SAARLAND UNIVERSITY
Prof. Jörg Hoffmann
Dr. Álvaro Torralba, Daniel Gnad, Marcel Steinmetz
Jeanette-Aline Daum, Inken Hagestedt, Christoph Michel
Sebastian Laska, Cosmina Croitoru, Valerie Poser
Ghazale Harati Nezhad, Maximilian Schwenger

ARTIFICIAL INTELLIGENCE

First Exam
Artificial Intelligence
Summer Term 2015, July 28

Name: .....

E-mail: .....

Matr.Nr.: .....

- Write your name on every sheet of paper you use.
- Write your solutions in English.
- In the table below, mark each exercise for which you submit a solution.
- Attach all pages of your solution to these exercises.

Good luck!

Sign here:

.....

Exercise	1	2	3	4	5	6	Σ
submitted							
score							
max	18	22	10	16	12	22	100

Exercise 1: Propositional Logic: CNF and DPLL (18=6+5+7 points)

(a) Transform the following formula to CNF:

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

(b) For each of the following formulas use the DPLL procedure to determine whether ϕ is satisfiable or unsatisfiable. Give a complete trace of the algorithm, showing the simplified formula for each recursive call of the DPLL function. Assume that DPLL selects variables in alphabetical order (i.e., A, B, C, D, E), and that the splitting rule first attempts the value False (F) and then the value True (T).

(b1) $\phi_1 = (A \vee B) \wedge (C \vee D) \wedge (C \vee \neg D) \wedge (\neg B \vee \neg C) \wedge (\neg A \vee E) \wedge (\neg D \vee \neg E)$

(b2) $\phi_2 = (D \vee \neg C) \wedge (\neg A \vee \neg B \vee \neg E) \wedge (C \vee D) \wedge (\neg B \vee E \vee \neg D) \wedge (\neg A \vee C) \wedge (B \vee \neg D) \wedge (A \vee \neg E) \wedge (B \vee C)$

Exercise 2: PL1 Normal Forms and Resolution (22 = 4 + 9 + 9 points)

Nemo is a happy fish that lives in a fishbowl. He knows that the following set of PL1 formulas is true in his fishbowl:

(i) $\exists x\big[Fish(x) \wedge \forall y[\big(Fish(y) \wedge \neg\exists z[Shark(z)] \wedge \neg\exists z[Protects(z,y)]\big) \rightarrow Eats(x,y)]\big]$

(ii) $\neg\exists x\big[Fish(x) \wedge \forall y[Fish(y) \rightarrow Eats(x,y)]\big]$

(iii) $\neg\exists x\big[\exists y[Fish(y) \wedge Protects(x,y)]\big]$

(iv) $\forall x[Fish(x) \vee Shark(x)]$

Do each of the following:

(a) Explain what the formulas mean, in plain English. Note: the meaning of $Eats(x,y)$ is that x eats y and the meaning of $Protects(x,y)$ is that x protects y.

(b) Here are the formulas from above in Prenex normal form:

(i) $\exists x\forall y\exists z_1\exists z_2[Fish(x) \wedge (\neg Fish(y) \vee Shark(z_1) \vee Protects(z_2,y) \vee Eats(x,y))]$

(ii) $\forall x\exists y[\neg Fish(x) \vee (Fish(y) \wedge \neg Eats(x,y))]$

(iii) $\forall x\forall y[\neg Fish(y) \vee \neg Protects(x,y)]$

(iv) $\forall x[Fish(x) \vee Shark(x)]$

Bring these modified formulas into Skolem normal form and then into clausal normal form. For Skolem normal form, the function symbols must be unique *across all formulas*. Write the set Δ of PL1 clauses. Throughout, you may abbreviate "$Fish(.)$" as "$F(.)$", "$Shark(.)$" as "$S(.)$", "$Protects(.,.)$" as "$P(.,.)$", and "$Eats(.,.)$" as "$E(.,.)$".

(c) Oh no! You've seen a shark in the fishbowl, Nemo's life is in danger! However, Nemo won't believe you. Use PL1 resolution to show that there is a shark in Nemo's fish bowl, to convince him that he should escape.

Exercise 3: Default Reasoning (10 points)

Given the following Default-Theory $\Delta = (D, W)$:

$$
\begin{aligned}
W = \{ & \\
& \forall X \forall Y \; immediately(X) \wedge \neg equal(X, Y) \Rightarrow \neg immediately(Y), \\
& annoying(a_1), urgent(a_1), \\
& urgent(a_2), important(a_2), \\
& annoying(a_3), urgent(a_3), important(a_3), \\
& \neg equal(a_1, a_2), \neg equal(a_2, a_1), \neg equal(a_1, a_3), \\
& \neg equal(a_3, a_1), \neg equal(a_2, a_3), \neg equal(a_3, a_2) \\
\}
\end{aligned}
$$

$D = \{d_1, d_2, d_3, d_4, d_5, d_6\}$ where:

$$(d_1) \quad \frac{annoying(a_1) : \neg immediately(a_1)}{\neg immediately(a_1)}$$

$$(d_2) \quad \frac{annoying(a_1) \wedge urgent(a_1) : immediately(a_1)}{immediately(a_1)}$$

$$(d_3) \quad \frac{important(a_2) \wedge urgent(a_2) : immediately(a_2)}{immediately(a_2)}$$

$$(d_4) \quad \frac{annoying(a_3) : \neg immediately(a_3)}{\neg immediately(a_3)}$$

$$(d_5) \quad \frac{important(a_3) \wedge urgent(a_3) : immediately(a_3)}{immediately(a_3)}$$

$$(d_6) \quad \frac{annoying(a_3) \wedge urgent(a_3) : immediately(a_3)}{immediately(a_3)}$$

Determine all possible extensions E_i of Δ. For each extension, give the default rules d_j that were fired.

Exercise 4: Adversarial Search (16=8+8 points)

Consider the following two-player game. The players mark one empty field of the grid in turn. The MAX-player marks the fields with X, and the MIN-player marks the fields with O. The grid may contain blocked fields, denoted by #, which are fields that cannot be marked by X nor by O. An empty field is a field that is not blocked and which is not marked by either player. The terminal states are all states so that no empty field is left. Let r_X be the number of rows that contain two or more adjacent fields marked with X, and let c_X be the number of columns that contain two or more adjacent fields marked with X. Similarly, let r_O be the number of rows that contain two or more adjacent fields marked with O, and let c_O be the number of columns that contain two or more adjacent fields marked with O. The utility of a terminal state is defined as $r_X + c_X - (r_O + c_O)$.

Assume the following start state, where the MIN-player O performed the last move:

X		
O		O
#	O	X

(a) Perform Minimax search. Always prefer marking the left-most empty cell first. If there is more than one left-most empty cell, then prefer marking the top-most of those cells first. Draw the search tree, writing down the states in the notation as above, and connecting states by drawing edges from each state to its successor states. Annotate each state with the calculated Minimax value. Note: As the MIN-player performed the last move, the root node in the search tree is a MAX node (MAX chooses the move here).

(b) In the Minimax search tree from (a), indicate the branches (edges to a successor state) that would be pruned when using Alpha-Beta search instead.

Exercise 5: STRIPS Formulations (12 points)

Consider the following planning task. A truck (denoted **T**) has to transport packages (denoted P_x) to their goal locations. To do so, it can drive between any two connected locations and (un)load a package at its current position. However, the truck is only able to drive if a package is loaded (empty truck drives are not allowed). Additionally, only a single package can be loaded at the same time. In the initial state, the truck and package P_1 are at position A and the package P_2 is at position B; the goal is to bring P_2 to location C. Figure 1 illustrates the initial state as well as the connections between the locations.

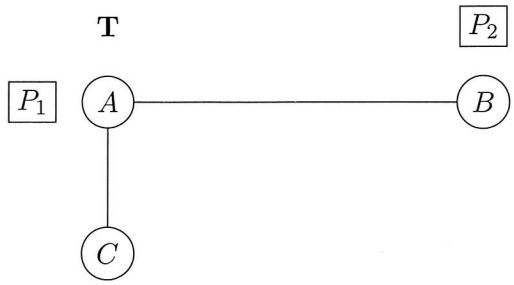

Figure 1: Map showing the initial state of the planning task.

In STRIPS, this task is formalized as follows:

$$
\begin{aligned}
I &= \{\mathrm{atT}(A), \mathrm{atP}(1, A), \mathrm{atP}(2, B), \mathit{empty}\} \\
G &= \{\mathrm{atP}(2, C)\} \\
A &= \{\mathit{drive}(x, y, i) \mid x, y \in \{A, B, C\}, i \in \{1, 2\}\} \\
&\cup \{\mathit{load}(i, x) \mid i \in \{1, 2\}, x \in \{A, B, C\}\} \\
&\cup \{\mathit{unload}(i, x) \mid i \in \{1, 2\}, x \in \{A, B, C\}\}
\end{aligned}
$$

$$
\begin{aligned}
\mathit{drive}(x, y, i) &= \mathit{pre} : \{\mathrm{atT}(x), \mathrm{atP}(i, T)\} \\
&\quad\ \mathit{add} : \{\mathrm{atT}(y)\} \\
&\quad\ \mathit{del} : \{\mathrm{atT}(x)\} \\
\mathit{load}(i, x) &= \mathit{pre} : \{\mathrm{atP}(i, x), \mathrm{atT}(x), \mathit{empty}\} \\
&\quad\ \mathit{add} : \{\mathrm{atP}(i, T)\} \\
&\quad\ \mathit{del} : \{\mathrm{atP}(i, x), \mathit{empty}\} \\
\mathit{unload}(i, x) &= \mathit{pre} : \{\mathrm{atP}(i, T), \mathrm{atT}(x)\} \\
&\quad\ \mathit{add} : \{\mathrm{atP}(i, x), \mathit{empty}\} \\
&\quad\ \mathit{del} : \{\mathrm{atP}(i, T)\}
\end{aligned}
$$

How can the task be modified to allow for *up to* two packages being loaded into the truck at the same time? If you need to add new facts to P or change existing ones, describe why you do that and what the new facts are used for. Write down the initial state and the new load/unload actions in STRIPS syntax.

Exercise 6: Delete Relaxation (22=5+17 points)

Consider again the example from Exercise 5, i.e. the STRIPS formalization as given above. (Recall, throughout this exercise, that actions in STRIPS are assumed to all have cost 1.)

(a) Give a shortest plan for the initial state if a plan exists. Give a shortest relaxed plan for the initial state if a relaxed plan exists. What are the values of $h^*(I)$ and $h^+(I)$?

(b) Run Greedy Best-First Search on the example using the h^+ heuristic. Draw the resulting search graph. Denote each search state by a tuple "XYZ", indicating that the truck is at position X, package P_1 is at Y and P_2 is at Z. The initial state, for example, is denoted "AAB". Annotate each state with its h^+ value, and with a number (1, 2, 3, ...) indicating the order of expansion. Indicate successor states by directed edges from the parent to the child. If you generate a duplicate state, mark it as such and ignore it for the rest of the search.

In case of a tie, i.e. if two or more states have the same minimal h^+ value, prefer states in which the truck is at C. If this still does not result in a unique state for expansion, you may break the remaining ties arbitrarily.

Note: You are only required to give the h^+ values, not the underlying relaxed plans, in your solution. However, we strongly recommend that you write up the relaxed plans as well, to avoid mistakes.