Name: _____

Logic and Mechanized Reasoning

First Midterm Exam (Practice) 2026

Write your answers in the space provided, using the back of the page if necessary. You may use additional scratch paper. Justify your answers, and provide clear, readable explanations.

Problem	**Points**	**Score**
1	60	
2	30	
3	60	
4	20	
5	30	
6	30	
7	34	
8	36	
Total	**300**	

Good luck!

Problem 1. (60 points) *Propositions*: Answer each question with true or false.

		$\top$	$\bot$
(a)	Every propositional formula with exactly three satisfying assignments is falsifiable.	$\boxed{\top}$	$\square$
(b)	For all propositional formulas A, B, C it holds that $$\{A, \neg A\} \models (A \to (\neg B \leftrightarrow C) \wedge (\neg C \vee B)) \to \neg A$$	$\boxed{\top}$	$\square$
(c)	Let I be an inductively defined set and let f be defined by structural recursion on I. Then f is a total function.	$\boxed{\top}$	$\square$
(d)	If A and B are any formulas in negation normal form, the formula $(A \vee \neg B) \wedge \neg A$ is also in negation normal form.	$\square$	$\boxed{\bot}$
(e)	For some formulas in negation normal form, there exists a logically equivalent formula in negation normal form that is exponentially smaller.	$\boxed{\top}$	$\square$
(f)	If p is the only variable occurring in a propositional formula, then the formula is in negation normal form.	$\square$	$\boxed{\bot}$
(g)	For an element a of type α it holds that $[a] :: [] = [[a]]$.	$\boxed{\top}$	$\square$
(h)	The expression $[[1, 2, 3], 4, 5]$ has type $List\ (List\ \mathbb{N})$.	$\square$	$\boxed{\bot}$
(i)	A recursive definition of F with a recursive case $F_{N+2} = F_{N+1} - F_N$ can have only one base case.	$\square$	$\boxed{\bot}$
(j)	Structural recursion formulates a definition by recursion on the structure of an inductively defined type.	$\boxed{\top}$	$\square$

Problem 2. (30 points) Prove the following:

$$1 + 2 + 4 + \ldots + 2^n = 2^{n+1} - 1.$$

Solution

Use induction on n. When $n = 0$, we have $1 = 1$. Assuming the claim holds for n, we have

$$\begin{aligned} 1 + 2 + 4 + \ldots + 2^n + 2^{n+1} &= (2^{n+1} - 1) + 2^{n+1} \\ &= 2 \cdot 2^{n+1} - 1 \\ &= 2^{n+2} - 1. \end{aligned}$$

Problem 3. The following Lean code defines a type of labeled binary trees and two functions. Here `allLe t n` returns `true` if every node of tree `t` is less than or equal to `n` and `false` otherwise, and `allGe t n` returns `true` if `n` is greater than or equal to every node of tree `t` and `false` otherwise.

```
inductive LBinTree (α : Type)
| empty : LBinTree α
| node  : α → LBinTree α → LBinTree α → LBinTree α

open LBinTree

def allLe : LBinTree Nat → Nat → Bool
| empty, _      => true
| node m s t, n => m ≤ n && allLe s n && allLe t n

def allGe : LBinTree Nat → Nat → Bool
| empty, _      => true
| node m s t, n => n ≤ m && allGe s n && allGe t n
```

Part a) (30 points) Define a function `isOrdered : LBinTree Nat → Bool` that determines whether the inorder traversal yields a list of natural numbers that is nondecreasing (each element is less than or equal to the one that follows it).

Solution

```
def ordered : LBinTree Nat → Bool
| empty      => true
| node n s t => ordered s && ordered t && allLe s n && allGe t n
```

See also `practice1_solutions.lean`.

Part b) (30 points) Define a function `insert : Nat → LBinTree Nat → LBinTree Nat` such that if `n` is a natural number and `t : LBinTree Nat` is already in order, then `insert n t` inserts `n` in `t` in the right place. You can assume that when this function is called, `isOrdered t` is `true`.

Solution

```
def insert (n : Nat) : LBinTree Nat → LBinTree Nat
| empty => node n empty empty
| node m s t => if n ≤ m then node m (insert n s) t else node m s
   (insert n t)
```

See also `practice1_solutions.lean`.

Problem 4. (20 points) Prove the following statement (directly, from the semantic definitions) or find a counterexample: For any set of propositional formulas Γ and for any propositional formulas A and B, if $\Gamma \models A$ and $\Gamma \models B$ then $\Gamma \models A \wedge B$.

Solution

Suppose $\Gamma \models A$ and $\Gamma \models B$. Let τ be any truth assignment such that for every formula C in Γ, $[\![C]\!]_\tau = \top$. Since $\Gamma \models A$, we have $[\![A]\!]_\tau = \top$, and since $\Gamma \models B$, we have $[\![B]\!]_\tau = \top$. By the definition of truth for propositional formulas, we have $[\![A \wedge B]\!]_\tau = \top$. So $\Gamma \models A \wedge B$.

Problem 5. Professor Lean has a dress code problem. He is aware that wearing neither a tie nor a shirt is impolite. It is clear that one should not wear a tie without a shirt. Personally, he considers wearing a tie and a shirt as overkill. How can we solve this problem?

Part a) (10 pt) Express the dress code problem as a propositional formula D using two propositional variables: s for wearing a shirt and t for wearing a tie.

Solution

$$\neg(\neg t \wedge \neg s) \wedge \neg(t \wedge \neg s) \wedge \neg(t \wedge s)$$

or

$$(t \vee s) \wedge (t \to s) \wedge \neg(t \wedge s).$$

(Other solutions are possible.)

Part b) (10 pt) Convert D into an equivalent formula in conjunctive normal form.

Solution

$$(t \vee s) \wedge (\neg t \vee s) \wedge (\neg t \vee \neg s)$$

Part c) (10 pt) Is the formula D satisfiable? Either answer *unsatisfiable* or provide a satisfying assignment.

Solution

Define τ by $\tau(s) = \top$ and $\tau(t) = \bot$. Then D is satisfied by τ.

Problem 6. (30 points) Put the following sentence in disjunctive normal form:

$$(\neg(p \wedge q) \rightarrow r) \wedge (r \rightarrow q).$$

Solution

The expression is equivalent to

$$((p \wedge q) \vee r) \wedge (\neg r \vee q)$$

which is equivalent to

$$(p \wedge q \wedge \neg r) \vee (r \wedge \neg r) \vee (p \wedge q \wedge q) \vee (r \wedge q)$$

and hence

$$(p \wedge q \wedge \neg r) \vee (p \wedge q) \vee (r \wedge q).$$

This last line simplifies to

$$(p \wedge q) \vee (r \wedge q).$$

Using a truth table gives

$$(p \wedge q \wedge r) \vee (p \wedge q \wedge \neg r) \vee (\neg p \wedge q \wedge r),$$

which also simplifies to the same thing.

Problem 7.

Recall the Sudoku problem discussed in class. A n-Sudoku is an $n^2 \times n^2$ grid with the numbers 1 to n^2 occurring uniquely in every row and column, and also uniquely in $n \times n$ subgrids. Solving a Sudoku can be done by encoding it as a satisfiability problem. In our encoding, variable $p_{i,j,k}$ with $i, j, k \in \{1, 2, 3, 4\}$ is true if and only if that the square on row i and column j has number k. The top-left square is in row 1 and column 1. For this question, consider the 2-Sudoku below.

2			
		3	
			1
	1		

Part a) (20 points) The encoding includes the following clauses. Apply unit propagation to these clauses, repeating until the unit propagation rule can no longer be applied.

Make a list of the clauses that unit propagation uses to extend the assignment, listing them in the order that they become unit under the extended assignments. Rearrange the literals in each clause so that the unit literal comes first.

- $p_{1,1,1} \vee p_{1,1,2} \vee p_{1,1,3} \vee p_{1,1,4}$
- $p_{1,1,3} \vee p_{1,2,3} \vee p_{2,1,3} \vee p_{2,2,3}$
- $\neg p_{1,1,1} \vee \neg p_{1,1,2}$
- $\neg p_{1,1,2} \vee \neg p_{1,1,3}$
- $\neg p_{1,1,2} \vee \neg p_{1,1,4}$
- $\neg p_{2,1,3} \vee \neg p_{2,3,3}$
- $\neg p_{2,2,3} \vee \neg p_{2,3,3}$
- $p_{1,1,2}$
- $p_{2,3,3}$

Solution

1. $p_{1,1,2}$
2. $p_{2,3,3}$
3. $\neg p_{1,1,1} \vee \neg p_{1,1,2}$
4. $\neg p_{1,1,3} \vee \neg p_{1,1,2}$
5. $\neg p_{1,1,4} \vee \neg p_{1,1,2}$
6. $\neg p_{2,1,3} \vee \neg p_{2,3,3}$
7. $\neg p_{2,2,3} \vee \neg p_{2,3,3}$
8. $p_{1,2,3} \vee p_{1,1,3} \vee p_{2,1,3} \vee p_{2,2,3}$

Note that $p_{1,1,1} \vee p_{1,1,2} \vee p_{1,1,3} \vee p_{1,1,4}$ does not become unit, but is satisfied on a single literal.

Part b) (14 points) Notice that for the shown 2-Sudoku, applying unit propagation on the entire encoding yields a solution to the puzzle. In this case, the puzzle has only one solution.

Suppose we start with the encoding of an n-Sudoku and that unit propagation yields a solution. Is this enough to guarantee that the puzzle has only one solution? Explain your answer.

Solution

Yes, unit propagation preserves the set of satisfying assignments. If unit propagation assigns all variables, then there is only one solution.

Problem 8. Let Γ be a CNF formula, which you can think of as a set of clauses, and let τ be a partial assignment to the variables of Γ.

Remember that we use $[\![\Gamma]\!]_\tau$ to denote the result of deleting all the clauses that contain a literal that τ makes true, and removing all literals that τ sets to false from the remaining clauses.

Recall also that τ is an *autarky* for Γ if the following holds: for every clause C in Γ, if τ touches C (i.e. assigns to one of the literals in C), then τ satisfies C.

Part a) (18 points) Prove that if τ is an autarky for Γ, then $[\![\Gamma]\!]_\tau \subseteq \Gamma$.

Solution

Suppose C is in $[\![\Gamma]\!]_\tau$. Then C is the result of deleting literals from some clause C' in Γ that is not satisfied by τ. Since τ is an autarky, we have that τ doesn't touch C'. So $C = C'$.

Part b) (18 points) Prove that if τ is an autarky for Γ, then $[\![\Gamma]\!]_\tau$ is equisatisfiable with Γ.

Solution

By part a), if Γ is satisfiable, then so is $[\![\Gamma]\!]_\tau$. In the other direction, suppose σ satisfies $[\![\Gamma]\!]_\tau$. Let ρ be σ together with τ. Then, in Γ, all the clauses that are satisfied by τ are satisfied by ρ. All the clauses that are not satisfied by τ are in $[\![\Gamma]\!]_\tau$, and so they are satisfied by σ, and hence by ρ.