Name: ____________________________

Logic and Mechanized Reasoning

Second Practice Exam

Spring 2024

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	13	
2	20	
3	20	
4	30	
5	26	
6	20	
7	21	
Total	**150**	

Good luck!

Problem 1.

Remember that if Γ is a set of clauses, $[\![\Gamma]\!]_\tau$ is the set of clauses obtained by deleting any clause that has a literal that is assigned $\top$ and removing from each clause any literal that is assigned $\bot$.

Remember also 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 $[\![\Gamma]\!]_\tau$?

Solution

It means the negation of ℓ does not occur in $[\![\Gamma]\!]_\tau$.

Part b) (10 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.

Solution

Since $\neg\ell$ does not occur in $[\![\Gamma]\!]_\tau$, $[\![\Gamma]\!]_{\tau'}$ is the result of deleting all clauses of $[\![\Gamma]\!]_\tau$ that contain ℓ. Since $[\![\Gamma]\!]_{\tau'}$ is a subset of $[\![\Gamma]\!]_\tau$, if $[\![\Gamma]\!]_\tau$ is satisfiable, then $[\![\Gamma]\!]_{\tau'}$ is satisfiable.

Conversely, suppose σ satisfies $[\![\Gamma]\!]_{\tau'}$, then $\sigma[\ell \mapsto \top]$ satisfies $[\![\Gamma]\!]_\tau$.

Problem 2. Remember that in Lean we define the type `Clause` to be *List Lit*.

Part a) (10 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.

Solution

```
def findComplement? : Clause → Clause → Option Lit
| [], C2         => none
| (l :: C1), C2 => if C2.contains l.negate then some l else
  findComplement? C1 C2
```

Part b) (10 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.

Solution

```
def resolve (C1 C2 : Clause) : Option Clause :=
match findComplement? C1 C2 with
  | some l => some $ (C1.erase l).union (C2.erase l.negate)
  | none   => none
```

Problem 3. (20 points)

Remember that a resolution proof of a clause D from hypotheses in Γ is a sequence of clauses $C_0, C_1, \ldots, C_n$ containing D such that for each $i \leq n$, either C_i is in Γ or C_i is obtained from previous clauses C_j and C_k.

Let τ be any truth assignment and let τ' be $\tau[P \to \top]$. Suppose there is a resolution proof of a clause D from hypotheses in $[\![\Gamma]\!]_{\tau'}$. Show that there is a resolution proof of either D or $D \vee \neg P$ from hypotheses in $[\![\Gamma]\!]_{\tau}$.

Solution

Suppose $C_0, C_1, \ldots, C_n$ is a resolution proof of D from hypotheses in $[\![\Gamma]\!]_{\tau'}$. Show by strong induction on i that there is a resolution proof of either C_i or $C_i \vee \neg P$ from hypotheses in $[\![\Gamma]\!]_{\tau}$.

Suppose the claim is true for all $j < i$. If C_i is an element of $[\![\Gamma]\!]_{\tau'}$, then C_i is of the form $[\![E]\!]_{\tau'}$ for some E. Then either E contains $\neg P$, in which case $[\![E]\!]_{\tau}$ is $[\![E]\!]_{\tau'} \vee \neg P$, or it doesn't, in which case $[\![E]\!]_{\tau}$ is $[\![E]\!]_{\tau'} \vee \neg P$.

Otherwise, C_i follows by the resolution rule applied to C_j and C_k for some j and k less than i. By the induction hypothesis, there is a resolution proof of C_j or $C_j \vee \neg P$ from hypotheses in $[\![\Gamma]\!]_{\tau}$ and a resolution proof of C_k or $C_k \vee \neg P$ from hypotheses in $[\![\Gamma]\!]_{\tau}$. Applying the resolution rule to these gives a proof of either C_i or $C_i \vee \neg P$.

Problem 4. Given the declaration `variable (P Q R : Prop)`, as best you can, try to write Lean proofs of the two theorems shown. For the tactic proofs, 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) (9 points)

```
example : P ∧ Q → Q ∨ R := by
```

Solution

```
example : P ∧ Q → Q ∨ R := by
  intro h
  rcases h with ⟨h1, h2⟩
  left
  exact h2
```

Part b) (6 points) Write down a proof term for the theorem in part a).

Solution

```
fun h => Or.inl (h.right)
```

Part c) (9 points)

```
example : (P → Q) → (Q → R) → (P → R) := by
```

Solution

```
example : (P → Q) → (Q → R) → (P → R) := by
  intro h1 h2 h3
  apply h2
  apply h1
  exact h3
```

Part d) (6 points) Write down a proof term for the theorem in part c).

Solution

```
fun h1 h2 h3 => h2 (h1 h3)
```

Problem 5

Suppose we color the natural numbers red, green, and blue, and design a language with predicates Red(x), Green(x), and Blue(x) and a binary relation $<$ to talk about the coloring.

Part a) (8 points) Write down a first-order sentence in the language that says that every natural number has exactly one color.

Solution

$$\forall x.\, (\mathrm{Red}(x) \vee \mathrm{Green}(x) \vee \mathrm{Blue}(x)) \wedge \\ \neg(\mathrm{Red}(x) \wedge \mathrm{Green}(x)) \wedge \neg(\mathrm{Red}(x) \wedge \mathrm{Blue}(x)) \wedge \neg(\mathrm{Green}(x) \wedge \mathrm{Blue}(x))$$

Part b) (8 points) Write down a sentence that says between any two numbers that are colored red there is a number that is colored green or blue.

Solution

$$\forall x, y.\, x < y \wedge \mathrm{Red}(x) \wedge \mathrm{Red}(\mathrm{y}) \to \exists z.\, (x < z \wedge z < y \wedge (\mathrm{Green}(z) \vee \mathrm{Blue}(z)))$$

Part c) (10 points) Write down a sentence that says that there are three numbers in a row that are colored blue. (Hint: it takes some thought to say that x, y, and z are consecutive numbers using only $<$ and $=$.)

Solution

$$\exists x, y, z.\, x < y \wedge y < z \wedge \\ \forall w.\, (x < w \wedge w < z \to w = y) \wedge \mathrm{Blue}(x) \wedge \mathrm{Blue}(y) \wedge \mathrm{Blue}(z)$$

Problem 6

Part a) (10 points) Prove the following or provide a counterexample: for any two formulas $A(x)$ and $B(x)$ in first-order logic, if $\exists x.\, A(x) \wedge B(x)$ is satisfiable, then so are $\exists x.\, A(x)$ and $\exists x.\, B(x)$.

Solution

Let $\mathfrak{M}$ be any model of $\exists x.\, A(x) \wedge B(x)$. Then there is an a in $|\mathfrak{M}|$ such that A and B are both true in $\mathfrak{M}$ when x is interpreted by a. This means that $\mathfrak{M}$ is a model of $\exists x.\, A(x)$ and $\exists x.\, B(x)$ as well.

Part b) (10 points) Prove the following or provide a counterexample: for any two formulas $A(x)$ and $B(x)$ in first-order logic, if $\exists x.\, A(x)$ and $\exists x.\, B(x)$ are satisfiable, then so is $\exists x.\, A(x) \wedge B(x)$.

Solution

This is false. Let $A(x)$ be $\mathrm{Even}(x)$ and let $B(x)$ be $\neg\mathrm{Even}(x)$. Then $\exists x.\, A(x)$ and $\exists x.\, B(x)$ are both true of the natural numbers with the usual interpretation of Even, but $\exists x.\, \mathrm{Even}(x) \wedge \neg\mathrm{Even}(x)$ is false in every model.

Problem 7

In each of the following cases, find a most general unifier of the pair of expressions or explain why no such unifier exists. In all these problems, $x, y, z, \ldots$ are variables and $a, b, c, \ldots$ are constants.

Part a) (7 points) $P(a, f(a, x), g(y))$ and $P(a, f(a, f(g(b), a)), x)$.

Solution

These cannot be unified. To unify $f(a, x)$ with $f(a, f(g(b), a))$, x must unify with $f(g(b), a)$. But we also have to unify x with $g(y)$. There is no way to unify $f(g(b), a)$ with $g(y)$ because the first starts with f and the second starts with g.

Part b) (7 points) $P(f(a), g(y))$ and $P(x, g(x))$.

Solution

$$x \mapsto f(a), y \mapsto f(a).$$

Part c) (7 points) $P(f(x), g(x))$ and $P(f(f(a)), g(f(y)))$.

Solution

$$x \mapsto f(a), y \mapsto a.$$