Last Name: First Name:

Matriculation Number:

Retake Exam
Artificial Intelligence 1

September 23, 2025

Please ignore the QR codes; do not write on them, they are for grading support

	To be used for grading, do not write here													
prob.	1.1	2.1	2.2	2.3	3.1	4.1	5.1	5.2	5.3	6.1	7.1	7.2	Sum	grade
total	10	8	6	7	8	9	8	8	6	8	8	4	90	
reached														

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 1.1 (Prolog in Prolog)
Consider the following Prolog program that represents Prolog in Prolog, i.e. Prolog terms, literals, and clauses are represented as Prolog terms:

```
isTerm(pterm(F,ARGS)) :- string(F), isTermList(ARGS).
isTerm(pvar(X)) :- string(X).

isTermList([]).
isTermList([H|T]) :- isTerm(H), isTermList(T).

isLiteral(plit(P,ARGS)) :- string(P), isTermList(ARGS).

isLiteralList([]).
isLiteralList([H|T]) :- isLiteral(H), isLiteralList(T).

isClause(pclause(H,B)) :- isLiteral(H), isLiteralList(B).
```

Here `string` is a built-in predicate that succeeds if its argument is a string.

1. Explain intuitively what the predicate `isClause(pclause(H,B))` computes? 2 Points

   Solution: It checks whether its argument represents a Prolog clause with head `H` and body `B`.

2. Write the Prolog clause `isNat(succ(N)) :- isNat(N)` as a Prolog term relative to the above program (i.e., such that `isClause` succeeds for it). 3 Points

   Solution:

   ```
   pclause(plit("isNat", [pterm("succ",[pvar("N")])]),
           [plit("isNat", [pvar("N")])]).
   ```

3. Assume that the Prolog term C contains no free variables. How is the result of the query `isClause(C)` affected by exchanging the lines 4 and 5? 2 Points

   Solution: It is not affected.

4. Extend the program above with a unary Prolog predicate `isProgram` that succeeds if its argument is of the form `pprog(P)` where P is a list of clauses. 3 Points

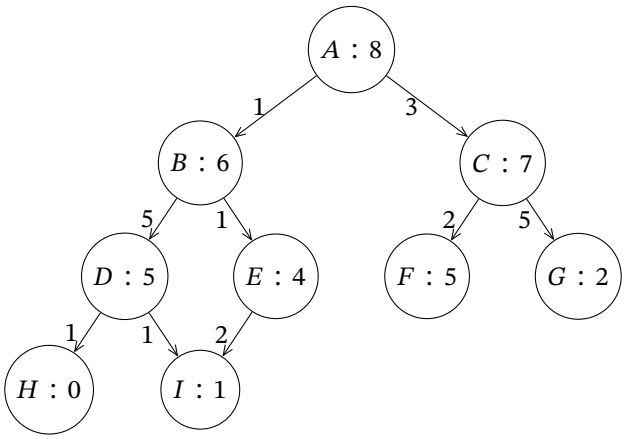
   Solution:

   ```
   isClauseList([]).
   isClauseList([H|T]) :- isClause(H), isClauseList(T).

   isProgram(pprog(C)) :- isClauseList(C).
   ```

2 Search

Problem 2.1 (Search Algorithms)
Consider the graph below. Every edge is labeled with its cost.
Each node is labeled with $n : h(n)$ where n is its name and $h(n)$ its value for heuristic h.



Assume you have already expanded the root node A. In each problem below, list the **next** 4 nodes that will be expanded using the respective search algorithm and (if needed) heuristic h.
If there is a tie, break it using alphabetical order of the nodes.

1. depth-first search — 1 Points

   Solution: B, D, H, I

2. breadth-first search — 1 Points

   Solution: B, C, D, E

3. uniform-cost search — 2 Points

   Solution: B, E, C, I

4. greedy search — 2 Points

   Solution: B, E, I, D

5. A^*-search — 2 Points

   Solution: B, E, I, C

Problem 2.2 (Remembering expanded states)
When searching in a graph, it can happen that the same node can be reached along two different paths. When inserting a node n into the fringe, the search algorithms presented in the course do not check if n has already been expanded before.

1. In about 1 sentence each, explain the advantage and disadvantage of additionally performing such a check. 3 Points

Solution: pro: It allows avoiding redundantly expanding the same node twice, thus potentially saving time or even avoiding a cycle.
con: It requires overhead in space and time to store the set of previously expanded nodes and to check membership in it, which is wasted if no node is found twice (e.g., in a tree).

2. Now assume we have implemented such a check. We consider only *breadth-first* and *uniform-cost* search. 3 Points
Explain in about 2 sentences whether it is correct to simply skip inserting n into the fringe if n has been expanded before.

Solution: BFS: Yes, because the children of n have already been inserted into the fringe during the previous expansion of n.
UCS: No, the second path to n may be cheaper than the path that was found previously. In that case, we have to expand n again and recompute the path costs of its children.

Problem 2.3 (Search Problem)
Consider the deterministic search problem $\langle \mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{I}, \mathcal{G} \rangle$ where

- $\mathcal{S} = \mathbb{Z}$
- $\mathcal{A} = \{-6, -4, 0, 5\}$
- $\mathcal{T}(a, s) = \{a + s\}$
- $\mathcal{I} = \{0\}$
- $\mathcal{G} = \{9\}$

1. Give the state resulting from applying the action sequence $-4, 5, 5$ to the initial state. 1 Points

Solution: 6

2. Give a solution to the problem. 3 Points

Solution: $5, 5, 5, -6$ (any sequence of actions that add up to 9)

3. Explain in about 2 sentences whether DFS a good choice for this problem. 2 Points

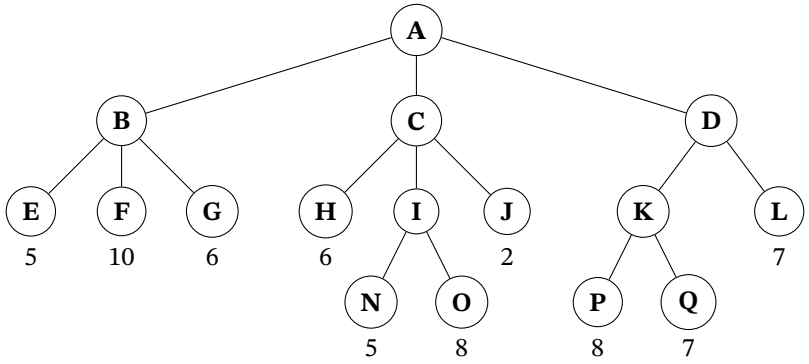
Solution: No. It will not find a solution: whichever action we try first, can be applied infinitely often.

4. If we change the set $\mathcal{A}$ to $\{-6, -4, 0, 2\}$, the problem changes substantially. In what way? 1 Points

Solution: There is no solution.

3 Adversarial Search

Problem 3.1 (Minimax)
Consider the following minimax game tree for the **maximizing** player's turn. The values at the leaves are the static evaluation function values of those states.



1. Which move will the player choose? 2 Points

Solution: *D*

2. What change to the label of only a **single** node can you make such that the player chooses a different move? 2 Points

Solution: The label L of must be lower than 5 (in which case move B is chosen). Making the label equal to 5 causes a draw, which was also accepted as correct.

For the remaining questions: Use $\alpha\beta$-pruning and expand child nodes in alphabetical order.
Ignore any change you may have applied in the previous problem.

3. Which nodes would be pruned? 2 Points

Solution: None

4. What change to the labels of the nodes N and O can you make such that J is pruned but not I? 2 Points

Solution: The labels of N and O must be below 5.

4 Constraint Satisfaction

Problem 4.1 (Assignments, Solutions, and Algorithms)
Consider the CSP given by

- Variables: $\{a, b, c, d\}$
- Domains: $D_a = D_b = D_c = D_d = \{0, 1, 2, 3\}$
- Constraints:
  - ☐ $a + b = 3$
  - ☐ $2 \leq c \cdot d < 4$
  - ☐ $a + c > 4$
  - ☐ $2a + b + c > 7$

1. Mark the constraints that make this CSP non-binary. 1 Points

   Solution: Only the last one (because it is ternary).

2. Give all solutions. 3 Points

   Solution: $(a, b, c, d) \in \{(2, 1, 3, 1), (3, 0, 2, 1), (3, 0, 3, 1)\}$

3. By testing arc-consistency only for the third constraint, which domain values can be eliminated immediately? 2 Points

   Solution: The resulting domains are $D_a = \{2, 3\}$ and $D_c = \{2, 3\}$.

4. After assigning $a = 2$, what is the domain of c after applying forward-checking using the third constraint? 1 Points

   Solution: $D_c = \{2, 3\}$

5. Give an equivalent binary CSP. 2 Points

   Solution: Remove the ternary constraint. This works because that constraint happens to be redundant: it is implied by the others.

5 Logic

Problem 5.1 (Propositional Logic)
Consider the formula A of propositional logic given by

$$(p \wedge q) \Rightarrow \big((p \Rightarrow \neg q) \Rightarrow q\big)$$

1. Give the propositional variables that occur in A. 2 Points

Solution: p and q

2. Give all assignments that satisfy A. 2 Points

Solution: All assignments, i.e., $\varphi(p), \varphi(q) \in \{\top, \bot\}$.

3. Give a DNF for A. 2 Points

Solution: Systematic transformation yields $\neg p \vee \neg q \vee (p \wedge q) \vee q$. But any tautology works, e.g., $q \vee \neg q$ or T.

4. What is the **smallest** number n for which we can give n formulas such that every other formula in p and q is equivalent to one of them? 2 Points

Solution: 16.
There are 4 assignments each returning one of 2 values. So there are at most 2^4 pairwise non-equivalent formulas. And there are indeed 16: a possible set contains T, p, q, $p \Leftrightarrow q$, $p \wedge q$, $p \vee q$, $p \Rightarrow q$, $q \Rightarrow p$, and their negations.

Problem 5.2 (First-Order Logic)
Consider the formula A of first-order logic with equality given by

$$\big(\forall x.\exists y.p(x,y)\big) \Rightarrow \forall x.p(x,f(x))$$

over the smallest signature that makes it well-formed.

1. Give that signature. 2 Points

Solution: The signature with a unary function symbol f and a binary predicate symbol p.

2. Give a model that satisfies A. 2 Points

Solution: Lots of options. The easiest way is to make p false everywhere or true everywhere, e.g., (U, I) with universe $U = \{0\}$, $I(f)(u) = 0$ and $I(p) = \emptyset$.

3. Now assume $\forall x.\exists y.p(x,y)$ is the only axiom. Give a second axiom such that the models (U, I) that satisfy both axioms are exactly the ones where $I(p)$ (which is a binary relation on U) also represents a unary function from U to U. 2 Points

Solution: The first axiom already ensures that $I(p)$ is a total relation. So we need to additionally make it functional by requiring, e.g., $\forall x.\forall y.\forall y'.p(x,y) \wedge p(x,y') \Rightarrow y = y'$.

4. In about 1 sentence, explain why the dual of A (i.e., with $\Leftarrow$ instead of $\Rightarrow$) is a theorem. 2 Points

Solution: For any x, we can use $f(x)$ as the witness for y that allows proving $\exists y.p(x,y)$.

Problem 5.3 (Proving in Natural Deduction)

1. Prove the formula 6 Points

$$\big(\forall x.p(x,f(x))\big) \Rightarrow \forall y.\exists u.p(y,u)$$

using a natural deduction–style proof.

Solution: Starting goal is the formula.

Apply implication-introduction. Let h be the resulting hypothesis $\forall x.p(x,f(x))$. New goal is $\forall y.\exists u.p(y,u)$.
Apply forall-introduction for arbitrary y. New goal is $\exists u.p(y,u)$.
Apply exists-introduction using the term $f(y)$ for u. New goal is $p(y,f(y))$.
Apply forall-elimination to h using the term $f(y)$ for x. That yields $p(y,f(y))$.
That closes the proof.

6 Knowledge Representation

Problem 6.1 (ALC Description Logic)
Consider the following ALC setting:

- concepts: `food`, `drink`, `person`
- relations: `likes`, `goesWith`

We abbreviate every concept/relation by its first letter.

1. Give an ALC ABox with 1 piece of food and 2 persons that like it. 2 Points

Solution: $x : p, y : p, z : f, x\,l\,z, y\,l\,z$

2. Give an ALC TBox that formalizes the property that any person who likes some food also likes some drink. 1 Points

Solution: $(p \sqcap \exists l.f) \sqsubseteq (\exists l.d)$

3. Give an ALC TBox that formalizes the property that persons cannot be liked. 1 Points

Solution: $\exists l.p \sqsubseteq \bot$

4. Give an ALC formalization for the concept of persons who only like food that a drink goes with. 2 Points

Solution: $p \sqcap \forall l.(f \sqcap \exists g.d)$

5. Give the translation to first-order logic of the ALC statement $(\forall l.f) \sqsubseteq (\exists l.d)$. 2 Points

Solution: $\forall x.\big(\forall y.l(x,y) \Rightarrow f(y)\big) \Rightarrow \big(\exists y.l(x,y) \wedge d(y)\big)$

7 Planning

Problem 7.1 (STRIPS Planning)
Consider a map of Australia. Starting from Sydney (S), we want to visit Darwin (D), Brisbane (B), and Perth (P), passing Adelaide (A).
Using $C = \{A, B, D, P, S\}$, we use the following STRIPS task:

- The facts are $\texttt{at}(x)$, $\texttt{visited}(x)$, and $\texttt{road}(x, y)$ for $x, y \in C$.
- The actions are $\texttt{drive}(x, y)$ for $x, y \in C$ with precondition/add-list/delete-list given by, respectively, $\{\texttt{at}(x), \texttt{road}(x, y)\}$, $\{\texttt{at}(y), \texttt{visited}(y)\}$, and $\{\texttt{at}(x)\}$.
- The initial state is $I = \{\texttt{at}(S), \texttt{visited}(S)\} \cup \mathit{Roads}$ where $\mathit{Roads} = \{\texttt{road}(A, P), \texttt{road}(A, D), \texttt{road}(D, A), \texttt{road}(A, S), \texttt{road}(S, A), \texttt{road}(S, B), \texttt{road}(B, S)\}$.
- The goal is $G = \{\texttt{visited}(B), \texttt{visited}(D), \texttt{visited}(P)\}$.

You may abbreviate all fact/action names by their first letter.

1. Give an optimal plan for this task, or argue why no plan exists. 2 Points

Solution: $\texttt{drive}(S, B), \texttt{drive}(B, S), \texttt{drive}(S, A), \texttt{drive}(A, D), \texttt{drive}(D, A), \texttt{drive}(A, P)$

2. Give an optimal plan for the delete-relaxed task, or argue why no plan exists. 2 Points

Solution: $\texttt{drive}(S, B), \texttt{drive}(S, A), \texttt{drive}(A, D), \texttt{drive}(A, P)$. (Some reordering is possible.)

3. Let $I' = I \cup \{\texttt{visited}(D)\}$. Give $h^*(I')$. 2 Points

Solution: 4

4. Let $h(s) = |G \setminus s|$. Explain whether h is an admissible heuristic. 2 Points

Solution: It is admissible. $h(s)$ is the number of cities that still have to be visited. Because each city takes at least one action to visit, we have $h(s) \leq h^*(s)$.

Problem 7.2 (Partial Order Planning)

1. In about 2 sentences, explain the key properties and significance of the Sussmann anomaly. 2 Points

   Solution: It is a planning problem in which the goal consists of 2 facts. If a planner first focuses on either one of them as a subgoal, it will reach a state in which the next action must undo the first subgoal to achieve the second subgoal. It shows that planning problems require more finesse than simply splitting the goal into subgoals and achieving them in order.

2. In 1-2 sentences, explain the general idea of a causal link. 2 Points

   Solution: It relates two actions that could potentially be applied in order because the first one's effect helps establish the second one's preconditions.