel with the help of the box and the tire pump when the robot and the three objects are at the same position. The bicycle is repaired when the robot has done a final overall check which requires both tires to be repaired and to be installed on the bicycle again. For this check, the box is also needed at the same position as the bicycle and 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 bicycle is at position "B", and the "Frontwheel" and the "Backwheel" are installed on the "Bicycle". The "Box" is at position "C" and the "Pump" at position "D". The actions available for the robot are:

- "*Go*" from one position to another. The four possible positions A, B, C, and D are connected in such a way that the robot can reach every other place in one "*Go*".
- "*Push*" an object from one place to another. The bicycle is not pushable and the wheels are only pushable if not installed on the bicycle; obviously the robot cannot push itself; every other object is always pushable. "*Push*" moves both the object and the robot.
- "*Remove*" a wheel from the bike.
- "$RepairWheel$" to fix a wheel with a flat tire.
- "$InstallWheel$" to put a wheel back on the bike.
- "*FinalCheck*" to make sure that not only the two wheels are repaired and installed but also the rest of the bike is in good condition. The box is needed at the same position for this.

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

(b) Write a STRIPS formalization **of the actions** *Remove* **and** *FinalCheck***, and only these two actions**. In doing so, please make use of "object variables", i.e., write the actions up in a parametrized way. State, for each parameter, by which objects it can be instantiated.

In both (a) and (b), make use of only the following predicates:

- $At(x, y)$: To indicate that object $x \in \{Robot, Bicycle, Frontwheel, Backwheel, Box, Pump\}$ is at position $y \in \{Bicycle, A, B, C, D\}$.
- $Pushable(x)$: To indicate that object $x \in \{Robot, Bicycle, Frontwheel, Backwheel, Box, Pump\}$ can be pushed.
- $Repaired(x)$: To indicate that object $x \in \{Robot, Bicycle, Frontwheel, Backwheel, Box, Pump\}$ is repaired.

- $FlatTire(x)$: To indicate that object $x \in \{Robot, Bicycle, Frontwheel, Backwheel, Box, Pump\}$ has a flat tire.

Solution:

(a) Initial state description:

$$\begin{aligned} I \;=\; & \{At(Robot, A), At(Bicycle, B), At(Frontwheel, Bicycle),\\ & At(Backwheel, Bicycle), At(Box, C), At(Pump, D), Pushable(Box),\\ & Pushable(Pump), FlatTire(Frontwheel), FlatTire(Backwheel)\} \end{aligned}$$

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

(b) Action descriptions:

- $Go(x, y) =$

$$\begin{aligned} pre &: \{At(Robot, x)\}\\ add &: \{At(Robot, y)\}\\ del &: \{At(Robot, x)\} \end{aligned}$$

  for all $x, y \in \{A, B, C, D\}$.

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

$$\begin{aligned} pre &: \{At(Robot, y), At(x, y), Pushable(x)\}\\ add &: \{At(Robot, z), At(x, z)\}\\ del &: \{At(Robot, y), At(x, y)\} \end{aligned}$$

  for all $x \in \{Robot, Bicycle, Wheel, Box, Pump\}$, $y, z \in \{A, B, C, D\}$.

- $Remove(x, y) =$

$$\begin{aligned} pre &: \{At(Robot, x), At(Bicycle, x), At(y, Bicycle)\}\\ add &: \{At(y, x), Pushable(y)\}\\ del &: \{At(y, Bicycle)\} \end{aligned}$$

  for all $x \in \{A, B, C, D\}$, $y \in \{Frontwheel, Backwheel\}$.
  [Note that the facts Pushable(Frontwheel) and Pushable(Backwheel) can also be initially given and never deleted because of the way the action "Push" is defined.]

- $RepairWheel(x, y) =$

$$\begin{aligned} pre &: \{At(Robot, x), At(y, x), FlatTire(y), At(Box, x), At(Pump, x)\}\\ add &: \{Repaired(y)\}\\ del &: \{FlatTire(y)\} \end{aligned}$$

  for all $x \in \{A, B, C, D\}$, $y \in \{Frontwheel, Backwheel\}$.

- $InstallWheel(x, y) =$

$$\begin{aligned} pre &: \{At(Robot, x), At(Bicycle, x), At(y, x), Repaired(y), Pushable(y)\}\\ add &: \{At(y, Bicycle)\}\\ del &: \{At(y, x), Pushable(y)\} \end{aligned}$$

  for all $x \in \{A, B, C, D\}$, $y \in \{Frontwheel, Backwheel\}$.

- $FinalCheck(x) =$

$$\begin{aligned} pre &: \{At(Robot, x), At(Bicycle, x), At(Box, x),\\ &\quad At(Frontwheel, Bicycle), At(Backwheel, Bicycle),\\ &\quad Repaired(Frontwheel), Repaired(Backwheel)\}\\ add &: \{Repaired(Bicycle)\} \end{aligned}$$