Name:

Birth Date:

Matriculation Number:

Exam
Künstliche Intelligenz 1

July 16., 2018

	To be used for grading, do not write here												
prob.	1.1	1.2	2.1	2.2	3.1	3.2	4.1	4.2	5.1	5.2	6.1	Sum	grade
total	6	12	5	5	3	12	2	10	10	6	10	81	
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", therefore errors in the "solutions" are not a problem in this case.

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 6 pt

6 min

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

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).
```

Problem 1.2 (DFS in Prolog) 12 pt

12 min

We want to implement DFS in ProLog using the following data structures for search trees:

```
subtrees([]).
subtrees([(Cost,T)|Rest]) :- number(Cost),istree(T), subtrees(Rest).
istree(tree(_,Children)) :- subtrees(Children).
```

Write a Prolog predicate dfs such that dfs(G,T,X,Y) on a tree T returns the path to the goal G in X and the cost of the path in Y

Solution:

```
dfs(GoalValue,tree(GoalValue,_),GoalValue,0).
dfs(GoalValue,tree(Value,[(Cost,T)|Rest]),Path,FinalCost) :- T = tree(IV,_), write(IV ),
dfs(GoalValue, T,P,C),string_concat(Value,P,Path),FinalCost is C+Cost; % go down one depth level
dfs(GoalValue,tree(Value,Rest),Path,FinalCost). % next child
```

---

2 Search

Problem 2.1 (A^* vs. BFS) 5 pt

Does A^* search always expand fewer nodes than BFS? Justify you answer. 5 min

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

Problem 2.2 (A looping greedy search) 5 pt

Draw a graph and give a heuristic so that a greedy search for a path from a node A to a node B gets stuck in a loop. Draw the development of the search tree, starting from A, until one node is visited for the second time. 5 min

Indicate, in one or two sentences, how the search algorithm could be modified or changed in order to solve the problem without getting stuck in a loop.

Solution:

1. Consider the Example from the lecture about traveling between cities in Romania. Then use the case of traveling from Iasi to Oradea.
2. Use A^*, or remember which nodes have been visited before and don't visit them again.

3 Adversarial Search

Problem 3.1 (Minimax Restrictions) 3 pt

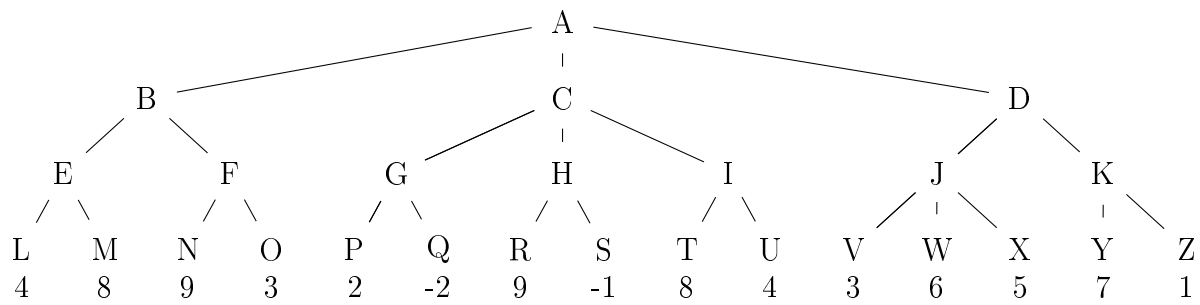
Name at least five criteria that a game has to satisfy in order for the minimax algorithm to be applicable. 3 min

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 (Game Tree) 12 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. 12 min



1. Compute the minimax game value of nodes A, B, C, and D
2. Which move would be selected by Max?
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.

Solution:

1. B = 8, C = 2, D = 6, A = 8
2. B
3. O, H (and R and S), I (and T and U), K (and Y and Z)

---

4 Constraint Satisfaction Problems & Inference

Problem 4.1 (Arc consistency) 2 pt

Define the concept of *arc consistency*. 2 min

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: 10 min

- Class 1 - Intro to Programming: meets from 8:00-9:00am
- Class 2 - Intro to Artificial Intelligence: meets from 8:30-9:30am
- Class 3 - Natural Language Processing: meets from 9:00-10:00am
- Class 4 - Computer Vision: meets from 9:00-10:00am

- Class 5 - Machine Learning: meets from 9:30-10:30am

The professors are:

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

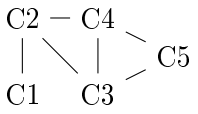
1. Formulate this problem as a CSP problem in which there is one variable per class, stating the domains, and constraints. Constraints should be specified formally and precisely, but may be implicit rather than explicit.
2. Give the constraint graph associated with your CSP
3. Show the domains of the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

Solution:

1.

Variables	Domains
C1	C
C2	B,C
C3	A,B,C
C4	A,B,C
C5	B,C

Constraints: C1 $\neq$ C2, C2 $\neq$ C3 , C3 $\neq$ C4, C4 $\neq$ C5, C2 $\neq$ C4, C3 $\neq$ C5

2. 

3.

Variable	Domain
C1	C
C2	B
C3	A,C
C4	A,C
C5	B,C

Note that C5 cannot possibly be C, but arc consistency does not rule it out.

4. C1 = C, C2 = B, C3 = C, C4 = A, C5 = B. One other solution is possible (where C3 and C4 are switched).

---

5 Logic

Problem 5.1 (First-Order Resolution) 10 pt

Prove the following formula using resolution. 10 min

$$\forall X \forall Y \forall Z \exists U \exists V \exists W [(P(X,Y) \Rightarrow (P(Z,a) \Rightarrow R(a))) \Rightarrow ((P(U,V) \wedge P(W,a)) \Rightarrow R(a))]$$

We assume a signature with $P \in \Sigma_2^p$, $R \in \Sigma_1^p$, and $a \in \Sigma_0^f$.

Solution: Negated and in CNF:

$$\exists X \exists Y \exists Z \forall U \forall V \forall W [(\neg P(X,Y) \vee \neg P(Z,a) \vee R(a)) \wedge P(U,V) \wedge P(W,a) \wedge \neg R(a)]$$

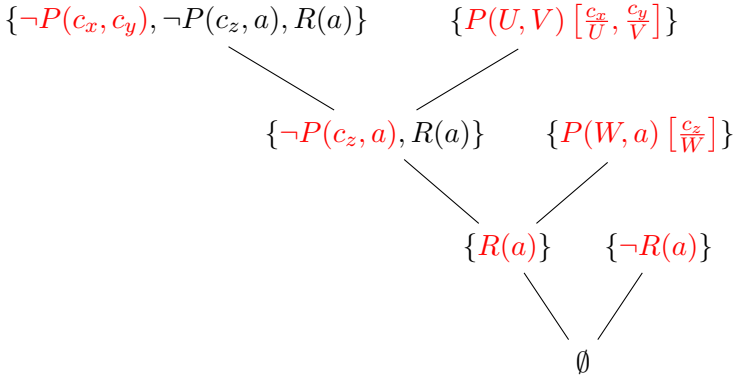
Skolemizing:

$$\forall U \forall V \forall W [(P(c_x,c_y) \Rightarrow (P(c_z,a) \Rightarrow R(a))) \Rightarrow ((P(U,V) \wedge P(W,a)) \Rightarrow R(a))]$$

We have the following clauses:

$$\{\neg P(c_x,c_y), \neg P(c_z,a), R(a)\}, \quad \{P(U,V)\}, \quad \{P(W,a)\}, \quad \{\neg R(a)\}$$

Now for the resolution, where in each step we need to unify the terms:



Problem 5.2 (Natural Deduction) 6 pt

Prove the following formula using the propositional Natural Deduction calculus. 6 min

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

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)

6 Planning

Problem 6.1 (STRIPS) 10 pt
10 min

You are given a water spout and two jugs, one holding p and one holding q gallons, where $p < q$ and p and q are relatively prime.

Starting with both jugs empty, the goal is to have exactly k gallons in one of the jugs. You can only fill the jugs from the spout fully.

We use the following predicates:

$$P = \{Jug_p(n) \mid 0 \leq n \leq p,\ n \in \mathbb{N}\} \cup \{Jug_q(n) \mid 0 \leq n \leq q,\ n \in \mathbb{N}\}$$

The intial state is $I = \{Jug_p(0), Jug_q(0)\}$ and the goal state $G = Jug_p(k)$.

Give the pre, add and del lists for the following actions:

- $Empty_p/Empty_q$: Empties jug p/q completely
- $FillUp_p/FillUp_q$: Fill up jug p/q fully
- For all x, y with $0 \leq x \leq p,\ 0 \leq y \leq q$:
  $Fillp_{x,y}/Fillq_{x,y}$: pour the contents of jug q/p into jug p/q until the former is empty or the latter is full.

Solution:

- $Empty_p$: $\mathsf{pre} = \{\}, \mathsf{add} = \{Jug_p(0)\}, \mathsf{del} = \{Jug_p(n) \mid 1 \leq n \leq p\}$
- $Empty_q$: $\mathsf{pre} = \{\}, \mathsf{add} = \{Jug_q(0)\}, \mathsf{del} = \{Jug_q(n) \mid 1 \leq n \leq q\}$
- $FillUp_p$: $\mathsf{pre} = \{\}, \mathsf{add} = \{Jug_p(p)\}, \mathsf{del} = \{Jug_p(n) \mid 0 \leq n < p\}$
- $FillUp_q$: $\mathsf{pre} = \{\}, \mathsf{add} = \{Jug_q(q)\}, \mathsf{del} = \{Jug_q(n) \mid 0 \leq n < q\}$
- For all x, y with $0 \leq x \leq p,\ 0 \leq y \leq q$, and with $m = \mathsf{min}(x + y, p)$:
  $Fillp_{x,y}$:

$$\mathsf{pre} = \{Jug_p(x), Jug_q(y)\},$$
$$\mathsf{add} = \{Jug_p(m), Jug_q(y - (m - x))\},$$
$$\mathsf{del} = \{Jug_p(z) \mid z \neq m\} \cup \{Jug_q(z) \mid z \neq m - x\}$$

  and with $m = \mathsf{min}(x + y, q)$:
  $Fillq_{x,y}$:

$$\mathsf{pre} = \{Jug_p(x), Jug_q(y)\},$$
$$\mathsf{add} = \{Jug_p(x - (m - y)), Jug_q(m)\},$$
$$\mathsf{del} = \{Jug_p(z) \mid z \neq m - y\} \cup \{Jug_q(z) \mid z \neq m\}$$
