Last Name: First Name:

Matriculation Number:

Seat:

Exam
Artificial Intelligence 1

Feb 11., 2019

	To be used for grading, do not write here				
prob.	1	2	Sum	grade	
total	12	8	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 4 pt 4 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.

If you have mastered addition and multiplication, feel free to try your hands on exponentiation as well.

Solution for=prolog-addition:

```
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 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:

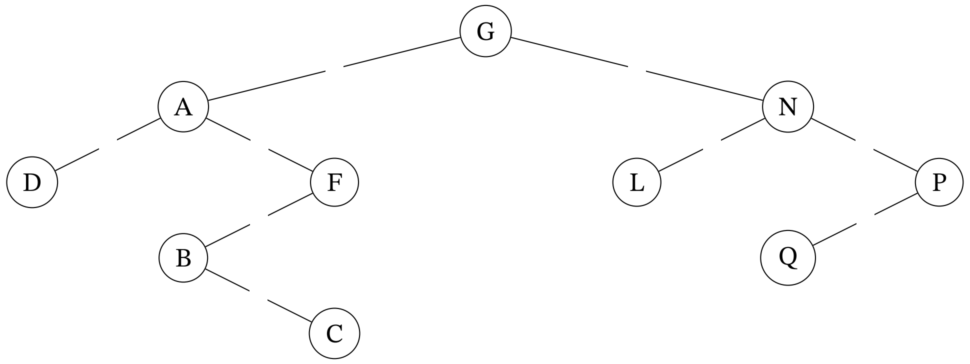
```
dfs(GoalValue,tree(GoalValue,_),GoalValue,0).
dfs(GoalValue,tree(Value,[(Cost,T)|Rest]),Path,FinalCost) :- T = tree(IV,_), write(IV ),
dfs(GoalValue, T,P,C),string_concat(Value,P,Path),FinalCost is C+Cost; % go down one depth l
dfs(GoalValue,tree(Value,Rest),Path,FinalCost). % next child
```

2 Search

Problem 1 (BFS) 3 pt

3 min

Apply BFS and DFS on the following tree exhaustively (the goal is not node G!).



List the nodes in the order they are expanded:

1. BFS
2. DFS

Solution:

1. BFS: G, A, N, D, F, L, P, B, Q, C
2. DFS: G, A, D, F, B, C, N, L, P, Q

Problem 2 (Admissibility limits) 5 pt

5 min

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

Solution: When $h(n) = 0$, the search will behave like an uninformed search, and when $h(n) = h^*(n)$ the search will only expand the nodes on the optimal path to a goal.

---

3 Adversarial Search

Problem 1 (Minimax Restrictions) 3 pt

3 min

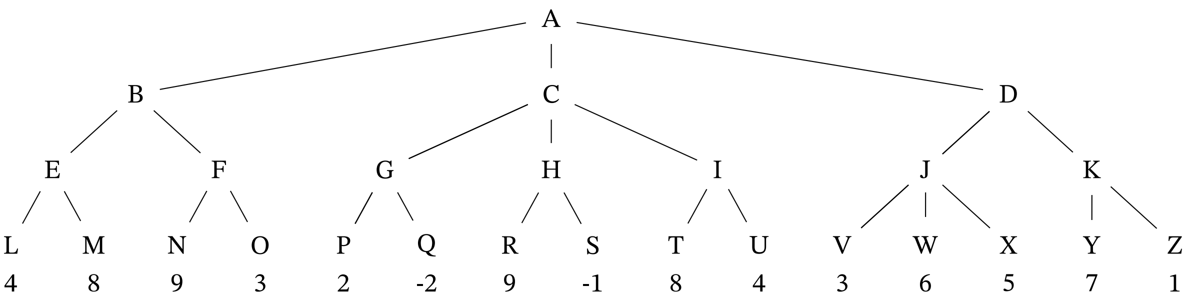
Name at least five criteria that a game has to satisfy in order for the *minimax algorithm* to be applicable.

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 (Game Tree) 7 pt 7 min

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. 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.

Submit your solution as a text file containing the following:

1. Line 1: 4 numbers, separated by space, corresponding to the nodes in alphabetical order, e.g., "1 3 2 5" means A=1, B=3, C=2, D=5.
2. Line 2: The upper-case letter for the selected move.
3. Line 3: The upper-case letters of the pruned moves.

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 1 (Constraint Networks) 6 pt

6 min

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.

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 (Arc Consistency for Chains) 10 pt

10 min

Consider the general less than chain below, which we interpret as a CSP: Each of the N variables X_i has the domain $\{1,\dots,M\}$. The constraints between adjacent variables X_i and X_{i+1} require that $X_i < X_{i+1}$.

$$X_1 < X_2 < X_3 < \cdots < X_N$$

1. For now, assume $N = M = 5$.

   (a) How many solutions does the CSP have?

   (b) What will the domain of X_1 be after enforcing the consistency of only the arc $X_1 \rightarrow X_2$?

   (c) What will the domain of X_1 be after enforcing the consistency of only the arcs $X_2 \rightarrow X_3$ and (then) $X_1 \rightarrow X_2$?

   (d) What will the domain of X_1 be after fully enforcing arc consistency?

2. Now consider the general case for arbitrary N and M.

   (a) Imagine you wish to construct a similar family of CSPs which forces one of the two following types of solutions: either all values must be ascending or all values must be descending, from left to right. For example, if $M = N = 3$, there would be exactly two solutions: $\{1,2,3\}$ and $\{3,2,1\}$. Explain how to formulate this variant. Your answer should include precise statements of variables and constraints.

Solution:

1. $N = M = 5$.

   (a) Just one: 1, 2, 3 …

   (b) $\{1,2,3,4\}$

   (c) $\{1,2,3\}$

   (d) $\{1\}$

2. (a) Several good answers. One is have ternary constraints on each adjacent triple that the triple should be either an increasing or decreasing triple. The overlap between triples enforces that the choice be global. Another is to introduce a global variable indicating ascent or descent and have ternary constraints between adjacent nodes and the global one, allowing, for example, triples like $\langle 1, 2, <\rangle$ or $\langle 2, 1, >\rangle$.

5 Logic

Problem 1 (First-Order Resolution) 10 pt 10 min

Prove the following formula using resolution.

$P, Q \in \Sigma_1^p; R \in \Sigma_2^p; c, d \in \Sigma_0^f$

$$\exists X.\forall Y.\exists W.\exists Z.\neg\,((R(Z,Y) \vee \neg P(Z)) \wedge (\neg Q(d) \vee P(c)) \wedge (Q(d) \vee \neg P(c))$$

$$\wedge(\neg R(Z,Y) \vee \neg P(W) \vee \neg Q(X)) \wedge P(c))$$

Hint: Note, that the formula is already (close to) a negated CNF, so if you spend any significant amount of time transforming the formula, you are most likely doing something wrong.

Solution: We negate:

$$\forall X.\exists Y.\forall W.\forall Z.\,((R(Z,Y) \vee \neg P(Z)) \wedge (\neg Q(d) \vee P(c)) \wedge (Q(d) \vee \neg P(c))$$

$$\wedge(\neg R(Z,Y) \vee \neg P(W) \vee \neg Q(X)) \wedge P(c))$$

Substituting bound variables:

$(R(Z,f_Y(X))\vee\neg P(Z))\wedge(\neg Q(d)\vee P(c))\wedge(Q(d)\vee\neg P(c))\wedge(\neg R(Z,f_Y(X))\vee\neg P(W)\vee\neg Q(X))\wedge P(c)$

Resolution:

$$\begin{aligned}
\{Q(d)^T, P(C)^F\} + \{P(c)^T\} &\Longrightarrow \{Q(d)^T\}\\
\{Q(d)^T\} + \{R(Z,f_Y(X))^F, P(W)^F, Q(X)^F\}[d/X] &\Longrightarrow \{R(Z,f_Y(d))^F, P(W)^F\}\\
\{P(c)^T\} + \{R(Z,f_Y(d))^F, P(W)^F\}[c/W] &\Longrightarrow \{R(Z,f_Y(d))^F\}\\
\{P(c)^T\} + \{R(Z,f_Y(X))^T, P(Z)^F\}[c/Z] &\Longrightarrow \{R(c,f_Y(X))^T\}\\
\{R(c,f_Y(X))^T\}[d/X] + \{R(Z,f_Y(d))^F\}[c/Z] &\Longrightarrow \{\}
\end{aligned}$$

Problem 2 (Natural Deduction) 8 pt 8 min

Let $R \in \Sigma_2^p, P \in \Sigma_1^p, c \in \Sigma_0^f$.

Prove the following formula in Natural Deduction:

$$((\forall X.\forall Y.R(Y,X) \Rightarrow P(Y)) \;\wedge\; (\exists Y.R(c,Y))) \Rightarrow P(c)$$

Solution:

1(Assumption)[1]	$(\forall X.\forall Y.R(Y,X) \Rightarrow P(Y)) \wedge (\exists Y.R(c,Y))$
2 $\wedge$ -Elimination on 1	$\forall X.\forall Y.R(Y,X) \Rightarrow P(Y)$
3 $\wedge$ -Elimination on 1	$\exists Y.R(c,Y)$
4$\forall$-Elimination on 2	$\forall Y.R(Y,X) \Rightarrow P(Y)$
5$\forall$-Elimination on 4	$R(c,X) \Rightarrow P(c)$
6$\forall$-Introduction on 5	$\forall X.R(c,X) \Rightarrow P(c)$
7(Assumption)[2]	$R(c,d)$
8$\forall$-Elimination on 6	$R(c,d) \Rightarrow P(c)$
9 $\Rightarrow$ -Elimination on 8, 7	$P(c)$
10$\exists$-Elimination[2] on 3, 9	$P(c)$
11 $\Rightarrow$ -Introduction[1] on 10	$((\forall X.\forall Y.R(Y,X) \Rightarrow P(Y)) \wedge (\exists Y.R(c,Y))) \Rightarrow P(c)$

---

6 Planning

Problem 1 (Planning Bike Repair) 12 pt

12 min

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.

Solution:

(a) Initial state description:

$$\begin{aligned} I \;=\; & \{At(Robot, A), At(Bicycle, B), At(Frontwheel, Bicycle), \\ & At(Backwheel, Bicycle), At(Box, C), At(Pump, D), Pushable(Box), \\ & Pushable(Pump), FlatTire(Frontwheel), FlatTire(Backwheel)\} \end{aligned}$$

Goal description: $G = \{Repaired(Bicycle)\}$

(b) Action descriptions:

- $Go(x, y) =$

$$\begin{aligned} pre &: \{At(Robot, x)\} \\ add &: \{At(Robot, y)\} \\ del &: \{At(Robot, x)\} \end{aligned}$$

for all $x, y \in \{A, B, C, D\}$.

- $Push(x, y, z) =$

$$\begin{aligned} pre &: \{At(Robot, y), At(x, y), Pushable(x)\} \\ add &: \{At(Robot, z), At(x, z)\} \\ del &: \{At(Robot, y), At(x, y)\} \end{aligned}$$

  for all $x \in \{Robot, Bicycle, Wheel, Box, Pump\}$, $y, z \in \{A, B, C, D\}$.

- $Remove(x, y) =$

$$\begin{aligned} pre &: \{At(Robot, x), At(Bicycle, x), At(y, Bicycle)\} \\ add &: \{At(y, x), Pushable(y)\} \\ del &: \{At(y, Bicycle)\} \end{aligned}$$

  for all $x \in \{A, B, C, D\}$, $y \in \{Frontwheel, Backwheel\}$.
  [Note that the facts Pushable(Frontwheel) and Pushable(Backwheel) can also be initially given and never deleted because of the way the action "Push" is defined.]

- $RepairWheel(x, y) =$

$$\begin{aligned} pre &: \{At(Robot, x), At(y, x), FlatTire(y), At(Box, x), At(Pump, x)\} \\ add &: \{Repaired(y)\} \\ del &: \{FlatTire(y)\} \end{aligned}$$

  for all $x \in \{A, B, C, D\}$, $y \in \{Frontwheel, Backwheel\}$.

- $InstallWheel(x, y) =$

$$\begin{aligned} pre &: \{At(Robot, x), At(Bicycle, x), At(y, x), Repaired(y), Pushable(y)\} \\ add &: \{At(y, Bicycle)\} \\ del &: \{At(y, x), Pushable(y)\} \end{aligned}$$

  for all $x \in \{A, B, C, D\}$, $y \in \{Frontwheel, Backwheel\}$.

- $FinalCheck(x) =$

$$\begin{aligned} pre &: \{At(Robot, x), At(Bicycle, x), At(Box, x), \\ &\quad At(Frontwheel, Bicycle), At(Backwheel, Bicycle), \\ &\quad Repaired(Frontwheel), Repaired(Backwheel)\} \\ add &: \{Repaired(Bicycle)\} \\ del &: \{\} \end{aligned}$$

  for all $x \in \{A, B, C, D\}$.