CSE 311: Foundations of Computing I Spring 2026

Homework 4 Part 1

Due: Thursday, May 7th by 6:00 PM

Instructions

Complete the problems on the following pages.

Collaboration policy. You are required to submit your own solutions. You are allowed to discuss the homework with other students. However, you must complete the problems in Cozy on your own. Moreover, you must be able to explain your solution at any time. We reserve ourselves the right to ask you to explain your work at any time in the course of this class.

Late policy. You have a total of **three** late days during the quarter, but you can only use one late day on any one homework, giving an additional day on both parts. Please plan ahead.

Solutions submission. Submit your solution at

http://cozy.cs.washington.edu

- Each part of each task is listed as its own problem.
- You have unlimited attempts on each part.
- All completed parts get full credit and uncompleted parts get no credit.
- Make sure that you **understand** each step that Cozy performs for you. In Part 2, you will not have Cozy's help to make sure each step is performed correctly.

Task 1 – Cozy Meta Theorem [8 pts]

Let A, B, and C be sets. For each of the following claims, write a formal proof in Cozy that the claim holds.

a) $A \cap (A \cup (B \cap (B \cup C))) = A$

b) $(A \setminus B) \cup C = (A \cup C) \setminus (B \setminus C)$

Cozy's documentation for meta theorem problems is available at:

https://cozy.cs.washington.edu/static/docs/meta.html

In short. Type a rule into the text box on the line you want to extend — the top input works *forward* from the starting proposition, and the bottom input works *backward* from the ending proposition — and press Enter. Use `defof <Name>` (resp. `undef <Name>`) to apply the definition of a set operation left-to-right (resp. right-to-left), and `equiv <Name>` (resp. `uneqv <Name>`) to apply a propositional equivalence law left-to-right (resp. right-to-left). The chain is complete when the forward and backward sides meet at the same proposition.

Task 2 – Cozy Structural Induction [8 pts]

Recall the recursive definition of lists of integers from lecture:

> **Basis Step**: nil $\in$ **List**
> **Recursive Step**: for any $a \in \mathbb{Z}$ and any $L \in$ **List**, cons(a, L)[1] $\in$ **List**

For each of the following, write a formal structural induction proof in Cozy.

a) Consider the list function len, which returns the length of a list:

$$\begin{aligned} \text{len}(\text{nil}) &:= 0 \\ \text{len}(\text{cons}(a, L)) &:= 1 + \text{len}(L) \quad \forall a \in \mathbb{Z}, \forall L \in \textbf{List} \end{aligned}$$

And echo, which repeats each element in place:

$$\begin{aligned} \text{echo}(\text{nil}) &:= \text{nil} \\ \text{echo}(\text{cons}(a, L)) &:= \text{cons}(a, \text{cons}(a, \text{echo}(L))) \quad \forall a \in \mathbb{Z}, \forall L \in \textbf{List} \end{aligned}$$

Use structural induction on L to prove that echo doubles the length of its input:

$$\forall L \in \textbf{List}\, (\text{len}(\text{echo}(L)) = 2 \cdot \text{len}(L))$$

b) In addition to echo, consider the list function sum, which sums the elements of a list:

$$\begin{aligned} \text{sum}(\text{nil}) &:= 0 \\ \text{sum}(\text{cons}(a, L)) &:= a + \text{sum}(L) \quad \forall a \in \mathbb{Z}, \forall L \in \textbf{List} \end{aligned}$$

And double, which multiplies each element by two:

$$\begin{aligned} \text{double}(\text{nil}) &:= \text{nil} \\ \text{double}(\text{cons}(a, L)) &:= \text{cons}(2 \cdot a, \text{double}(L)) \quad \forall a \in \mathbb{Z}, \forall L \in \textbf{List} \end{aligned}$$

Use structural induction on L to prove that echo and double produce lists with the same sum:

$$\forall L \in \textbf{List}\, (\text{sum}(\text{echo}(L)) = \text{sum}(\text{double}(L)))$$

Cozy's documentation for structural induction problems is available at:

https://cozy.cs.washington.edu/static/docs/comfy.html

In short. At each (sub-)goal, type either "`induction on <var>`" to split the proof into one case per constructor of `<var>`'s type, or "`calculation`" to give a direct calculational proof. A calculation proof builds a chain forward from the left side of the goal and backward from the right side, meeting in the middle; at each step you can apply a function definition with `defof <name>_k` (resp. `undef <name>_k`), where `_k` indicates the case number, substitute one side of a known fact `N` into the current expression with `subst N`, or assert algebraic equality by typing "`= <expression>`" on the forward side (or "`<expression> =`" on the backward side).

[1] In lecture and elsewhere in this class, we write the recursive list constructor as $a :: L$ rather than cons(a, L) — e.g., the list $[1, 2]$ is written $1 :: 2 ::$ nil. Cozy, however, requires the functional form cons(a, L) — e.g., the same list is entered as cons$(1, \text{cons}(2, \text{nil}))$. The two notations refer to exactly the same constructor.