

Course title: Introduction to Artificial Intelligence

Course code: **MAT1206E**

Credits: **3**

Exam No.: **02**

For students of course class: **MAT1206E 3**

Time allowed: **90 minutes** (not including the time for handing out the paper)

Question 1. (1 point) An agent with memory moves in a plane. At clock ticks of a regular interval $\Delta t = 0.5$ s, its sensors give its exact position (x, y) in meters. The last three measurements are $(2, 1)$ at time $t - 2\Delta t$, $(3, 3)$ at time $t - \Delta t$ and $(5, 6)$ at time t .

- (a) Why is a deterministic agent with memory not a function from the set of all inputs to the set of all outputs, in the mathematical sense? How can the agent be modelled so that it becomes a function without losing its memory?
- (b) Estimate the components $v_x(t)$ and $v_y(t)$ of the velocity of the agent at time t from the current and the previous measurement, and estimate the components $a_x(t)$ and $a_y(t)$ of its acceleration at time t . How many earlier positions must the agent keep in its memory for each of the two computations?

Question 2. (2.5 points)

- (a) A greenhouse controller works with the following knowledge base KB : (1) If the soil is dry, then the pump is switched on or the gardener is notified. (2) The pump is switched on if and only if the water tank is not empty. (3) The water tank is empty. (4) The soil is dry. Use the propositional variables D (the soil is dry), P (the pump is switched on), N (the gardener is notified) and E (the water tank is empty).
 - (i) Formalize KB and transform it into conjunctive normal form (CNF). Number the clauses.
 - (ii) Prove $KB \models N$ with the resolution calculus. Write every resolution step as $\text{Res}(m, n) : (\text{resolvent})_k$, where the resolvent of clauses m and n is numbered k .
- (b) Consider the following statements: (1) For every rabbit, there is a plant that the rabbit eats. (2) Anything that eats at least one plant is a plant eater. (3) Roger is a rabbit. Use the predicates $rabbit(x)$ (x is a rabbit), $plant(x)$ (x is a plant), $eats(x, y)$ (x eats y), $plant_eater(x)$ (x is a plant eater) and the constant $roger$ (Roger).
 - (i) Formalize the statements in first-order predicate logic and transform them into conjunctive normal form (CNF) for use in the resolution proof in part (ii). Number the clauses. If a Skolem function is needed, call it s .
Recall that resolution is applied to clauses whose variables are implicitly universally quantified. Skolemization eliminates existential quantifiers while preserving satisfiability: the original formula is satisfiable if and only if the Skolemized formula is satisfiable. However, the two formulas need not be logically equivalent.
 - (ii) Prove by resolution that Roger is a plant eater. For every resolution step, give the two parent clauses, the most general unifier (MGU) and the resolvent.

Question 3. (0.5 points) The knowledge base KB in first-order predicate logic consists of the formulas $kiwi(kai)$, $\forall x (kiwi(x) \Rightarrow bird(x))$ and $\forall x (bird(x) \Rightarrow fly(x))$. Here kai is a constant (a kiwi called Kai), and $kiwi(x)$, $bird(x)$ and $fly(x)$ mean that x is a kiwi, that x is a bird and that x can fly.

- Suppose that $\forall x (kiwi(x) \Rightarrow \neg fly(x))$ is added to KB . Can $fly(kai)$ be derived from the extended knowledge base, and can $\neg fly(kai)$? What does this mean for the extended knowledge base?
- Which property of first-order predicate logic explains why adding a formula to KB cannot prevent the derivation of $fly(kai)$? State this property.

Question 4. (2 points) Consider the following PROLOG program with numbered clauses, where $child(X, Y)$ means that X is a child of Y and $descendant(X, Y)$ that X is a descendant of Y .

```

1 descendant(X,Y) :- child(X,Z), descendant(Z,Y).
2 descendant(X,Y) :- child(X,Y).
3 child(anna,bea).
4 child(bea,carla).

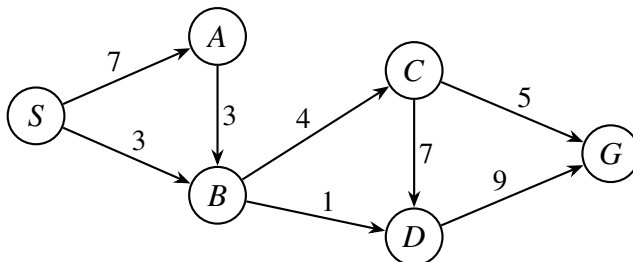
```

- Which clause of $descendant$ is the base clause and which is the recursive clause? State the declarative meaning of clause 1.
- Draw the search tree for the query $?- descendant(anna, Y)$. Each box contains the list of remaining goals, each edge is labelled with the number of the clause tried for the selected goal (a clause of the same predicate whose head does not unify leads to a fail leaf), and each leaf is $[]$ (with the answer) or fail. Give the answers in the order in which PROLOG returns them, and explain this order.
- Clause 1 is replaced by the following clause, which has the same declarative meaning:

1' $descendant(X, Y) :- descendant(Z, Y), child(X, Z)$.

What happens when the query of (b) is asked again? Explain why.

Question 5. (2 points) In the search problem below, S is the starting state, G is the only goal state, the arrows are the only actions (labelled with their costs), and the table gives the admissible heuristic cost estimate h .



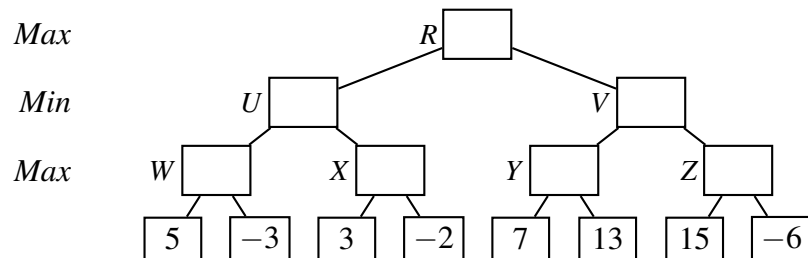
state s	S	A	B	C	D	G
$h(s)$	8	8	6	5	6	0

- Apply IDA* search with the heuristic evaluation function $f(n) = g(n) + h(n)$, where $g(n)$ is the cost of the path from the root to n . Generate the successors of each node in alphabetical order. Write each search node as its path from S , for example $S-A$. Draw and complete the table below, one row per iteration. Give cut-off nodes' f -values in parentheses; in the last column, write the next threshold or the goal node reached in the final iteration.

θ	Expanded nodes, in order	Cut off (f -value)	Next threshold
$\vdots$			

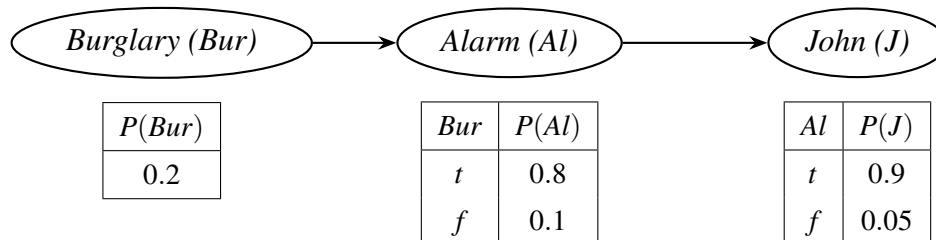
Which path is found, and what is its cost?

- (ii) Which nodes are expanded in more than one iteration? In exchange for these repeated expansions, which weakness of A* does IDA* avoid?
- (b) In the game tree below, Max moves at the root R ; the ratings of the leaves are given from Max's point of view. Compute the minimax value of every inner node. Which move does Max choose at R ?



Question 6. (2 points)

- (a) The Bayesian network below has the binary variables Bur (a burglary happens), Al (the alarm sounds) and J (John calls), each with the values t and f . Throughout this question, $P(X)$ abbreviates $P(X = t)$ and $\neg X$ stands for $X = f$; for example, the table of J gives $P(J | Al) = 0.9$ and $P(J | \neg Al) = 0.05$.



- (i) Write $P(Bur, Al, J)$ as a product of probabilities given by the network.
- (ii) Compute $P(Al)$ and $P(Bur | Al)$.
- (iii) Compute $P(J | Bur)$ and $P(J)$. Are J and Bur independent? Are they conditionally independent given Al ? Justify your answers.
- (b) For two binary variables A and B , only $P(A) = 0.6$ and $P(B | A) = 0.7$ are known. Let p_1, p_2, p_3, p_4 denote $P(A, B), P(A, \neg B), P(\neg A, B), P(\neg A, \neg B)$, respectively.
- (i) Write the given knowledge and the normalization condition as equations in $p_1, \dots, p_4$.
- (ii) Determine the distribution (p_1, p_2, p_3, p_4) selected by the principle of maximum entropy (MaxEnt). Justify the values that are not fixed by the equations.
- (iii) Compute $P(B)$ and $P(B | \neg A)$ for this distribution.

VIETNAM NATIONAL UNIVERSITY, HANOI
VNU UNIVERSITY OF SCIENCE

ANSWER KEY AND MARKING SCHEME
MIDTERM EXAMINATION, SEMESTER I, ACADEMIC YEAR 2026–2027
Course title: Introduction to Artificial Intelligence

Course code: **MAT1206E**

Credits: **3**

Exam No.: **02**

For students of course class: **MAT1206E 3**

Solution 1.

[1 point]

(a) The output of an agent with memory depends not only on the current input but also on the contents of its memory. For the same input, different memory contents can lead to different outputs, so the output is not uniquely determined by the input, and the agent is not a function.	0.25
If the contents of the memory are considered as a further input, then the output is unique, because the agent is deterministic, and the agent becomes a function.	0.25
(b) Velocity: $v_x(t) \approx \frac{x(t) - x(t - \Delta t)}{\Delta t} = \frac{5 - 3}{0.5} = 4$ and $v_y(t) \approx \frac{6 - 3}{0.5} = 6$, both in m/s; one earlier position is needed.	0.25
Acceleration: $a_x(t) \approx \frac{x(t) - 2x(t - \Delta t) + x(t - 2\Delta t)}{(\Delta t)^2} = \frac{5 - 6 + 2}{0.25} = 4$ and $a_y(t) \approx \frac{6 - 6 + 1}{0.25} = 4$, both in m/s ² ; two earlier positions are needed, at $t - 2\Delta t$ and $t - \Delta t$, besides the current one at t . Keeping one earlier position and the previous velocity components $v_x(t - \Delta t)$ and $v_y(t - \Delta t)$ instead is also accepted.	0.25

Solution 2.

[2.5 points]

(a)(i) Formalization: $KB \equiv (D \Rightarrow P \vee N) \wedge (P \Leftrightarrow \neg E) \wedge E \wedge D$.	0.25
Eliminating the implication and the equivalence gives $D \Rightarrow P \vee N \equiv \neg D \vee P \vee N$ and $P \Leftrightarrow \neg E \equiv (P \Rightarrow \neg E) \wedge (\neg E \Rightarrow P) \equiv (\neg P \vee \neg E) \wedge (\neg \neg E \vee P)$, and the double negation gives $(\neg P \vee \neg E) \wedge (E \vee P)$. Hence $KB \equiv (\neg D \vee P \vee N)_1 \wedge (\neg P \vee \neg E)_2 \wedge (E \vee P)_3 \wedge (E)_4 \wedge (D)_5$	0.25
(a)(ii) Add the negated query $(\neg N)_6$ to KB . One possible proof: Res(1,5): $(P \vee N)_7$, Res(6,7): $(P)_8$, Res(2,8): $(\neg E)_9$, Res(4,9): $()_{10}$. The empty clause shows that $KB \wedge \neg N$ is unsatisfiable; therefore $KB \models N$.	0.5
(b)(i) Formalization of the three statements: $\forall x (rabbit(x) \Rightarrow \exists y (plant(y) \wedge eats(x, y))), \quad \forall x \forall y (eats(x, y) \wedge plant(y) \Rightarrow plant_eater(x)),$ $rabbit(roger).$ The equivalent form $\forall x (\exists y (eats(x, y) \wedge plant(y)) \Rightarrow plant_eater(x))$ of the second statement is also accepted.	0.25

CNF of the first statement: eliminating the implication and moving $\exists y$ to the front gives the prenex normal form $\forall x \exists y \left(\neg rabbit(x) \vee (plant(y) \wedge eats(x,y)) \right)$, and the distributive law gives $\forall x \exists y \left((\neg rabbit(x) \vee plant(y)) \wedge (\neg rabbit(x) \vee eats(x,y)) \right).$ The variable y depends on x, so Skolemization replaces y by $s(x)$; then the universal quantifier is deleted. With the other two statements, the clauses are $(\neg rabbit(x) \vee plant(s(x)))_1, \quad (\neg rabbit(x) \vee eats(x,s(x)))_2,$ $(\neg eats(x,y) \vee \neg plant(y) \vee plant_eater(x))_3, \quad (rabbit(roger))_4.$	0.5																		
(b)(ii) The query is $Q \equiv plant_eater(roger)$; add the negated query $(\neg plant_eater(roger))_5$. One possible proof (the parent clauses never share variables): <table><thead><tr><th>Step</th><th>Resolvent</th><th>Proved by</th></tr></thead><tbody><tr><td>6.</td><td>$\neg eats(roger,y) \vee \neg plant(y)$</td><td>Res(3,5), MGU $\sigma: x/roger$</td></tr><tr><td>7.</td><td>$eats(roger,s(roger))$</td><td>Res(2,4), MGU $\sigma: x/roger$</td></tr><tr><td>8.</td><td>$\neg plant(s(roger))$</td><td>Res(6,7), MGU $\sigma: y/s(roger)$</td></tr><tr><td>9.</td><td>$plant(s(roger))$</td><td>Res(1,4), MGU $\sigma: x/roger$</td></tr><tr><td>10.</td><td>$()$</td><td>Res(8,9), MGU ε (the empty substitution)</td></tr></tbody></table> The empty clause is derived, so $plant_eater(roger)$ follows: Roger is a plant eater.	Step	Resolvent	Proved by	6.	$\neg eats(roger,y) \vee \neg plant(y)$	Res(3,5), MGU $\sigma: x/roger$	7.	$eats(roger,s(roger))$	Res(2,4), MGU $\sigma: x/roger$	8.	$\neg plant(s(roger))$	Res(6,7), MGU $\sigma: y/s(roger)$	9.	$plant(s(roger))$	Res(1,4), MGU $\sigma: x/roger$	10.	$()$	Res(8,9), MGU ε (the empty substitution)	0.75
Step	Resolvent	Proved by																	
6.	$\neg eats(roger,y) \vee \neg plant(y)$	Res(3,5), MGU $\sigma: x/roger$																	
7.	$eats(roger,s(roger))$	Res(2,4), MGU $\sigma: x/roger$																	
8.	$\neg plant(s(roger))$	Res(6,7), MGU $\sigma: y/s(roger)$																	
9.	$plant(s(roger))$	Res(1,4), MGU $\sigma: x/roger$																	
10.	$()$	Res(8,9), MGU ε (the empty substitution)																	

Solution 3.

[0.5 points]

(a) Both can be derived. From $kiwi(kai)$ and $\forall x (kiwi(x) \Rightarrow bird(x))$ we derive $bird(kai)$, and with $\forall x (bird(x) \Rightarrow fly(x))$ we derive $fly(kai)$; from $kiwi(kai)$ and the added formula we derive $\neg fly(kai)$. Hence the extended knowledge base is inconsistent, and every formula can be derived from it.	0.25
(b) PL1 is monotonic: for every knowledge base KB and every formula ϕ, the set of formulas derivable from KB is a subset of the formulas derivable from $KB \cup \{\phi\}$. Therefore $fly(kai)$, which is derivable from KB, stays derivable whatever formulas are added.	0.25

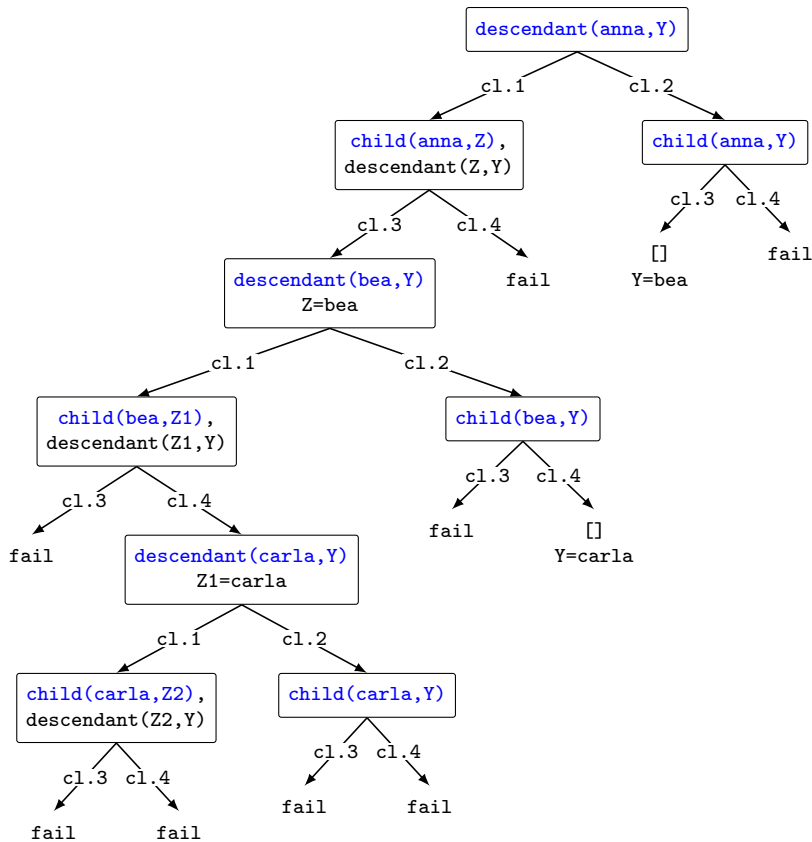
Solution 4.

[2 points]

(a) Clause 2 is the base clause; clause 1 is the recursive clause, because its body calls descendant again. Declarative meaning of clause 1: if X is a child of Z and Z is a descendant of Y, then X is a descendant of Y.	0.5
---	------------

(b) Search tree (the selected goal is blue; Z1, Z2 belong to fresh copies of clause 1; the same tree in another clear notation, also without colour, is accepted):

0.75



PROLOG returns first Y = carla and then Y = bea. PROLOG searches depth-first from left to right and tries the clauses from top to bottom; because the recursive clause 1 comes before the base clause 2, the indirect answer (carla, through bea) is found before the direct one (bea).

0.25

(c) PROLOG returns no answer at all and does not terminate; it never gives the answer false, although Y = carla and Y = bea still follow logically. In practice, the PROLOG system stops with a stack overflow before giving any answer; this answer is also accepted. The first clause tried for descendant(anna, Y) is now clause 1', whose first subgoal descendant(Z, Y) is again solved first with clause 1'; this gives descendant(Z1, Y), then descendant(Z2, Y), and so on. Depth-first search therefore follows an infinitely deep leftmost branch, and the base clause 2 and the child facts are never reached. The declarative meaning is unchanged, but the procedural meaning is not: termination depends on the order of the clauses and of the subgoals.

0.5

Solution 5.**[2 points]**

(a)(i) IDA* starts with $\theta = f(S) = 8$; after an iteration without a goal, the next threshold is the smallest f -value among the nodes cut off in that iteration, so θ takes the values 8, 9, 10 and 12 (no cut-off node has $f = 11$):				1
θ	Expanded nodes, in order	Cut off (f -value)	Next threshold	
8	S	$S-A$ (15), $S-B$ (9)	$\theta = 9$	
9	$S, S-B$	$S-A$ (15), $S-B-C$ (12), $S-B-D$ (10)	$\theta = 10$	
10	$S, S-B, S-B-D$	$S-A$ (15), $S-B-C$ (12), $S-B-D-G$ (13)	$\theta = 12$	
12	$S, S-B, S-B-C$	$S-A$ (15), $S-B-C-D$ (20)	goal node $S-B-C-G$ reached ($f = 12 \leq \theta$)	
A goal node is reached only when its f -value does not exceed θ ; for example, $S-B-D-G$ (13) is cut off at $\theta = 10$. Listing $S-B-C-G$ among the expanded nodes of the last iteration is also accepted.				
IDA* finds the path $S-B-C-G$ with cost $3 + 4 + 5 = 12$.				0.25
(a)(ii) The nodes S and $S-B$ are expanded again in later iterations (S in all four iterations, $S-B$ in the last three); naming the states S and B is also accepted. In exchange, IDA* searches depth-first and stores only the current path and its successors, so it avoids the high memory requirements of A*, which keeps all generated but not yet expanded nodes in <code>NodeList</code> . Naming the other weakness of A*, that <code>NodeList</code> must be kept sorted, is also accepted.				0.25
(b) Minimax values: $W = \max(5, -3) = 5$, $X = \max(3, -2) = 3$, $U = \min(5, 3) = 3$; $Y = \max(7, 13) = 13$, $Z = \max(15, -6) = 15$, $V = \min(13, 15) = 13$; $R = \max(3, 13) = 13$. Max chooses the move to V , the child with the larger minimax value.				0.5

Solution 6.**[2 points]**

(a)(i) By the Bayesian-network factorization, $P(Bur, Al, J) = P(Bur)P(Al Bur)P(J Al)$. The value of this product, $0.2 \cdot 0.8 \cdot 0.9 = 0.144$, is also accepted.		0.25
(a)(ii) Marginalization: $P(Al) = P(Al Bur)P(Bur) + P(Al \neg Bur)P(\neg Bur) = 0.8 \cdot 0.2 + 0.1 \cdot 0.8 = 0.16 + 0.08 = 0.24$ Bayes' theorem: $P(Bur Al) = \frac{P(Al Bur)P(Bur)}{P(Al)} = \frac{0.16}{0.24} = \frac{2}{3} \approx 0.667$.		0.5
(a)(iii) Given its parent Al , the node J is conditionally independent of Bur , which is neither a parent nor a descendant of J . Conditioning on Al therefore gives $P(J Bur) = P(J Al)P(Al Bur) + P(J \neg Al)P(\neg Al Bur) = 0.9 \cdot 0.8 + 0.05 \cdot 0.2 = 0.73$ Marginalization gives $P(J) = 0.9 \cdot 0.24 + 0.05 \cdot 0.76 = 0.216 + 0.038 = 0.254$. Any correct method is accepted, for example $P(J Bur) = \frac{P(J, Bur)}{P(Bur)} = \frac{0.146}{0.2} = 0.73$.		0.25

Since $P(J \mid Bur) = 0.73 \neq 0.254 = P(J)$, J and Bur are not independent: a burglary makes the alarm, and therefore a call from John, much more likely. They are conditionally independent given Al: by the network semantics above, $P(J = j \mid Al = a, Bur = b) = P(J = j \mid Al = a)$, or equivalently $P(J = j, Bur = b \mid Al = a) = P(J = j \mid Al = a)P(Bur = b \mid Al = a)$ for all values $j, a, b \in \{t, f\}$.	0.25
(b)(i) $P(A) = p_1 + p_2 = 0.6$; $P(B \mid A) = \frac{p_1}{p_1 + p_2} = 0.7$, equivalently $0.7p_2 - 0.3p_1 = 0$, so $p_1 = 0.7 \cdot 0.6 = 0.42$; normalization: $p_1 + p_2 + p_3 + p_4 = 1$.	0.25
(b)(ii) The equations give $p_1 = 0.42$, $p_2 = 0.6 - 0.42 = 0.18$ and only $p_3 + p_4 = 0.4$; one equation is missing. MaxEnt chooses, among all distributions satisfying the equations, the one with the largest entropy $H = -\sum_{i=1}^4 p_i \ln p_i$, that is, the one that adds the least information. The variables p_3 and p_4 occur symmetrically in the constraints, so they are indifferent and the maximum lies at $p_3 = p_4 = 0.2$ (maximizing $-p_3 \ln p_3 - p_4 \ln p_4$ subject to $p_3 + p_4 = 0.4$ with a Lagrange multiplier gives the same result). Hence $(p_1, p_2, p_3, p_4) = (0.42, 0.18, 0.2, 0.2)$.	0.25
(b)(iii) $P(B) = p_1 + p_3 = 0.42 + 0.2 = 0.62$ and $P(B \mid \neg A) = \frac{p_3}{p_3 + p_4} = \frac{0.2}{0.4} = 0.5$.	0.25