Last Name: First Name:

Matriculation Number:

Exam
Symbolic Methods for AI

September 29. 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	1.2	1.3	2.1	2.2	2.3	3.1	4.1	5.1	5.2	6.1	6.2	6.3	Sum	grade
total	6.5	9	2	4	4	3	8	8	4	3	2	2	5	60.5	
reached															

In the course Symbolic Methods for AI 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 Talking about Mathematics

Problem 1.1 (Talking about Sets)
Given the following sets

1. $A = \{a, b, c, d, e\}$
2. $B = \{d, f, h\}$
3. $C = \{d, f, g, i\}$

where the $a, \ldots, h, i$ are pairwise distinct.
Define each of the following operations on sets **in MathTalk** and apply it to the given sets:

1. Intersection (general formula in MathTalk) 1 Points

   Solution: intersection: $P \cap Q := \{x \mid x \in P \wedge x \in Q\}$

2. $A \cap B =$ 0.5 Points

   Solution: $A \cap B = \{d\}$

3. Union (general formula in MathTalk) 1 Points

   Solution: union: $P \cup Q := \{x \mid x \in P \vee x \in Q\}$

4. $B \cup C =$ 0.5 Points

   Solution: $B \cup C = \{d, f, h, g, i\}$

5. Set difference (general formula in MathTalk) 1 Points

   Solution: set difference: $P \setminus Q := \{x \mid x \in P \wedge x \notin Q\}$

6. $A \setminus B =$ 0.5 Points

   Solution: $A \setminus B = \{a, b, c, e\}$

7. Cartesian product (general formula in MathTalk) 1 Points

   Solution: Cartesian product: $P \times Q := \{(p,q) \mid p \in P \wedge q \in Q\}$

8. $B \times C =$ 1 Points

   Solution: $B \times C = \{(d,d), (d,f), (d,g), (d,i), (f,d), (f,f), (f,g), (f,i), (h,d), (h,f), (h,g), (h,i)\}$

Problem 1.2 (Properties of Relations)
Let R and S be relations on some given set A.

1. Prove or refute that 3 Points

   If R and S are symmetric then $R \cap S$ is symmetric.

   Solution: Let (x,y) be in $R \cap S$. From the definition of intersection (x,y) is in R and S. Since R and S are symmetric we get $(y,x) \in R$ and $(y,x) \in S$ by the definition of symmetry. This implies that (y,x) is in $R \cap S$ by the definition of intersection. Hence, $R \cap S$ is symmetric.

2. Prove or refute that 3 Points

   If R is reflexive then all subsets of R are reflexive.

   Solution: Let $R := \{(a,a),(a,b),(b,b)\}$. R is a reflexive relation on $\{a,b\}$. Howevper the subset $\{(a,a),(a,b)\}$ of R is not reflexive.

3. Prove or refute that 3 Points

   If R is transitive then the converse relation R^{-1} is transitive.

   Solution: Assume we have $(a,b),(b,c) \in R^{-1}$. Then, by definition of the converse relation we have $(b,a),(c,b) \in R$. Since R is transitive, we also know that $(c,a) \in R$. This implies again by definition of the converse relation that $(a,c) \in R^{-1}$. Hence R^{-1} is transitive.

Problem 1.3 (Truths about Proofs) 2 Points
Which of the following statements about the truth of mathematical statements are correct?

- ☐ If I have a counterexample for A, then A is false.
- ☐ If I have an example for A, then A is true.
- ☐ If I do not have a proof for $\neg A$, then A is true.
- ☐ If I do not have a proof for A, then A is false.
- ☑ If I have a proof for A, then A is true.
- ☑ If I have a proof for $\neg A$, then A is false.

2 Standard ML

Problem 2.1 (SML Predicate for Reflexive Relations) 4 Points
Write an SML predicate (i.e. a function with result type `bool`) that given a list S and a list R of pairs whose components are in S, returns `true`, iff the relation R is reflexive on the set represented by the list S.

Hint: You can use a predicate `member: 'a -> 'a list -> bool` that checks whether the first ar-

gument is element of the second argument (a list) without defining it yourself.

Solution: One possible implementation is

```
fun refl [] R = true
  | refl (x::xs) R = member (x,x) R andalso refl xs R
```

Problem 2.2 (Mapping and Appending)
`mapcan` is a higher-order function that maps a list-valued function over a list and concatenates all the result lists into a single list.

1. What is the SML type of the function `mapcan` (explain in terms of the types of its arguments and results). 2 Points

   Solution: The SML type of `mapcan` is `('a -> 'b list) * 'a list -> 'b list`. The function `mapcan` takes two arguments: a list-valued function, which has the type `('a -> 'b list)` and a list of some type `'a list`. The output of the function is the concatenation of the resulting lists so it is again a list `'b list`.

2. Implement `mapcan` in SML recursively. 2 Points

   Solution:

   ```
   fun mapcan(f,nil) = nil
     | mapcan(f,h::t) = (f h)@(mapcan(f,t))
   ```

Problem 2.3 (Mapping and Appending) 3 Points
Can the function `mapcan` from above be written using `foldl`/`foldr`? If so give an implementation, if not, argue why not.

Solution:

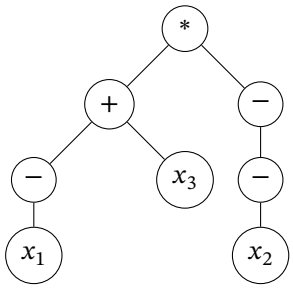
```
fun mapcan_with(f,l) = foldl(fn (v,s) => s@f(v)) nil l;
```

3 Graphs and Trees

Problem 3.1 (Graph and Trees)

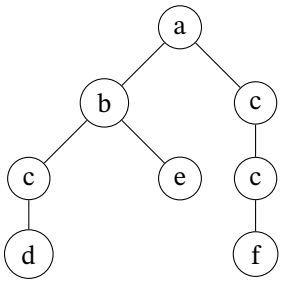
1. Draw the parse tree of the expression string `(-X1+X3)*(-(-X2))` with respect to the grammar for arithmetic expressions presented in the SMAI course. 3 Points

   Solution:



2. Give an example of a different node labeled graph that is isomorphic to the parse tree above and write its mathematical representation in terms of concrete sets and functions. 3 Points

Solution: One such graph is



A parse tree is a node labeled graph, i.e. a graph

$$\langle\{1,2,3,4,5,6,7,8\},\{(1,2),(2,3),(3,4),(2,5),(1,6),(6,7),(7,8)\}\rangle$$

and a node labeling function:

$$1\mapsto a, 2\mapsto b, 3\mapsto c, 4\mapsto d, 5\mapsto e, 6\mapsto c, 7\mapsto c, 8\mapsto f$$

3. Can the graph underlying the parse tree be isomorphic to a cyclic graph? Justify why, or why not? 2 Points

Solution: No, trees are acyclic and cycles are preserved under graph isomorphisms.

4 Mathematical Structures

Problem 4.1 (Foobar Structures)
A structure $\langle S, v\rangle$, where S is a set, and $v: S\times S\to\mathbb{N}$ is a total function is called a **foobar**, iff

- $v(a,a)=1$ for all $a\in S$ and
- for all $a,b\in S$ we have $a=b$, if $v(a,b)=v(b,a)$.

1. Provide a function v such that $\langle\{1\},v\rangle$ is a foobar 2 Points

Solution: $v(1,1)=1$

2. Provide a function v such that $\langle\{1,2,3\}, v\rangle$ is a foobar. 2 Points

Solution: E.g. $v(x,y) = \begin{cases} 2 & \text{if } x < y \\ 1 & \text{if } x = y \\ 3 & \text{if } x > y \end{cases}$

3. Prove or refute: 4 Points

If $\langle S, v\rangle$ and $\langle S', v'\rangle$ are foobars, then $\langle S \times S', v''\rangle$ is a foobar where $v''((a,b),(c,d)) = v(a,c) + v'(b,d)$.

Solution: This is false; here is a counterexample:
Let $S = S' = \{1\}$ and $v(x,y) = v'(x,y) = 1$. Then $v''((1,1),(1,1)) = 2$, violating the first condition of a foobar.

5 Complexity Analysis

Problem 5.1 (Time complexity of an SML function)
Consider the the following SML function:

```
1 fun twist(nil) = nil
2   | twist([hd1]) = [hd1]
3   | twist(hd1::(hd2::tl)) = twist(hd1::tl)@twist(hd2::tl)
```

1. How many times is line 3 executed to compute `twist [5,6,2,9]`? 2 Points

Solution: To compute a list of 4 elements, we execute line 3 eight times: For each recursive calls (and there are 3 for a four-element list) the number of executions doubles (two recursive calls).

2. What is the time complexity of `twist` ($\Theta(f)$) in terms of the length of the input list. Justify your answer! 2 Points

Solution: The recursive case of the function above has two recursive calls to argument lists that are one element shorter, so the evaluations double with every increment in input length. So, time complexity is $\Theta(2^n)$.

Problem 5.2 (Landau sets) 3 Points
Order the Landau sets below by specifying which ones are proper subsets and which ones are equal (e.g.: $\mathcal{O}(a) \subset \mathcal{O}(b) \subset \mathcal{O}(c) \equiv \mathcal{O}(d) \subset \mathcal{O}(e) \ldots$)

$$\mathcal{O}(n^2);\ \mathcal{O}(n!);\ \mathcal{O}(|\sin n|);\ \mathcal{O}(n^n);\ \mathcal{O}(1);\ \mathcal{O}(2^n);\ \mathcal{O}(2n^2 + 2^{72})$$

Solution: $\mathcal{O}(|\sin n|) \equiv \mathcal{O}(1) \subset \mathcal{O}(2n^2 + 2^{72}) \equiv \mathcal{O}(n^2) \subset \mathcal{O}(2^n) \subset \mathcal{O}(n!) \subset \mathcal{O}(n^n)$

6 Formal Languages and Grammars

Problem 6.1 (Grammars) 2 Points

Consider the following phrase structure grammar in BNF form:

$$\begin{array}{rcl} F & ::= & N(Ts) \mid F \wedge F \mid \forall N.F \\ T & ::= & N \mid N(Ts) \\ Ts & ::= & T \mid T, Ts \\ N & ::= & f \mid g \mid h \mid p \mid q \mid x \mid y \mid z \end{array}$$

Which of the following strings are derivable from F:

- ☑ $p(g(x, g(y)))$
- ☐ $f(p(x) \wedge q(y))$[1]
- ☑ $p(f(x, y, z)) \wedge \forall x.p(y)$
- ☐ $\forall x, y.p(x, y)$[2]

Solution:

Problem 6.2 (Grammars) 2 Points

Which of the following are true statements:

- ☑ Every regular language has a context-free grammar.
- ☑ If every production rule of a grammar has a nonterminal symbol in its body, the language is empty.
- ☐ If the heads of all production rules in a grammar have exactly one symbol, the language is finite.[3]
- ☐ If two grammars produce the same language, they have the same number of nonterminal symbols.[4]

Solution:

Problem 6.3 (Abstract and Concrete Grammars) 5 Points

Let L be a term language. Describe – in your own words the difference between an abstract grammar and a corresponding concrete grammar for L.

Hint: Think about brackets and grammar types/classes and the complexity difference between the two kinds.

Solution: An abstract grammar is a tree grammar – a grammar, where the target data structure is trees not strings – for the expression trees in L. A concrete grammar is a string grammar for a surface

[1] *$\wedge$ occurs inside argument of f*

[2] *multiple names after $\forall$*

[3] *context-free grammars can produce infinite languages*

[4] *a language can have many different grammars*

language of L' of L -that spells out details like bracketing and such. In particular, the parse trees of L'-strings are just the expressions in L. Generally abstract grammars are simpler than concrete grammars, which have to account for syntactic details like particular bracketing.
