Name:

Birth Date:

Matriculation Number:

Retake Exam
Artificial Intelligence 1

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														

Exam Grade: Bonus Points: Final Grade:

Organizational Information

Please read the following directions carefully and acknowledge them with your signature.

1. Please place your student ID card and a photo ID on the table for checking
2. The grading information on the cover sheet holds with the proviso of further checking.
3. no resources or tools are allowed except for a pen.
4. You have 90 min(sharp) for the test
5. You can reach 83 points if you fully solve all problems. You will only need 80 points for a perfect score, i.e. 3 points are bonus points.
6. Write the solutions directly on the sheets.
7. If you have to abort the exam for health reasons, your inability to sit the exam must be certified by an examination at the University Hospital. Please notify the exam proctors and have them give you the respective form.
8. Please make sure that your copy of the exam is complete (20 pages excluding cover sheet and organizational information pages) and has a clear print. **Do not forget to add your personal information on the cover sheet and to sign this declaration.**

Declaration: With my signature I certify having received the full exam document and having read the organizational information above.

Erlangen, August 10., 2020 .....
(signature)

Organisatorisches

Bitte lesen die folgenden Anweisungen genau und bestätigen Sie diese mit Ihrer Unterschrift.

1. Bitte legen Sie Ihren Studentenausweis und einen Lichtbildausweis zur Personenkontrolle bereit!
2. Die angegebene Punkteverteilung gilt unter Vorbehalt.
3. Es sind keine Hilfsmittel erlaubt außer eines Stifts.
4. Die Lösung einer Aufgabe muss auf den vorgesehenen freien Raum auf dem Aufgabenblatt geschrieben werden; die Rückseite des Blatts kann mitverwendet werden. Wenn der Platz nicht ausreicht, können bei der Aufsicht zusätzliche Blätter angefordert werden.
5. Wenn Sie die Prüfung aus gesundheitlichen Gründen abbrechen müssen, so muss Ihre Prüfungsunfähigkeit durch eine Untersuchung in der Universitätsklinik nachgewiesen werden. Melden Sie sich in jedem Fall bei der Aufsicht und lassen Sie sich das entsprechende Formular aushändigen.
6. Die Bearbeitungszeit beträgt 90 min.
7. Sie können 83 Punkte erreichen, wenn Sie alle Aufgaben vollständig lösen. Allerdings zählen 80 Punkte bereits als volle Punktzahl, d.h. 3 Punkte sind Bonuspunkte.
8. Überprüfen Sie Ihr Exemplar der Klausur auf Vollständigkeit (20 Seiten exklusive Deckblatt und Hinweise) und einwandfreies Druckbild! **Vergessen Sie nicht, auf dem Deckblatt die Angaben zur Person einzutragen und diese Erklärung zu unterschreiben!**

Erklärung: Durch meine Unterschrift bestätige ich den Empfang der vollständigen Klausurunterlagen und die Kenntnisnahme der obigen Informationen.

Erlangen, August 10., 2020 .....

(Unterschrift)

Please consider the following rules; otherwise you may lose points:

- If you continue an answer on another page, please indicate the problem number on the new page and give a page reference on the old page.
- Always justify your statements (we would like to give poins for incorrect answers). Unless you are explicitly allowed to, do not just answer "yes", "no", or "42".
- If you write program code, give comments!

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.

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.

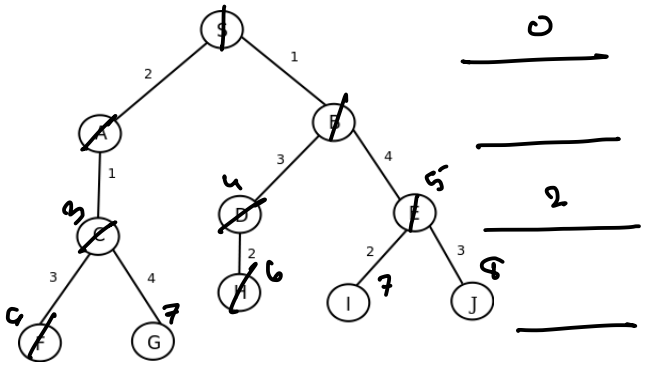
Hint: It may be a good idea to write a parent/2 predicate. Only use it to write the predicate ancestor/2.

---

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	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

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	N	N	Y	N
Complete	Y	N	Y	Y

greedy A*
N Y
N Y

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 $h(n) = h^*(n)$?

$h(n)=0$ it will behave like uninformed search

$h(n)=h^*(n)$ → 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?

A* will expand the nodes in fringe in an ascending order of f_n

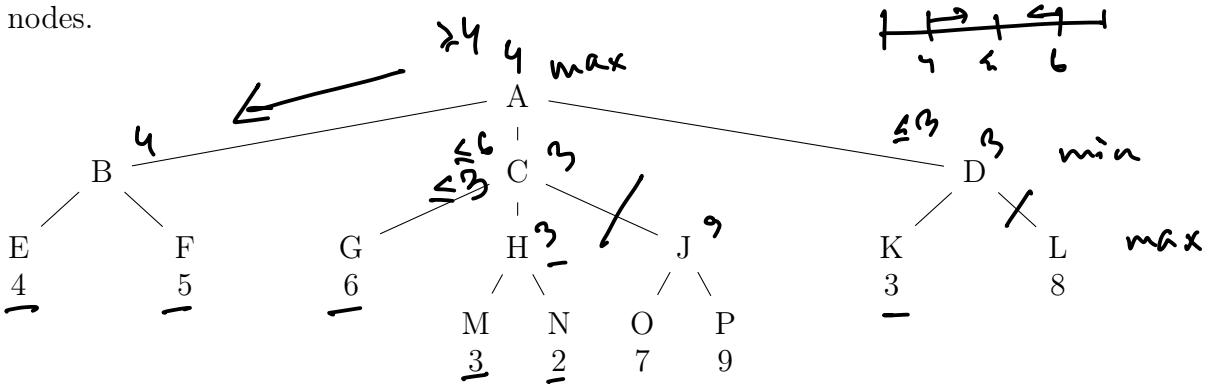
$f(n) = g(n) + h(n)$ → I to n path cost

greedy. $f(n) = \underline{h(n)}$

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.



3 pt

1. What is the minimax value of node A? 1 pt
2. Which move would be selected by Max? 4 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?

This page was intentionally left blank for extra space

4 Constraint Satisfaction Problems & Inference

Problem 4.1 (Arc consistency) 5 pt

Define the concept of *arc consistency*.

In (u, v) pair, u is arc-Consistent w.r.t v
if there is no constraint between u & v
or for every value $d \in D_u$, there is some $d' \in D_v$
such that C_{uv} is satisfy.

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.

3 pt

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.

2 pt

2. Give the constraint graph associated with your CSP.

3 pt

3. Show the domains of the variables after running arc-consistency on this initial graph (after having already enforced any unary constraints).

2 pt

4. List all optimal cutsets for the constraint graph associated with the CSP.

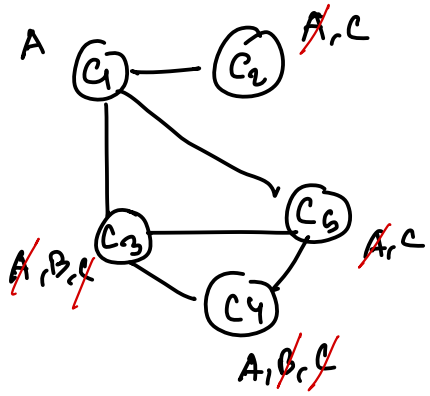
01.

V		D
C1	8:30 - 9:30	A
C2	8:00 - 9:00	A C
C3	9:00 - 10:00	A B C
C4	9:30 - 10:30	A B C
C5	9:00 - 10:00	A C

C

$C_1 \neq C_2$ $C_3 \neq C_4$ $C_4 \neq C_5$
$C_1 \neq C_3$ $C_3 \neq C_5$
$C_1 \neq C_5$

02.



03.

C_1 — A
C_2 — C
C_3 — B
C_4 — A
C_5 — C

04. $\{C_3\}$ or $\{C_5\}$ → Creates Cycle

This page was intentionally left blank for extra space

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

01.
1. $A \vee B$ ~~Assumption 1~~
2. B ~~Assumption 2~~
3. $B \vee A$ $\vee I$ on 2
4. A ~~Assumption 2~~
5. $B \vee A$ $\vee I$ on 4
6. $B \vee A$ $\vee E$
7. $(A \vee B) \Rightarrow (B \vee A)$ $\Rightarrow I$ on 1, 6

ND.

$$\frac{A \vee B \qquad \begin{matrix}[A]^1 \\ \vdots \\ C\end{matrix} \qquad \begin{matrix}[B]^1 \\ \vdots \\ C\end{matrix}}{C}\ \vee E$$

02.
1. $A \Rightarrow B$ ~~Assumption 1~~
2. $B \Rightarrow C$ ~~Assumption 2~~
3. A ~~Assumption 3~~
4. B $\Rightarrow E$ on 1, 3
5. C $\Rightarrow E$ on 2, 4
6. $A \Rightarrow C$ $\Rightarrow I$ on 3, 5
7. $(B \Rightarrow C) \Rightarrow (A \Rightarrow C)$ $\Rightarrow I$ on 2, 6
8. $(A \Rightarrow B) \Rightarrow ((B \Rightarrow C) \Rightarrow (A \Rightarrow C))$ $\Rightarrow I$ on 1, 7

Problem 5.2 (First-Order Tableaux) 7 pt

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

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

$(\forall x . P(x) \Rightarrow Q(x)) \Rightarrow ((\forall Y . P(Y)) \Rightarrow (\forall z . Q(z)))^F$

$\forall x . P(x) \Rightarrow Q(x)^T$

$\forall Y . P(Y) \Rightarrow \forall z . Q(z)^F$

$\forall Y . P(Y)^T$

$\forall z . Q(z)^F$

$P(w) \Rightarrow Q(w)^T$

$P(c)^T$

$Q(c)^F$

$P(w)^F$ | $Q(w)^T$

$\bot [c/w]$ | $\bot [c/w]$

??

This page was intentionally left blank for extra space

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 $x\colon C$ in $\mathcal{T}_{\mathcal{ALC}}$, where C is the concept above.

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

$x : \forall takes.GLOIN \sqcup \forall takes.AI1$

$x : \exists takes.(\overline{AI1} \sqcap \overline{GLOIN})$

$x\ takes\ y$

$y : \overline{AI1} \sqcap \overline{GLOIN}$

$y : \overline{AI1}$

$y : \overline{GLOIN}$

$x : \forall takes.GLOIN$	$x : \forall takes.AI1$
$y : GLOIN$	$y : AI1$
$\bot$	$\bot$

This page was intentionally left blank for extra space

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.

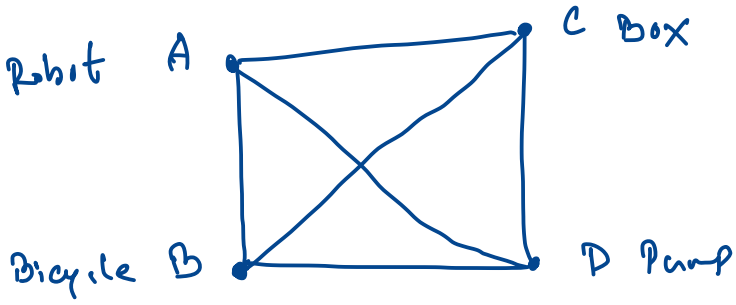
(a) Write a STRIPS formalization of the initial state and goal descriptions.

(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.



go(x, y)
push(x, y, z)

(a) $I = \{$ At(Robot, x), At(Bicycle, B), At(Box, C), At(Pump, D), At(FW, Bicycle), At(BW, Bicycle), pushable(Pump), pushable(Box), Flattire(FW), Flattire(BW) $\}$

$G = \{$ Repaired(Bicycle) $\}$

(b) Remove(x, y) = pre: $\{$ At(Robot, x), At(Bicycle, x), At(y, Bicycle) $\}$ (y: FW, BW)
add: $\{$ At(y, x), pushable(y) $\}$
del: $\{$ At(y, Bicycle) $\}$

$x \in \{A, B, C, D\}$
$r \in \{FW, BW\}$

FindCheck(x) = pre: $\{$ At(Robot, x), At(Bicycle, x), At(Bike, x), At(FW, Bicycle), At(BW, Bicycle), Repaired(FW), Repaired(BW) $\}$
add: $\{$ Repaired(Bicycle) $\}$
del: $\{\}$

$x \in \{A, B, C, D\}$

This page was intentionally left blank for extra space

This page was intentionally left blank for extra space

This page was intentionally left blank for extra space