Name:

Birth Date:

Matriculation Number:

Exam
Artificial Intelligence 1

Feb 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	5.1	5.2	5.3	6.1	7.1	Sum	grade
total	6	6	2	8	4	10	16	7	5	3	7	18	92	
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 92 points if you fully solve all problems. You will only need 87 points for a perfect score, i.e. 5 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 (22 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, Feb 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 92 Punkte erreichen, wenn Sie alle Aufgaben vollständig lösen. Allerdings zählen 87 Punkte bereits als volle Punktzahl, d.h. 5 Punkte sind Bonuspunkte.
8. Überprüfen Sie Ihr Exemplar der Klausur auf Vollständigkeit (22 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, Feb 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 (Girls are witches) 6 pt

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.

0, 1, 1, 2, 3, 5, ...

Problem 1.2 (Prolog Fibbonaci) 6 pt

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

```
fibonacci(0,0).
fibonacci(1,1).
fib(N,X) :- N>1,
            N1 is N-1,
            N2 is N-2,
            fib(N1,Y),
            fib(N2,Z),
            X is Y+Z.
```

This page was intentionally left blank for extra space

2 Search

Problem 2.1 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.

No, there can be cycle

yes, if it is tree, then no cycles

Problem 2.2 (Relaxed Problem) 8 pt

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.

Problem P → s → s' → cost

Relaxed P' → s → s' → cost

graphically, P' has all arc from P / maybe more

so, optimal path in P' same / or better than optimal path P

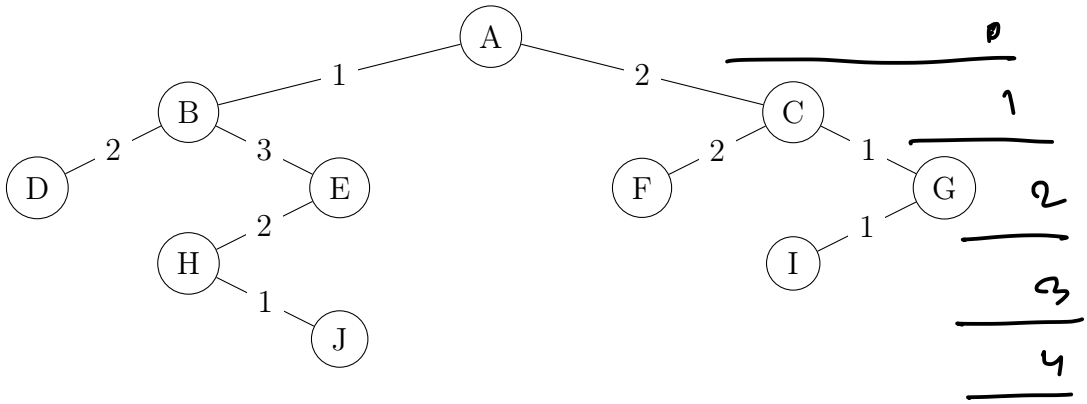
↳ using additional arcs.

→ therefore $c'(s) \leq C(s)$

which is admissible

Problem 2.3 (Uninformed search) 4 pt

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:

01. d=0 A
d=1 A B C
d=2 A B D E C F G
d=3 A B D E H C F G I
d=4 A B D E H J C F G I

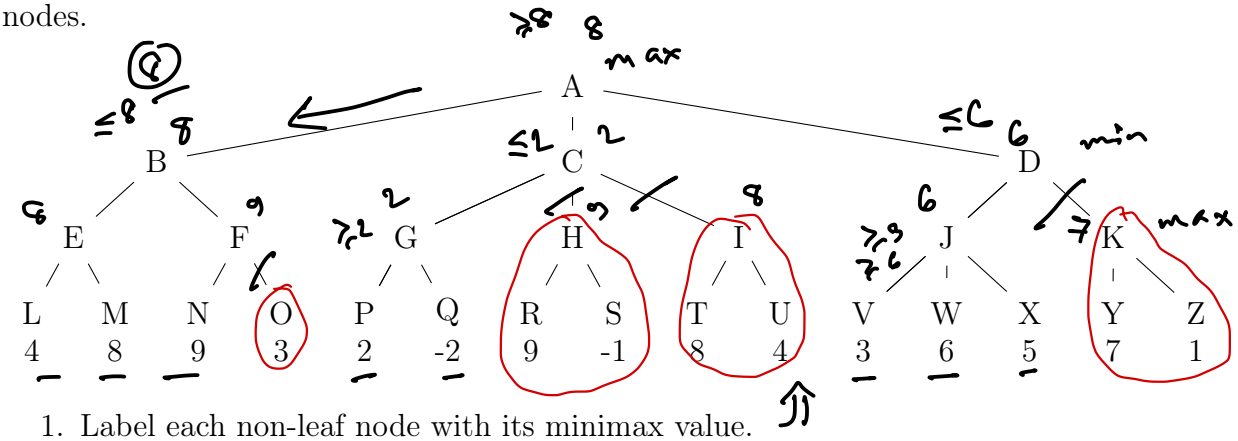
02. f: ~~B~~ ~~C~~ ~~D~~ ~~E~~ ~~F~~ ~~G~~ ~~I~~ ~~H~~ J
c: 1 2 3 4 4 3 4 6 7
v: A B C D G E F I H J

This page was intentionally left blank for extra space

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 at the leaves are the static evaluation function values of the states at each of those nodes.



1. Label each non-leaf node with its minimax value. 4 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. 3 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?

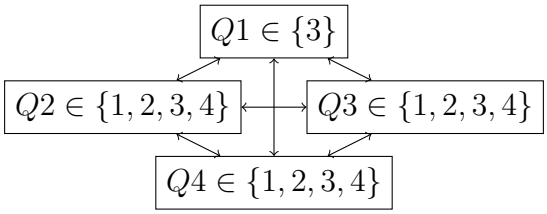
This page was intentionally left blank for extra space

4 Constraint Satisfaction Problems & Inference

Problem 4.1 16 pt

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.

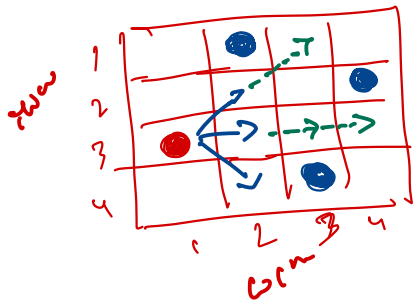
	Q1	Q2	Q3	Q4
Initial domain	3	1, ~~2~~, ~~3~~, ~~4~~	~~1~~, 2, ~~3~~, 4	1, 2, 3, 4
After $Q2 \to Q1$	3	{1}	{1,2,3,4}	{1,2,3,4}
After $Q3 \to Q1$	3	{1}	{2,4}	{1,2,3,4}
After $Q2 \to Q3$	3	{1}	{2,4}	{1,2,3,4}
After $Q3 \to Q2$	3	{1}	{4}	{1,2,3,4}

2 pt

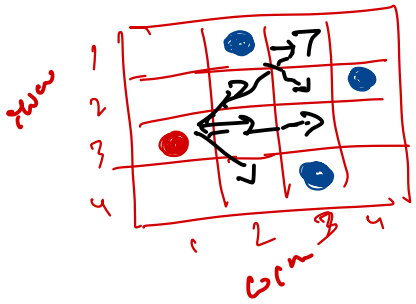
4. In general, when will the arc consistency algorithm halt?

This page was intentionally left blank for extra space

01. $Q_1 = \{3\}$ → $Q_2 = \{1\}$, $Q_3 = \{2, 4\}$, $Q_4 = \{1, 2, 4\}$



02. (V, w) pair, V is arc-consistent w.r.t w if for every val of $v \in D_v$, there is a val for $w \in D_w$ such that C_{vw} is satisfied.



04. arc-consistency algo halt! when no constraints changes the domain anymore

5 Logic

Problem 5.1 (Natural Deduction) 7 pt

Prove (or disprove) the validity of the following formulae in Natural Deduction:

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

01. 1. $(P \Rightarrow Q) \wedge P$ Assumption
2. $P \Rightarrow Q$ $\wedge E_L$ on 1
3. P $\wedge E_R$ on 1
4. Q $\Rightarrow E$ on 2,3
5. $((P \Rightarrow Q) \wedge P) \Rightarrow Q$ $\Rightarrow I$ on 1,4

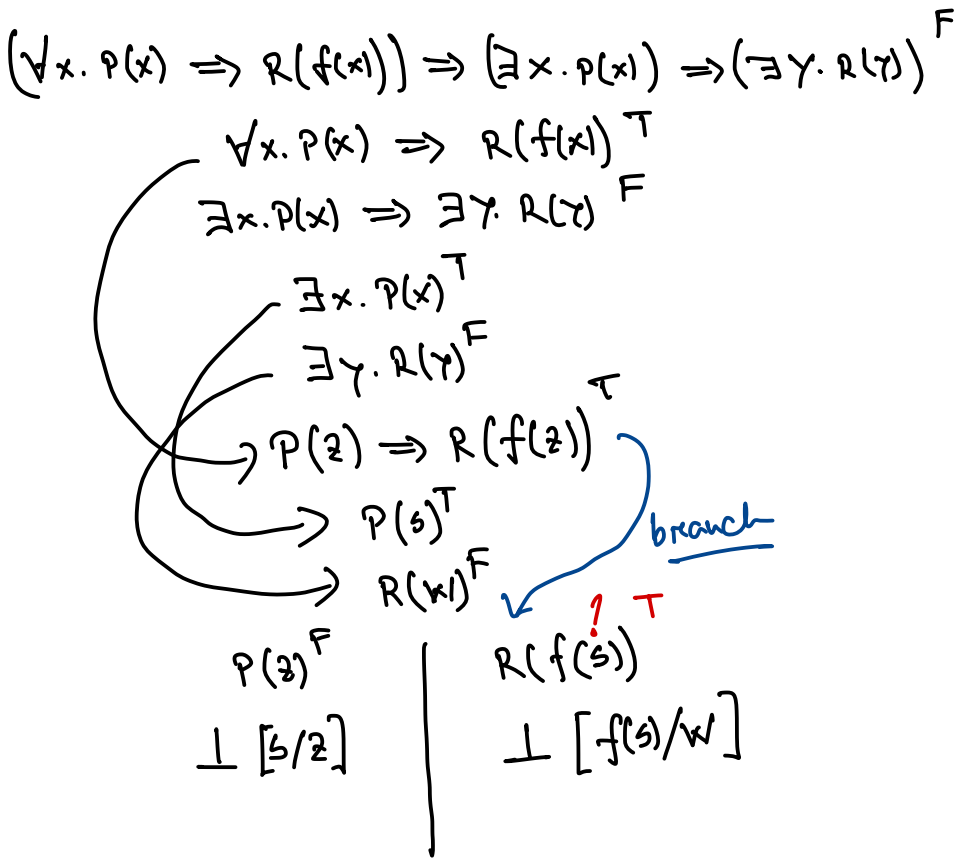
02. 1. $\neg Q \Rightarrow \neg P$ Assumption
2. $\neg Q$ Assumption
3. P Assumption
4. $\neg P$ $\Rightarrow E$ on 1, 2
5. F FI on 3,4
6. $\neg\neg Q$ $\neg I$ on 2,5
7. $P \Rightarrow \neg\neg Q$ $\Rightarrow I$ on 3, 6
8. $(\neg Q \Rightarrow \neg P) \Rightarrow (P \Rightarrow \neg\neg Q)$ $\Rightarrow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

This page was intentionally left blank for extra space

Problem 5.2 (First-Order Tableau) 5 pt

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

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



Problem 5.3 (Unification) 3 pt

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.

$$\frac{f(X, g(Y,X)) =^? f(Y,Z)}{X =^? Y \wedge g(Y,X) =^? Z} \; \forall_{dec}$$

$$\frac{X =^? Y \wedge g(Y,X) =^? Z}{X =^! Y \wedge g(Y,Y) =^? Z} \; \forall_{elim}$$

MGU: $[Y/X], [g(Y,Y)/Z]$

if, we give one more unifier then

$[a/X], [a/Y], [g(a,a)/Z]$

Not MGU because $[a/Y]$ → (extra)

6 Knowledge Representation

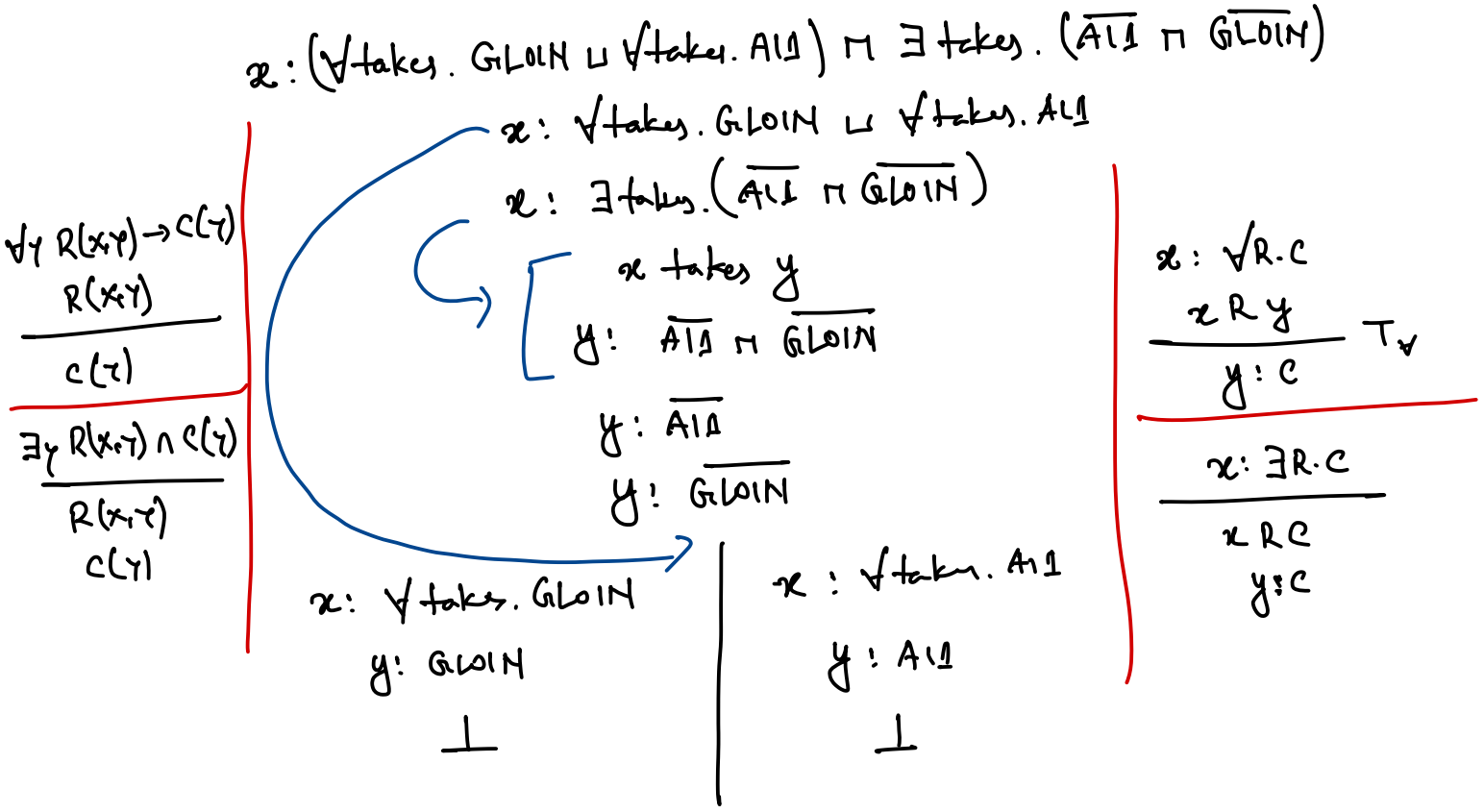
Problem 6.1 (Tableau-Calculus for ALC) 7 pt

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.



This page was intentionally left blank for extra space

7 Planning

Problem 7.1 18 pt

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.

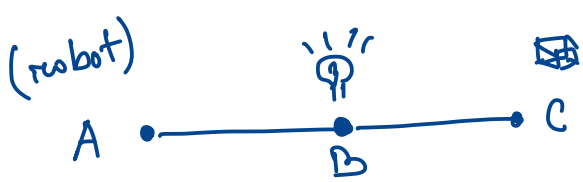
Note that the robot can only push an object or climb onto an object if both of them are at the same location. Furthermore, in case of pushing an object the robot changes to the destination location as well.

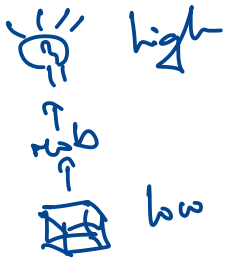
(a) Write a STRIPS formalization of the initial state and goal descriptions.

(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.





This page was intentionally left blank for extra space

01. $I = \{$ At(Robot, A), At(Bulb, B), At(Box, C), Hight(Robot, Low), Height(Bulb, High), Hight(Box, Low), Pushable(Box), Climbable(Box) $\}$

$G_2 = \{$ Repaired(Bulb) $\}$

02. Action

· GO(x, y) = pre: $\{$ Hight(Robot, Low), At(Robot, x) $\}$
add: $\{$ At(Robot, y) $\}$
del: $\{$ At(Robot, x) $\}$
for all x, y $\in \{A, B, C\}$

· Push(x, y, z) = pre: $\{$ At(Robot, y), Pushable(x), At(x, y), Height(Robot, Low), Height(x, Low) $\}$
add: $\{$ At(x, z), At(Robot, z) $\}$
del: $\{$ At(Robot, y), At(x, y) $\}$
for all $x \in \{$Box, Bulb, Robot$\}$
$y, z \in \{A, B, C\}$

· ClimbUP(x, y) = pre: $\{$ At(Robot, y), Climbable(x), At(x, y), Hight(Robot, Low), Height(x, Low) $\}$
add: $\{$ Height(Robot, high) $\}$
del: $\{$ Height(Robot, Low) $\}$
for all $x \in \{$ Box, Bulb, Robot $\}$
$y \in \{A, B, C\}$

This page was intentionally left blank for extra space

. 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: { }

x ∈ {Box, Bulb, Robot}
y ∈ {A, B, C}
z ∈ {High, Low}

Repair{Bulb, y} = pre: { At(Robot, y), At(Bulb, y), At(Box, y)
Height(Robot, high), Height(Bulb, high) }
add: { Repaired(Bulb) }
del: { }

This page was intentionally left blank for extra space

This page was intentionally left blank for extra space