Last Name: First Name:

Matriculation Number:

Seat:

Exam
Künstliche Intelligenz 1

Feb 12., 2018

	To be used for grading, do not write here				
prob.	1	2	Sum	grade	
total	12	10	80		
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 (The Zip Function) 4 pt

4 min

The zip function takes two lists with lengths that differ at most by 1, and outputs a list of lists containing one element from the first list and the element with the same index from the other list, possibly followed by a one-element list with the left-over argument.

Create a ProLog predicate with 3 arguments: the first two would be the two lists you want to zip, and the third one would be the result. For instance:

```
?- zip([1,2,3],[4,5,6],L).?- zip([1,2],[3,4,5],L).
L = [[1, 4], [2, 5], [3, 6]L = [[1, 3], [2, 4], [5]].
```

Feel free to implement any helper functions.

Hint: Remember that you can pattern match a list L as [HEAD|TAIL].

Solution:

```
zip([L],[],[[L]]).
zip([],[L],[[L]]).
zip([],[],[]).
zip([H1|T1],[H2|T2],L) :- zip(T1,T2,T), L = [[H1,H2]|T).
```

Problem 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 l
dfs(GoalValue,tree(Value,Rest),Path,FinalCost). % next child
```

2 Search

Problem 1 (Astar vs. Greedy) 5 pt

5 min

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.

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

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

3 Adversarial Search

Problem 1 (Minimax Restrictions) 3 pt

3 min

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

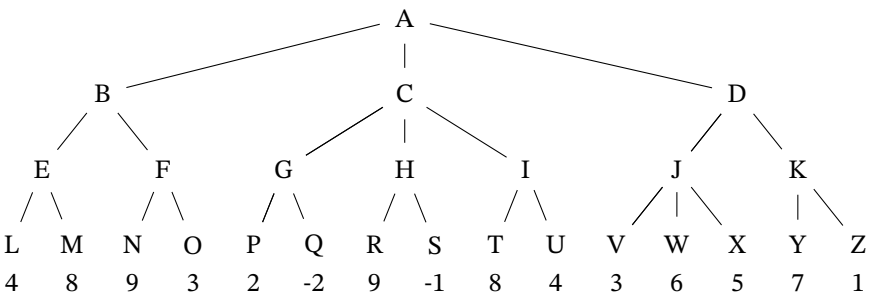
Solution: Any five of:

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

Problem 2 (Minimax Algorithm and Alphabeta Pruning) 10 pt

Consider the following (complete!) search tree:

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 1 (Constraint Networks) 4 pt

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. 4 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 2 (Constraint Networks) 10 pt

Describe the following problems as constraint networks 10 min

1. Sudoku
2. The 8 queens problem

Solution: Here are some possible answers:

Sudoku

- Variables: One variable $v_{i,j}$ for each coordinate (i, j) in the sudoku field.
- Domains: $\{1, ..., 9\}$ for each variable.
- Constraints: For each pair $v_{i,j}, v_{k,\ell}$ a binary constraint that is invalid iff
  - $i = k$ and $v_{i,j} = v_{k,\ell}$ (same row)
  - $j = \ell$ and $v_{i,j} = v_{k,\ell}$ (same column)
  - $v_{i,j}$ and $v_{k,\ell}$ are in the same block and $v_{i,j} = v_{k,\ell}$.

8-queens-problem

- Variables: $\{Q_1, ..., Q_8\}$ (where Q_i represents the queen in row i).
- Domains: $\{1, ..., 8\}$ (representing the columns).
- Constraints: For each pair Q_i, Q_j (with $i < j$) a binary constraint that is satisfied iff $Q_i \neq Q_j$ (no two queens in the same column) and $Q_j \neq Q_i + (j - i)$ and $Q_j \neq Q_i - (j - i)$ (not diagonal).

---

5 Logic

Problem 1 (Propositional Resolution) 5 pt 5 min

Prove the following formula using the resolution calculus:

$$\neg(\neg P \Rightarrow Q) \vee (Q \vee R) \vee \neg(P \Rightarrow R)$$

To convert the formula to CNF, you can use the extended rules for the transformation calculus from the lectures, for connectives $\vee$, $\neg$, $\Rightarrow$, and $\wedge$. It is ok to do a few steps at a time, but you should annotate each row with the rules you used. Name the rules after the connective and label at the assumption. For example, the rule

$$\frac{\mathbf{C} \vee (\mathbf{A} \vee \mathbf{B})^{\mathsf{T}}}{\mathbf{C} \vee \mathbf{A}^{\mathsf{T}} \vee \mathbf{B}^{\mathsf{T}}}$$

would be named $\vee^{\mathsf{T}}$.

Solution:

1. Transformation to CNF.

$$\begin{array}{c}
(\neg\neg P \Rightarrow Q \vee Q \vee R \vee \neg P \Rightarrow R)^{\mathsf{F}} \\
\hline
\neg(\neg P \Rightarrow Q)^{\mathsf{F}} ; Q^{\mathsf{F}} ; R^{\mathsf{F}} ; \neg(P \Rightarrow R)^{\mathsf{F}}, \text{apply } \vee^{\mathsf{F}} \text{ three times} \\
\hline
(\neg P \Rightarrow Q)^{\mathsf{T}} ; Q^{\mathsf{F}} ; R^{\mathsf{F}} ; (P \Rightarrow R)^{\mathsf{T}}, \text{apply } \neg^{\mathsf{F}} \text{ twice} \\
\hline
\neg P^{\mathsf{F}} \vee Q^{\mathsf{T}} ; Q^{\mathsf{F}} ; R^{\mathsf{F}} ; P^{\mathsf{F}} \vee R^{\mathsf{T}}, \text{apply} \Rightarrow^{\mathsf{T}} \text{ twice} \\
\hline
P^{\mathsf{T}} \vee Q^{\mathsf{T}} ; Q^{\mathsf{F}} ; R^{\mathsf{F}} ; P^{\mathsf{F}} \vee R^{\mathsf{T}}, \text{apply } \neg^{\mathsf{F}} \\
\hline
\end{array}$$

$CNF = \{P^{\mathsf{T}} \vee Q^{\mathsf{T}}, Q^{\mathsf{F}}, R^{\mathsf{F}}, P^{\mathsf{F}} \vee R^{\mathsf{T}}\}$

Problem 2 (First-Order Tableau) 10 pt 10 min

Prove the following formula using the first-order free variable tableaux calculus. We have $P \in \Sigma_2^p$, $f, g \in \Sigma_1^f$ and $a, b \in \Sigma_0^f$.

$$\exists X\Big(\big(\exists Y \neg P(Y, f(b))\big) \vee \big(\neg P(b, f(X)) \Rightarrow \neg P(g(a), f(X))\big)\Big)$$

Solution:

$$\exists X\,((\exists Y \neg P(Y, f(b))) \vee (\neg P(b, f(X)) \Rightarrow \neg P(g(a), f(X))))^F$$
$$(\exists Y \neg P(Y, f(b))) \vee (\neg P(b, f(V_X)) \Rightarrow \neg P(g(a), f(V_X)))^F$$
$$\exists Y \neg P(Y, f(b))^F$$
$$\neg P(b, f(V_X)) \Rightarrow \neg P(g(a), f(V_X))^F$$
$$\neg P(V_Y, f(b))^F$$
$$P(V_Y, f(b))^T$$
$$\neg(P(b, f(V_X)))^T$$
$$\neg P(g(a), f(V_X))^F$$
$$P(b, f(V_X))^F$$
$$P(g(a), f(V_X))^T$$
close using b/V_Y, b/V_X

6 Planning

Problem 1 12 pt

12 min

Cheeta is an intelligent and lazy monkey. Your task is to help him grab a banana.

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)
