SAARLAND UNIVERSITY

ARTIFICIAL INTELLIGENCE

Prof. Jörg Hoffmann
Dr. Peter Kissmann, Álvaro Torralba
Jeanette Aline Daum, Inken Hagestedt, Jesko Hecking-Harbusch,
Sebastian Laska, Sreyasi Nag Chowdhury, Frank Nedwed

Second Exam
Artificial Intelligence
Summer Term 2014, October 13

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							100

Exercise ?: Constraint Satisfaction Problems (?=?+? points)

Consider the following constraint network $\gamma = (V, D, C)$:

- Variables: $V = \{A, B, C, D, E, F, G\}$.
- Domains: For all $v \in V$: $D_v = \{1, 2, 3, 4, 5\}$.
- Constraints: $A \geq B + 1$; $|A - C| \leq 1$; $D - C = 2$; $|D - E^2| \leq 2$; $D \geq 2F$; $F = 2G$.

Run the AcyclicCG(γ) algorithm. Precisely, execute its 4 steps as follows:

(a) Draw the constraint graph of γ. Pick A as the root and draw the directed tree obtained by step 1. Give a resulting variable order as can be obtained by step 2.

(b) List the calls to Revise($\gamma, v_{parent(i)}, v_i$) in the order executed by step 3, and for each of them give the resulting domain $D_{v_{parent(i)}}$ of $v_{parent(i)}$.

(c) For each recursive call of BacktrackingWithInference during step 4, give the domain D'_{v_i} of the selected variable v_i after Forward Checking, and give the value $d \in D'_{v_i}$ assigned to v_i. Note: Step 4 runs BacktrackingWithInference with variable order $v_1, \ldots, v_n$. This means that, at the ith recursion level, "select some variable v for which a is not defined" will select v_i.

(Solution)

(a) The constraint graph is shown in Figure ?? (i). The directed tree is shown in Figure ?? (ii). A possible variable order that can be obtained is A, B, C, D, E, F, G.

(i) (ii)

Figure 1: Solution to Exercise 1 (a).

(b) The calls to Revise($\gamma, v_{parent(i)}, v_i$) and reduced domains are:

- Revise(F, G); $D_F = \{2, 4\}$.
- Revise(D, F); $D_D = \{4, 5\}$.
- Revise(D, E); $D_D = \{4, 5\}$.
- Revise(C, D); $D_C = \{2, 3\}$.
- Revise(A, C); $D_A = \{1, 2, 3, 4\}$.
- Revise(A, B); $D_A = \{2, 3, 4\}$.

(c) Possible D'_{v_i} and $d \in D'_{v_i}$ are:

- $D'_A = \{2, 3, 4\}$; $d = 2$.
- $D'_B = \{1\}$; $d = 1$.
- $D'_C = \{2, 3\}$; $d = 2$.
- $D'_D = \{4\}$; $d = 4$.
- $D'_E = \{2\}$; $d = 2$.
- $D'_F = \{2\}$; $d = 2$.
- $D'_G = \{1\}$; $d = 1$.

(/Solution)

Exercise 3: Clause Learning (?=?+? points)

Consider the DPLL procedure with clause learning. For the outcome of the last call to UP in each of the settings (a) and (b) described below, draw the implication graphs (there are up to two possible implication graph in each of (a) and (b)). Draw all conflict graphs contained in the implication graphs. For each conflict graph, which is the clause that we can learn with the clause learning method presented in the lecture? (Note: Give *only* the information requested here; in particular do *not* run a complete trace of DPLL.)

(a) Consider the clause set $\Delta = \{\{\neg P, \neg Q\}, \{\neg P, R\}, \{P, \neg R\}, \{\neg P, Q, \neg R\}\}$.

Assume that the first call of the splitting rule chooses the proposition P and assigns it the value T. Then UP is run. This results in *two* possible implication graphs.

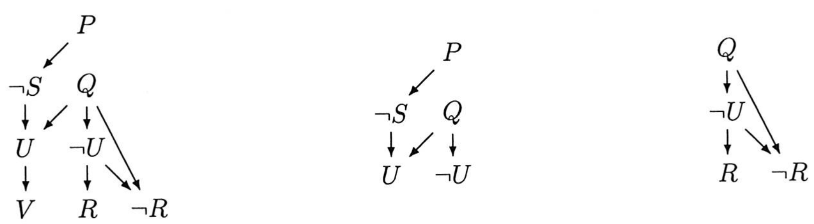
(b) Consider the clause set $\Delta = \{\{\neg P, \neg S\}, \{S, \neg Q, U\}, \{\neg U, V\}, \{\neg Q, \neg U\}, \{U, R\}, \{\neg Q, U, \neg R\}\}$.

Assume that the first call of the splitting rule chooses the proposition P and assigns it the value T; then UP is run; then the second call of the splitting rule chooses the proposition Q and assigns it the value T. Then UP is run again. This results in *one* possible implication graph.

(Solution)

(a) We get two implication graphs, shown in the following figure. The conflict graphs are the same as the implication graphs. The learned clause for each conflict graph is $\{\neg P\}$.

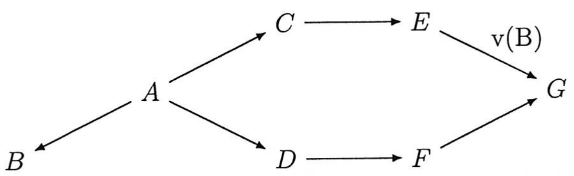
(b) The implication graph is shown in the left graph of the following figure. There are two conflict graphs contained in the implication graph, shown in the middle and the right of the figure. The learned clause for the conflict graph in the middle is $\{\neg P, \neg Q\}$. The learned clause for the conflict graph in the right is $\{\neg Q\}$.



(/Solution)

Exercise ??: Relaxed Planning Graphs (???=?+?? points)

Consider the following planning task. We have seven locations $\{A, B, C, D, E, F, G\}$. Initially, we are at A, and the goal is to have visited G. The connection between the locations is given by the following figure. Note that all connections are unidirectional, i.e., they can be traversed only in the specified direction.



Additionally, there is a door blocking the connection from E to G. This door is opened automatically when location B is visited.

The STRIPS formulation of this task is as follows:

- The facts are $P = \{at(x) \mid x \in \{A, B, C, D, E, F, G\}\} \cup \{v(x) \mid x \in \{A, B, C, D, E, F, G\}\}$, where $at(x)$ means that we are currently at location x and $v(x)$ means that we have already visited location x.
- The initial state is $I = \{at(A)\}$.
- The goal is $G = \{v(G)\}$.
- The actions are $A = \{go(A, B), go(A, C), go(A, D), go(C, E), go(D, F), go(E, G), go(F, G)\}$.

The actions all have uniform costs 1 and are specified as follows:

- $go(E, G) = \begin{array}{rl} \text{pre:} & \{at(E), v(B)\} \\ \text{add:} & \{at(G), v(G)\} \\ \text{del:} & \{at(E)\} \end{array}$
- for all $(x, y) \in \{(A, B), (A, C), (A, D), (C, E), (D, E), (F, G)\}$:
  $go(x, y) = \begin{array}{rl} \text{pre:} & \{at(x)\} \\ \text{add:} & \{at(y), v(y)\} \\ \text{del:} & \{at(x)\} \end{array}$

(a) Calculate the relaxed planning graph (RPG) for the specified task and include all intermediate steps. That is, give all fact layers F_i and all action layers A_i.

(b) Compute $h^{FF}(I)$, i.e., extract a relaxed plan from the calculated RPG and use it to determine the heuristic value. Start by marking all goal facts. While going back from the last to the first layer, in each step specify which actions are selected to support the marked facts of the current layer and which facts are newly marked in which layer for the selected supporting actions. *Assume that for supporting $v(G)$ the algorithm will select $go(E, G)$!* Do not forget to write down the retrieved relaxed plan and the value of $h^{FF}(I)$.

(c) Give an optimal relaxed plan for the initial state. What is $h^+(I)$?

(Solution)

(a) The layers of the RPG look as follows:

- $F_0 = \{at(A)\}$
- $A_0 = \{go(A, B), go(A, C), go(A, D)\}$
- $F_1 = F_0 \cup \{at(B), v(B), at(C), v(C), at(D), v(D)\}$
- $A_1 = A_0 \cup \{go(C, E), go(D, F)\}$
- $F_2 = F_1 \cup \{at(E), v(E), at(F), v(F)\}$
- $A_2 = A_1 \cup \{go(E, G), go(F, G)\}$
- $F_3 = F_2 \cup \{at(G), v(G)\}$

(b) Initially, only $v(G)$ in F_3 is marked.

F_3: only $v(G)$ is marked. To support it, action $go(E, G)$ is marked in A_2 (according to the supporter function mentioned in the exercise).

A_2: only $go(E, G)$ is marked. To support it, the facts $at(E)$ in F_2 and $v(B)$ in F_1 are marked.

F_2: only $at(E)$ is marked. To support it, action $go(C, E)$ is marked in A_1.

A_1: only $go(C, E)$ is marked. To support it, the fact $at(C)$ is marked in F_1.

F_1: the facts $at(C)$ and $v(B)$ are marked. To support the former, action $go(A, C)$ is marked in A_0. For the latter, action $go(A, B)$ is marked in A_0.

A_0: the actions $go(A, B)$ and $go(A, C)$ are marked. To support both, the fact $at(A)$ is marked in F_0.

F_0: This marked fact $at(A)$ is part of the current state. The search stops here.

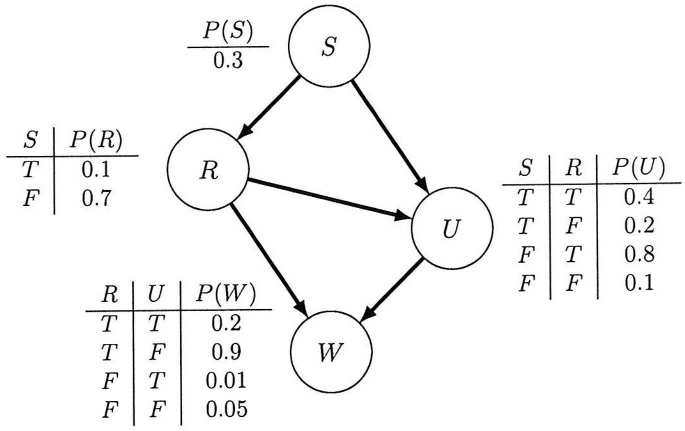
Thus, the marked actions are $go(A, C), go(A, B), go(C, E), go(E, G)$, giving us $h^{FF}(I) = 4$.

(c) An optimal relaxed plan is $\langle go(A, D), go(D, F), go(F, G)\rangle$, giving us $h^+(I) = 3$.

(/Solution)

Exercise ??: Bayesian Networks (???=?+?? points)

Consider the following Bayesian network BN:



$P(S)$
0.3

S	$P(R)$
T	0.1
F	0.7

S	R	$P(U)$
T	T	0.4
T	F	0.2
F	T	0.8
F	F	0.1

R	U	$P(W)$
T	T	0.2
T	F	0.9
F	T	0.01
F	F	0.05

There are four random variables, S, R, U, and W; all of them are Boolean. S models whether or not it is sunny, R if it is raining, U if you use an umbrella, and W if you get wet.

BN reflects the properties of the domain: It is less often sunny than it is cloudy. Chances of rain are much higher if it is not sunny. People tend to use umbrellas more often when it is raining, though some might also use it as protection of the sun. You get more wet if you are not using an umbrella; you can even get wet without rain, e.g., due to some water on the ground, though chances for that are slim.

In what follows, we use the notational conventions from the lecture: For Boolean variable *Name*, we write *name* for *Name* = *true* and ¬*name* for *Name* = *false*.

(a) Compute the probability of the atomic event where it is not sunny, but rainy, you use an umbrella and do not get wet. That is, calculate $P(\neg s, r, u, \neg w)$.

Include intermediate steps at a level of granularity as in the lecture slides examples. In particular, write down *which* probabilities provided in BN can be combined *how* to obtain $P(\neg s, r, u, \neg w)$.

(b) Use inference by enumeration to compute the probability of it being sunny given that you get wet and use an umbrella. That is, calculate $P(s \mid w, u)$.

Include intermediate steps at a level of granularity as in the lecture slides examples. In particular, state what the query variable, the evidence, and the hidden variables are; and write down *which* probabilities provided in BN can be combined *how* to obtain $P(s \mid w, u)$. Also do not forget to specify how α is calculated.

Note: In both (a) and (b), computing the final result requires a pocket calculator. If you do not have a pocket calculator, just omit that calculation. If you do have a pocket calculator, in (b) there is no need to write down all the decimal places, you may round to three decimal places; however, in (a) you must give all four decimals.

(Solution)

(a) By the Chain rule, $P(\neg s, r, u, \neg w) = P(\neg w \mid \neg s, r, u) \cdot P(u \mid \neg s, r) \cdot P(r \mid \neg s) \cdot P(\neg s)$.

By conditional independence, the latter is equal to $P(\neg w \mid r, u) \cdot P(u \mid \neg s, r) \cdot P(r \mid \neg s) \cdot P(\neg s)$.

Inserting the numbers provided by BN, we get $P(\neg s, r, u, \neg w) = 0.8 \cdot 0.8 \cdot 0.7 \cdot 0.7 = 0.3136$.

(b) The query variable X here is S. The evidence $\mathbf{e}$ is w, u. The hidden variables $\mathbf{Y}$ are $\{R\}$.

Using Normalization+Marginalization $\mathbf{P}(X \mid \mathbf{e}) = \alpha \sum_{\mathbf{y} \in \mathbf{Y}} \mathbf{P}(X, \mathbf{e}, \mathbf{y})$ we get $\mathbf{P}(S \mid w, u) = \alpha \sum_{\mathbf{y} \in \{R\}} \mathbf{P}(S, w, u, \mathbf{y}) = \alpha \sum_{v_R} \mathbf{P}(S, w, u, v_R)$.

By the Chain rule, $\mathbf{P}(S, w, u, v_R) = \mathbf{P}(w \mid v_R, S, u) \cdot \mathbf{P}(u \mid v_R, S) \cdot \mathbf{P}(v_R \mid S) \cdot \mathbf{P}(S)$.

By conditional independence, the latter is equal to $P(w \mid v_R, u) \cdot \mathbf{P}(u \mid v_R, S) \cdot \mathbf{P}(v_R \mid S) \cdot \mathbf{P}(S)$.

Putting this together, we get $\mathbf{P}(S \mid w, u) = \alpha \sum_{v_R} P(w \mid v_R, u) \cdot \mathbf{P}(u \mid v_R, S) \cdot \mathbf{P}(v_R \mid S) \cdot \mathbf{P}(S)$.

Moving variables outward, we get $\mathbf{P}(S \mid w, u) = \alpha \mathbf{P}(S) \cdot \sum_{v_R} P(w \mid v_R, u) \cdot \mathbf{P}(u \mid v_R, S) \cdot \mathbf{P}(v_R \mid S)$.

So, in total we get

$$\begin{aligned}
\mathbf{P}(S \mid w, u) &= \alpha \mathbf{P}(S) \cdot [P(w \mid r, u) \cdot \mathbf{P}(u \mid S, r) \cdot \mathbf{P}(r \mid S) \\
&\quad + P(w \mid \neg r, u) \cdot \mathbf{P}(u \mid S, \neg r) \cdot \mathbf{P}(\neg r \mid S) \\
&= \alpha \langle P(s) \cdot [P(w \mid r, u) \cdot P(U \mid s, r) \cdot P(r \mid s) \\
&\quad + P(w \mid \neg r, u) \cdot P(u \mid s, \neg r) \cdot P(\neg r \mid s)], \\
&\quad P(\neg s) \cdot [P(w \mid r, u) \cdot P(u \mid \neg s, r) \cdot P(r \mid \neg s) \\
&\quad + P(w \mid \neg r, u) \cdot P(u \mid \neg s, \neg r) \cdot P(\neg r \mid \neg s)] \rangle
\end{aligned}$$

Inserting the numbers provided by BN, we get

$$\begin{aligned}
\mathbf{P}(S \mid b) &= \alpha \langle 0.3 \cdot [0.2 \cdot 0.4 \cdot 0.1 + 0.01 \cdot 0.2 \cdot 0.9], \\
&\quad 0.7 \cdot [0.2 \cdot 0.8 \cdot 0.7 + 0.01 \cdot 0.1 \cdot 0.3] \rangle \\
&= \alpha \langle 0.00294, 0.07861 \rangle
\end{aligned}$$

Hence $\alpha = \frac{1}{0.00294 + 0.07861} = \frac{1}{0.08155} \approx 12.26$ and we get $\mathbf{P}(S \mid w, u) \approx \langle 0.036, 0.964 \rangle$. In particular, $P(s \mid w, u) \approx 0.036$. In other words, if we get wet and use an umbrella the chance that it is sunny is only 3.6%.

(/Solution)

Exercise ?: Herbrand Expansion and Propositional Resolution (? points)

Consider the set θ of Skolem normal form PL1 formulas:

(i) All students go to the university by car.

$$\forall x[\neg Student(x) \vee GoByCar(x)]$$

(ii) Every person that goes by car, needs to drive a car.

$$\forall x[\neg GoByCar(x) \vee (Car(f(x)) \wedge Drives(x, f(x))]$$

(iii) Any person needs a license to drive a car.

$$\forall x, y[\neg Drives(x, y) \vee \neg Car(y) \vee License(x)]$$

(iv) Bob is a student that does not have a license.

$$Student(Bob) \wedge \neg License(Bob)$$

In what follows, you may abbreviate the predicate names, i.e., instead of $Student(x)$ use $S(x)$, instead of $GoByCar(x)$ use $GBC(x)$, instead of $Car(x)$ use $C(x)$, instead of $Drives(x, y)$ use $D(x, y)$, and instead of $License(x)$ use $L(x)$.

(a) Write up the set of constants and functions occurring in θ, $CF(\theta)$.

(b) Write the Herbrand Universe $HU(\theta)$ and the Herbrand Expansion $HE(\theta)$. If some of them is an infinite set, specify its contents using "..." notation and for $HU(\theta)$ give at least 3 examples (using all constants and functions at least once), and for $HE(\theta)$ give at least 4 examples (using all constants and functions and formulas at least once).

(c) Bring the formulas in $HE(\theta)$ into CNF, resulting in a set Δ of clauses. Then use propositional resolution to prove that (a finite subset of) Δ is unsatisfiable. Again, use "..." notation in an infinite case, but in this case your examples can be only the subset of clauses needed for proving unsatisfiability (Hint: strictly less than 10 clauses are needed).

(Solution)

(a) $CF(\theta) = \{Bob, f\}$.

(b) The Hebrand Universe is an infinite set of ground terms:

$$HU(\theta^*) = \{Bob, f(Bob), f(f(Bob)), \dots\}$$

The Hebrand Expansion is an infinite set of propositional formulas:

$$
\begin{aligned}
HE(\theta^*) = \{ \quad & \neg S(Bob) \vee GBC(Bob), \\
& \ldots, \\
& \neg GBC(Bob) \vee (C(f(Bob)) \wedge D(Bob, f(Bob))], \\
& \neg GBC(f(Bob)) \vee (C(f(f(Bob))) \wedge D(f(Bob), f(f(Bob)))], \\
& \ldots \\
& \neg D(Bob, f(Bob)) \vee \neg C(f(Bob)) \vee L(Bob) \\
& \neg D(f(Bob), Bob) \vee \neg C(Bob) \vee L(f(Bob)) \\
& \ldots \\
& S(Bob) \wedge \neg L(Bob)\}
\end{aligned}
$$

(c) The set Δ is:

$$
\begin{aligned}
\Delta = \{ \quad & \{\neg S(Bob), GBC(Bob)\} \\
& \{\neg GBC(Bob), C(f(Bob))\} \\
& \{\neg GBC(Bob), D(Bob, f(Bob))\} \\
& \{\neg D(Bob, f(Bob)), \neg C(f(Bob)), L(Bob)\} \\
& \{S(Bob)\} \\
& \{\neg L(Bob)\} \\
\ldots\}
\end{aligned}
$$

(d) Resolution Tree:

$$
\begin{tikzcd}
 & \{S(Bob)\} \arrow[dr] & & \{\neg S(Bob), GBC(Bob)\} \arrow[dl] & & \\
\begin{array}{c}\{\neg GBC(Bob), \\ C(f(Bob))\}\end{array} \arrow[dr] & & \{GBC(Bob)\} \arrow[dl] \arrow[dr] & & \begin{array}{c}\{\neg GBC(Bob), \\ D(Bob, f(Bob))\}\end{array} \arrow[dl] & \\
 & \{C(f(Bob))\} \arrow[ddr] & & \{D(Bob, f(Bob))\} \arrow[d] & & \begin{array}{c}\{\neg D(Bob, f(Bob)), \\ \neg C(f(Bob)), \\ L(Bob)\}\end{array} \arrow[dll] \\
 & & & \{\neg C(f(Bob)), L(Bob)\} \arrow[dl] & & \\
 & & \{L(Bob)\} \arrow[dr] & & \{\neg L(Bob)\} \arrow[dl] & \\
 & & & \square & &
\end{tikzcd}
$$

(/Solution)

Exercise ?: Admissible Heuristics (? points)

Consider a Mars rover that has to drive around the surface, to go to different types of interesting places, as shown in the example below. The possible actions consist of moving the rover to an adjacent location, either horizontally or vertically.

y_3			t_1		R	
y_2	t_2					
y_1	t_2					t_1
	x_1	x_2	x_3	x_4	x_5	x_6

There are two types of interesting places, that we denote by t_1 and t_2, respectively. The goal is to find a plan that makes the rover visit at least one location of each type. An optimal plan for the example is moving to the left four times and then moving down. The state can be described as a tuple $\langle R, T\rangle$, where $R = (x_r, y_r)$ is the current position of the rover and $T = \{t_1, t_2\} \setminus T_{visited}$ is the set of types that remain to be visited. Thus, a state is a goal iff T is empty.

For each type of place t, we denote as P_t the set of positions (coordinates) of a given type. In the example, $P_{t_1} = \{(x_3, y_3), (x_6, y_1)\}$ and $P_{t_2} = \{(x_1, y_2), (x_1, y_1)\}$.

The heuristics make use of the Manhattan Distance between two positions, which is defined as follows: $MD((x_i, y_j), (x_k, y_l)) = |i - k| + |j - l|$

For each of the following heuristic functions determine whether it is admissible or not. Take into account that the state could be any in which the position of the Rover or the types are different from the example. In case that the heuristic is admissible give a compelling argument of why it must be admissible for any state. In case that the heuristic is inadmissible provide a counterexample. *Note that for all heuristics, we additionally have $h(s) = 0$ if $s = \langle R, \emptyset\rangle$.*

(a) The Manhattan Distance to the closest position of the missing type that is farthest away from the rover. $h_1(\langle R, T\rangle) = \max_{t \in T} \min_{p \in P_t} MD(R, p)$

(b) The Manhattan Distance to the farthest away position of the missing type that is closest to the rover. $h_2(\langle R, T\rangle) = \min_{t \in T} \max_{p \in P_t} MD(R, p)$

(c) The sum of the Manhattan Distances to the closest position of each missing type.

$$h_3(\langle R, T\rangle) = \sum_{t \in T} \min_{p \in P_t} MD(R, p)$$

(d) The Manhattan Distance to the closest position of any missing type plus the minimum of the Manhattan Distances between a position of type t_1 and another of type t_2, whenever neither of the types have been visited yet.

$$h_4(\langle R, T\rangle) = \min_{t \in T} \min_{p \in P_t} MD(R, p) + \begin{cases} \min_{p_1 \in P_{t_1}, p_2 \in P_{t_2}} MD(p_1, p_2) & T = \{t_1, t_2\} \\ 0 & \text{otherwise} \end{cases}$$

(Solution)

(a) h_1 is admissible. The rover needs to visit at least one position of each type. For each type, we cannot do any better than going to the one closest to R. As we need to visit all types in T, the maximum is admissible. Furthermore, the Manhattan Distance is an admissible estimate of the real distance, so that h_1 will never overestimate the cost to reach the goal.

(b) h_2 is inadmissible. Consider the following example state s.

y_2	R	t_1		t_2
y_1		t_2		t_1
	x_1	x_2	x_3	x_4

For this state, $h_3(s) = min(3, 4) = 3$, while $h^*(s) = 2$.

(c) h_3 is inadmissible. Consider the following example state s.

y_1	R	t_1	t_2
	x_1	x_2	x_3

For this state, $h_4(s) = 1 + 2 = 3$, while $h^*(s) = 2$.

(d) h_4 is admissible. In case that only one type remains h_4 is equal to h_1, so it will also be admissible. In case that both types of positions must be visited, the solution path will necessarily include reaching positions of both types, such that the path to the goal can be divided into two parts: reaching the first position and going from the first to the second position. Going to the closest position of any time is an admissible estimate for reaching the first position. The shortest distance between two positions of any type is an admissible estimate for the distance between the two positions. As the two parts of the path are independent, the estimations can be admissibly added. Additionally, the Manhattan Distance is an admissible estimate of the real distance, so that h_4 will never overestimate the cost to reach the goal.

(/Solution)