Name:

Birth Date:

Matriculation Number:

Exam
Artificial Intelligence 1

July 29., 2019

	To be used for grading, do not write here														
prob.	1.1	2.1	2.2	2.3	3.1	3.2	4.1	4.2	4.3	5.1	5.2	5.3	6.1	Sum	grade
total	4	3	3	3	3	10	5	5	5	6	8	10	10	75	
reached															

Exam Grade: Bonus Points: Final Grade:

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.

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.

In any case, grading student's answers is not a process of simply "comparing with the reference solution".

In the course Artificial Intelligence I/II we award 5 bonus points for the first student who reports a factual error (please report spelling/formatting errors as well) in an assignment or old exam and 10 bonus points for an alternative solution (formatted in LaTeX) that is usefully different from the existing ones.

1 Prolog

Problem 1.1 4 pt

1. Program a Prolog predicate uadd for addition and umult for multiplication in unary representation. 4 min

---

Hint: The number 3 in unary representation is the `ProLog` term s(s(s(o))), i.e. application of the arbitrary function s to an arbitrary value o iterated three times.

Hint: Note that `ProLog` does not allow you to program (binary) functions, so you must come up with a three-place predicate. You should use add(X,Y,Z) to mean $X+Y=Z$ and program the recursive equations $X+0=X$ (base case) and $X+s(Y)=s(X+Y)$.

---

2. Write a Prolog predicate ufib that computes the n^{th} Fibonacci Number (0, 1, 1, 2, 3, 5, 8, 13,... add the last two to get the next), using the addition predicate above.

Solution:

```
uadd(X,o,X).
uadd(X,s(Y),s(Z)) :- uadd(X,Y,Z).

umult(_,o,o).
umult(X,s(Y),Z) :- umult(X,Y,W), uadd(X,W,Z).

ufib(o,o).
ufib(s(o),s(o)).
ufib(s(s(X)),Y):-ufib(s(X),Z),ufib(X,W),uadd(Z,W,Y).
```

---

2 Search

Problem 2.1 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. 3 pt 3 min

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

Problem 2.2 (A^* Theory)

What is the condition on the heuristic function that makes A^* optimal? Does a heuristic with this condition always exist? 3 pt 3 min

Solution:Admissible heuristic - always underestimates the real cost to the goal. This always exists: $h(x)=0$.

Problem 2.3 (A^* vs. BFS)

Does A^* search always expand fewer nodes than BFS? Justify you answer. 3 pt 3 min

Solution:No. With a bad heuristic, A^* can be forced to explore the whole space, just like BFS.

3 Adversarial Search

Problem 3.1 (Minimax Restrictions)

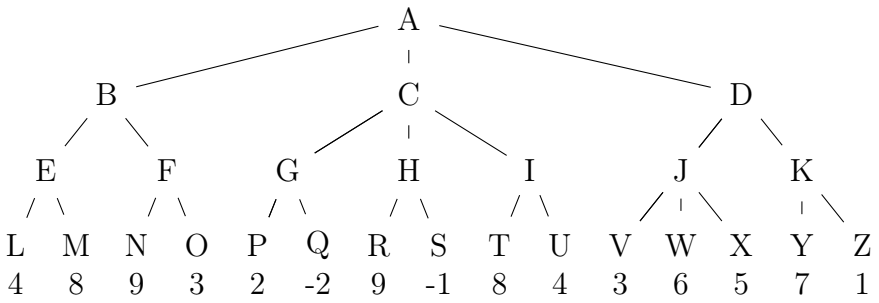
Name at least five criteria that a game has to satisfy in order for the minimax algorithm to be applicable. 3 pt 3 min

Solution:Any five of:

- Two-player
- Determininstic
- Fully observable
- Players alternate
- Finitely many / discrete game states
- Zero-sum
- Game ends after finitely many rounds

Problem 3.2 (Minimax Algorithm and Alphabeta Pruning)

Consider the following (complete!) search tree: 10 pt 10 min



1. What is the minimax value at node B?
2. Which subtrees in the tree can be pruned during alpha-beta search? What is the criterion for pruning a subtree?

4 Constraint Satisfaction Problems & Inference

Problem 4.1 (Constraint Networks)

A constraint network is a triple $\langle V, D, C \rangle$. Explain the roles of V, D, and C. If you use the word "constraint" you have to define and briefly explain it. 5 pt 5 min

Solution:V: a set of variables

D: a set of sets D_v of values for each $v \in V$

C: a set of constraints; i.e. relations on (the domains of the) variables.

Problem 4.2 (Arc consistency)

Define the concept of *arc consistency*. 5 pt

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. 5 min

Problem 4.3 (50 Queens)

Formalize the *50 Queens Problem* as a constraint network. Hint: You do not have to write down all the constraints explicitly, but it has to be clear what the exact constraints are. 5 pt

5 min

5 Logic

Problem 5.1 (Natural Deduction)

Prove the validity of the following formula in Natural Deduction: 6 pt

$((A \vee B) \wedge (A \Rightarrow C) \wedge (B \Rightarrow C)) \Rightarrow C$ 6 min

Solution:

(1)	1	$(A \vee B) \wedge ((A \Rightarrow C) \wedge (B \Rightarrow C))$	Assumption
(2)	1	$(A \vee B)$	$\wedge E_\ell$ (on 1)
(3)	1	$(A \Rightarrow C) \wedge (B \Rightarrow C)$	$\wedge E_r$ (on 1)
(4)	1	$(A \Rightarrow C)$	$\wedge E_\ell$ (on 3)
(5)	1	$(B \Rightarrow C)$	$\wedge E_r$ (on 3)
(6)	1,6	A	Assumption
(7)	1,6	C	$\Rightarrow E$ (on 4 and 6)
(8)	1,8	B	Assumption
(9)	1,8	C	$\Rightarrow E$ (on 5 and 8)
(10)	1	C	$\vee E$ (on 2, 7 and 9)
(11)		$((A \vee B) \wedge (A \Rightarrow C) \wedge (B \Rightarrow C)) \Rightarrow C$	$\Rightarrow I$ (on 1 and 10)

Problem 5.2 (First-Order Tableaux)

Prove the following formula using the first-order free variable tableaux calculus. We have $P \in \Sigma_1^p$. 8 pt

8 min

$$\exists X.\, (P(X) \Rightarrow \forall Y.P(Y))$$

Solution:

(1)	$\exists X.(P(X) \Rightarrow \forall Y.P(Y))^F$	
(2)	$P(V_X) \Rightarrow \forall Y.P(Y)^F$	(from 1)
(3)	$P(V_X)^T$	(from 2)
(4)	$\forall Y.P(Y)^F$	(from 2)
(5)	$P(c_Y)^F$	(from 4)
(6)	$\bot[c_Y/V_X]$	

Problem 5.3 (First-Order Resolution)

Prove the following formula using resolution. 10 pt

10 min

$P \in \Sigma_1^p, R \in \Sigma_2^p, a, b \in \Sigma_0^f$

$$\exists X.\forall Y.\exists Z.\exists W.\,((\neg P(Z) \wedge \neg R(b,a)) \vee \neg R(a,b) \vee R(W,a) \vee (P(Y) \wedge R(X,b)))$$

Solution:We negate:

$$\forall X.\exists Y.\forall Z.\forall W.(P(Z) \vee R(b,a)) \wedge R(a,b) \wedge \neg R(W,a) \wedge (\neg P(Y) \vee \neg R(X,b))$$

We skolemize:

$$(P(Z) \vee R(b,a)) \wedge R(a,b) \wedge \neg R(W,a) \wedge (\neg P(f_Y(X)) \vee \neg R(X,b))$$

This yields the clauses $\{P(Z)^T, R(b,a)^T\}, \{R(a,b)^T\}, \{R(W,a)^F\}, \{P(f_Y(X))^F, R(X,b)^F\}$. We resolve:

$$\{P(Z)^T, R(b,a)^T\} + \{R(W,a)^F\}[b/W] \Longrightarrow \{P(Z)^T\}$$
$$\{R(a,b)^T\} + \{P(f_Y(X))^F, R(X,b)^F\}[a/X] \Longrightarrow \{P(f_Y(a))^F\}$$
$$\{P(Z)^T\}[f_Y(a)/Z] + \{P(f_Y(a))^F\} \Longrightarrow \{\}$$

---

6 Planning

Problem 6.1 Cheeta is an intelligent and lazy monkey. Your task is to help him grab a banana. 10 pt 10 min

Initially, the monkey and a boat are "Low" on the ground at position "A" and the banana hangs "High" at position "B" on a tree. Cheeta can climb up the tree where the banana hangs, meaning that it can climb from height "Low" to "High" at B. Cheeta can grab the banana when they are both at the same position and at the same height.

There is a river between "A" and "B", and Cheeta can only travel between these positions by boat. Cheeta can only enter the boat if they are at the same position and height (the boat is a height "Low").

The following STRIPS model is used. The facts are:

- $At(x,y)$: $x \in \{Boat, Banana, Cheeta\}$ is at position $y \in \{A, B, Boat\}$.
- $Height(x,y)$: $x \in \{Boat, Banana, Cheeta\}$ is at height $y \in \{Low, High\}$.
- $Climbable(x)$: Cheeta can climb at position $x \in \{A, B\}$.
- $Grabbed()$: Cheeta has grabbed the banana.

The goal for Cheeta is to grab the banana using the following available actions:

- $Drive(x,y)$ to get from x to y
- $GetIn(x)$ to get in the boat (at location x)

- $GetOut(x)$ to get out of the boat (at location x)
- $Climb(x, y, z)$ to climb at position x from height y to height z
- $Grab(x, y)$ to grab the banana (at position x and height y)

(a) Properly define the actions $Drive(x, y)$ and $GetIn(x)$

(b) Give the initial state and a solution to this planning problem

---

Solution:

(a)
- $Drive(x, y) =$

  $pre : \{At(Cheeta, Boat), At(Boat, x), Height(Boat, Low)\}$
  $add : \{At(Boat, y)\}$
  $del : \{At(Boat, x)\}$

  for all $x, y \in \{A, B\}$.
- $GetIn(x) =$

  $pre : \{At(Cheeta, x), At(Boat, x), Height(Cheeta, Low), Height(Boat, Low)\}$
  $add : \{At(Cheeta, Boat)\}$
  $del : \{At(Cheeta, x)\}$
- $GetOut(x) =$

  $pre : \{At(Cheeta, Boat), At(Boat, x), Height(Cheeta, Low), Height(Boat, Low)\}$
  $add : \{At(Cheeta, x)\}$
  $del : \{At(Cheeta, Boat)\}$

  for all $x \in \{A, B\}$.
- $ClimbUp(x, y, z) =$

  $pre : \{At(Cheeta, x), Climbable(x), Height(Cheeta, y), Next(y, z)\}$
  $add : \{Height(Cheeta, z)\}$
  $del : \{Height(Cheeta, y)\}$

  for all $x \in \{A, B\}$, $y, z \in \{Low, Middle, High\}$.
- $ClimbDown(x, y, z) =$

  $pre : \{At(Cheeta, x), Climbable(x), Height(Cheeta, y), Next(z, y)\}$
  $add : \{Height(Cheeta, z)\}$
  $del : \{Height(Cheeta, y)\}$

  for all $x \in \{A, B\}$, $y, z \in \{Low, Middle, High\}$.

- $UseLiana() =$

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

- $Grab(x, y) =$

$$pre : \{At(Cheeta, x), At(Banana, x), Height(Cheeta, y), Height(Banana, y)\}$$
$$add : \{Grabbed()\}$$
$$del : \{At(Banana, x), Height(Banana, y)\}$$

for all $x \in \{A, B\}$, $y \in \{Low, Middle, High\}$.

(b)
