Name:

Birth Date:

Matriculation Number:

Retake Exam
Künstliche Intelligenz 1

Jul 24., 2017

	To be used for grading, do not write here										
prob.	1.1	1.2	2.1	2.2	3.1	4.1	4.2	4.3	5.1	Sum	grade
total	3	8	5	5	15	5	4	6	18	69	
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”.

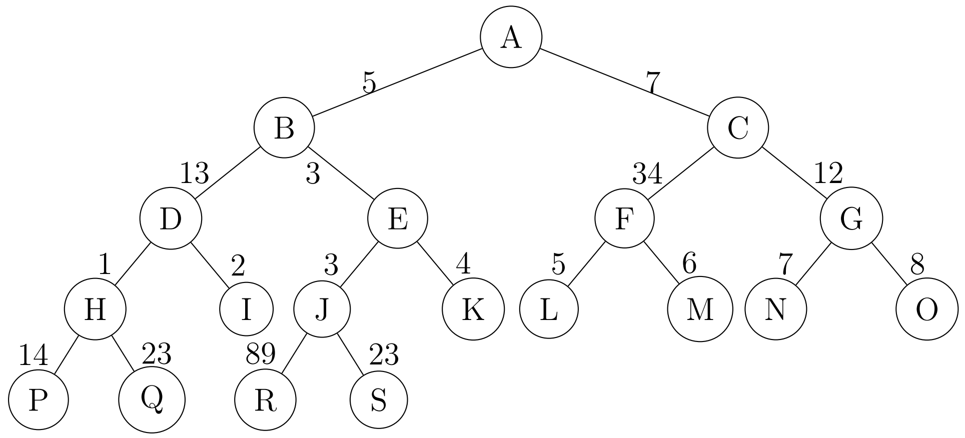
In the course Artificial Intelligence I/II we award 5 bonus points for the first student who reports a factual error (please report spelling/formatting errors as well) in an assignment or old exam and 10 bonus points for an alternative solution (formatted in LaTeX) that is usefully different from the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

1 Search

Problem 1.1 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 1.2 (Search Strings)
You are given a tree of the following form: 8 pt 8 min



None of the nodes in this tree are goal notes. In what order would the nodes be traversed for the following search types:

1. breadth-first search
2. depth-first search
3. uniform-cost search
4. greedy search, where you take the "alphabetic distance" (the number of letters) between the node label and the letter S as the heuristic.

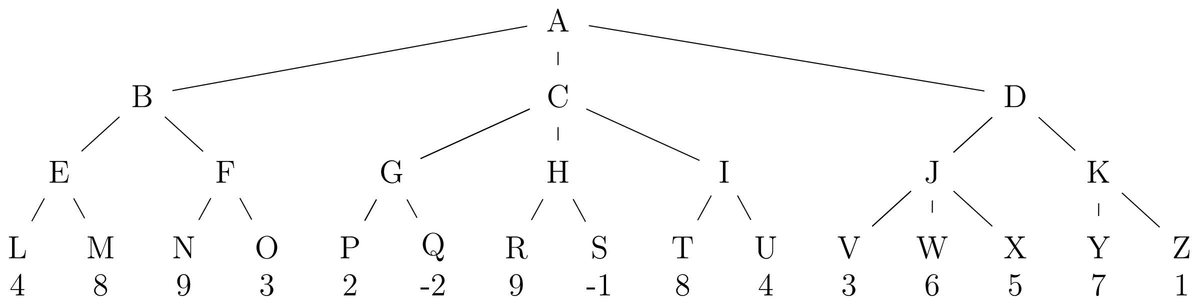
Give the solution as a simple string i.e. ABCDE

Solution:

1. ABCDEFGHIJKLMNOPQRS
2. ABDHPQIEJRSKCFLMGNO
3. ABCEJKDHGINOPSFQLMR
4. ACGONFMLBEKJSRDIHQP

2 Adversarial Search

Problem 2.1 (Game Tree)
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. 5 pt 5 min



20 pt

1. Label each non-leaf node with its minimax value. See above — 5 pt
2. Which move would be selected by Max? — 15 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. — 10 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
   (a) the minimax value computed at the root?
   (b) The number of nodes pruned by the alpha-beta algorithm?

Solution:

1. A:8, B:8, C:2, D:6, E:8, F:9, G:2, H:9, I:8, J:6, K:7
2. B
3. OHRSITUKYZ
4. (a) no, (b) yes

Problem 2.2 (Minimax Restrictions)

Name at least five criteria that a game has to satisfy in order for the minimax algorithm to be applicable. 5 pt

5 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

---

3 Constraint Satisfaction Problems & Inference

Problem 3.1 (Scheduling CS Classes)

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: 15 pt 15 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. 4 pt
2. Give the constraint graph associated with your CSP (e.g. by giving the edges). 2 pt
3. Show the domains of the variables after running arc-consistency on this initial graph (after having already enforced any unary constraints). 4 pt
4. Give one solution to this CSP. 1 pt
5. Your CSP should look nearly tree-structured. Briefly explain (one sentence or less) why we might prefer to solve tree-structures CSPs. 2 pt
6. Name (or briefly describe) a standard technique for turning these kinds of nearly tree-structured problems into tree-structured ones. 2 pt

---

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.

$$\begin{tikzcd}
C2 \arrow[r, dash] \arrow[dd, dash] \arrow[ddr, dash] & C4 \arrow[dd, dash] \arrow[dr, dash] & \\
 & & C5 \\
C1 & C3 \arrow[ur, dash] &
\end{tikzcd}$$

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. Minimal answer: Can solve them in polynomial time. If a graph is tree structured (i.e. has no loops), then the CSP can be solved in O(nd2) time as compared to general CSPs, where worst-case time is O(dn). For tree-structured CSPs you can choose an ordering such that every node's parent precedes it in the ordering. Then you can greedily assign the nodes in order and will find a consistent assignment without backtracking.

6. Minimal answer: cutset conditioning. One standard technique is to instantiate cutset, a variable (or set of variables) whose removal turns the problem into a tree structured CSP. To instantiate the cutset you set its values in each possible way, prune neighbors, then solve the reduced tree structured problem (which is fast).

---

4 Logic

Problem 4.1 (Propositional Tableaux)
Prove the following formula by exhibiting a closed $\mathcal{T}_0$ tableau. 5 pt

5 min

$$(\mathbf{A} \wedge \mathbf{B} \wedge \mathbf{C}) \Rightarrow (\mathbf{A} \Rightarrow \mathbf{B} \wedge \mathbf{C})$$

Problem 4.2 (Inference and Entailment)

4 pt

4 min

1. Define and briefly explain the entailment and inference relations.
2. How do we write "**A** entails **B**" and "from **A** we can infer/deduce **B**" in symbolism?
3. What is the difference between the two relations.

Solution:

1. The entailment relation $\models$ is defined via the models: We say $\mathbf{A} \models \mathbf{B}$, iff any model $\mathcal{M}$ that makes **A** true also makes **B** true. The inference relation $\vdash$ is defined by a set of purely syntactic inference rules.
2. "**A** entails **B**" is written as $\mathbf{A} \models \mathbf{B}$ and "from **A** we can infer/deduce **B**" as $\mathbf{A} \vdash \mathbf{B}$
3. Both are relations between expressions of the logic that try to model the process of argumentation from statements known to be true to new true statements. But entailment uses the notion of truth in a model, whereas the inference does this by a set of inference rules which are "known to preserve truth". In the best of all cases, the entailment and inference relations coincide. Then we have the "miracle of logics".

Problem 4.3 (Resolution Calculus)

Use the resolution calculus to prove the validity of the expression: 6 pt

3 min

$$(X \wedge Y) \vee (X \wedge \neg Y) \Rightarrow X$$

Solution:

5 Planning

Problem 5.1 (Planning Bike Repair)

18 pt

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

- $FinalCheck(x) =$

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

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