Name: ____________________________

Logic and Mechanized Reasoning

Second Midterm Exam

November 2, 2021

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	10	
2	12	
3	12	
4	10	
5	10	
6	12	
7	12	
Total	**78**	

Good luck!

Problem 1. (10 points) Use the Tseitin transformation (rather than distributing) to find a CNF formula that is equisatisfiable with this one:

$$p \wedge (q \vee (r \wedge \neg s) \vee \neg(q \to t)).$$

(You don't have to worry about making it the smallest such formula.)

Problem 2.

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 on row 1 and column 1. For this question, consider the 2-Sudoku below.

2			
		3	
			1
	1		

Part a) (8 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.

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

Part b) (4 points) Notice that for the shown 2-Sudoku, applying unit propagation on the entire encoding results in 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 unit propagation results in a solution. Is this enough to guarantee that the puzzle has a only one solution? Explain your answer.

Problem 3. 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) (6 points) Prove that if τ is an autarky for Γ, then $[\![\Gamma]\!]_\tau \subseteq \Gamma$.

Part b) (6 points) Prove that if τ is an autarky for Γ, then $[\![\Gamma]\!]_\tau$ is equisatisfiable with Γ.

Problem 4.

Consider the CNF formula $\Gamma = (p \vee q) \wedge (\neg p \vee r) \wedge (\neg q \vee \neg r)$

Part a) (6 points) Compute all possible non-tautological resolvents of Γ until fixpoint. In other words, keep resolving clauses until no new resolvents are found.

Part b) (4 points) Explain how the procedure above can be used to determine whether or not a CNF formula is satisfiable.

Problem 5.

Remember that the DPLL search tries to find a satisfying assignment for a set of clauses Γ by doing a backtracking search on partial assignments τ. At a node τ in the search, DPLL tries to find a satisfying assignment to $[\![\Gamma]\!]_\tau$.

Part a) (3 points) What does it mean to say that a literal ℓ is a *pure literal* in $[\![\Gamma]\!]_\tau$?

Part b) (7 points) Suppose ℓ is pure in $[\![\Gamma]\!]_\tau$ and let τ' be the assignment $\tau[\ell \mapsto \top]$. Show that $[\![\Gamma]\!]_{\tau'}$ is satisfiable if and only if $[\![\Gamma]\!]_\tau$ is satisfiable.

Problem 6. Remember that in Lean we define the type `Clause` to be *List Lit*.

Part a) (6 points) Define a function `findComplement? : Clause → Clause → Option Lit` that finds a literal ℓ such that ℓ occurs in the first clause and $\neg\ell$ occurs in the second. If there isn't one, the function should return `none`. You can assume that you have a function `List.contains : α → List α → Bool` that determines whether an element is in a list.

Part b) (6 points) Define a function `resolve : Clause → Clause → Option Clause` that applies the resolution rule to two clauses, assuming there is a complementary pair, and returns `none` otherwise. You can assume that you have a function `List.erase : α → List α → List α` that deletes an element from a list, if it is present.

Problem 7. Given the declaration `variable (p q r : Prop)`, as best you can, try to write Lean proofs of the two theorems shown. Also tell us what the goal looks like after each line in your proof, i.e. the hypotheses and conclusion. Don't worry too much about the syntax or writing the goal exactly like Lean does; we are more interested in seeing that you know what steps are allowed.

Part a) (6 points)

```
example : p ∧ q → q ∨ r := by
```

Part b) (6 points)

```
example : (p → q) → (q → r) → (p → r) := by
```