CSE 311: Foundations of Computing I Spring 2025

Problem Set 6

Due: Wednesday, May 21st by 11:00pm

Instructions

Submit your solutions in Gradescope. Your Gradescope submission should follow these rules:

- Each numbered task should be solved on its own page (or pages). Do not write your name on the individual pages. (Gradescope will handle that.)
- When you upload your pages, make sure each one is **properly rotated**. If not, you can use the Gradescope controls to turn them to the proper orientation.
- Follow the Gradescope prompt to **link tasks to pages**. You do not need to link tasks that you did not include, e.g., Task 7 (extra credit).
- You are not required to typeset your solution, but your submission must be **legible**. It is your responsibility to make sure solutions are readable — we will *not* grade unreadable write-ups.

Task 1 – Sub(set)way [16 pts]

After the last Mod(ulo) Pizza escapade, the TA's have decided that they've had enough of pizza for the next x years for some $x \in \mathbb{N}$ (or something along those lines). Consequently, the TA's have decided to pursue something new: Sub(set)way, a well established sub(set) sandwich establishment! However, the TA's can't figure out how to split up the change for their huge group order... That's where you come in. We have encoded our change debacle in a subset proof, and we need you to prove it. Last time, we "lied" and said we'd save a slice of pizza - but this time we REALLY "promise" we'll get you a sandwich, should you so choose to help!

Let A and B be the following sets:

$$A := \{n \in \mathbb{Z} : n \equiv_6 4\}$$
$$B := \{n \in \mathbb{Z} : n^2 \equiv_6 4\}$$

Now, consider the following claim:

$$A \subseteq B$$

Write an **English proof** that the claim holds.

Follow the structure of our template for subset proofs.

Note: even though we want you to write your proof directly in English, it must still look like the translation of a formal proof. In particular, you must include all steps that would be required of a formal proof, excepting only those that we have explicitly said are okay to skip in English proofs (e.g., Elim $\exists$).

Task 2 – Keeping Up With the Cartesians [16 pts]

Let A, B, and C be sets. Consider the following claim:

$$B \times A \subseteq C \times B$$

a) Suppose that $A = \{1\}$, $B = \{1, 2\}$, and $C = \{1, 2, 3\}$.

Calculate the values of the sets $B \times A$ and $C \times B$. Check whether the claim holds.

b) Suppose that $A = \{1, 2\}$, $B = \{1\}$, and $C = \{1, 2, 3\}$.

Calculate the values of the sets $B \times A$ and $C \times B$. Check whether the claim holds.

c) Write an **English proof** that the claim holds given that $A \subseteq B$ and $B \subseteq C$.

(This updated claim describes the situation in part (a) but not part (b).)

Follow the structure of our template for subset proofs.

Note: even though we want you to write your proof directly in English, it must still look like the translation of a formal proof. In particular, you must include all steps that would be required of a formal proof, excepting only those that we have explicitly said are okay to skip in English proofs.

Task 3 – Our Finest Power [16 pts]

Let A, B, and C be sets. Consider the following claim:

$$\mathcal{P}(A \cap (B \cup C)) \subseteq \mathcal{P}(A \cap B) \cup \mathcal{P}(A \cap C)$$

a) Suppose that $A = \{1, 2\}$, $B = \{1, 3\}$, and $C = \{2, 4\}$.

Calculate the values of the sets $\mathcal{P}(A \cap (B \cup C))$ and $\mathcal{P}(A \cap B) \cup \mathcal{P}(A \cap C))$. Check whether the claim holds.

b) Suppose that $A = \{1\}$, $B = \{1, 2\}$, and $C = \{1, 3\}$.

Calculate the values of the sets $\mathcal{P}(A \cap (B \cup C))$ and $\mathcal{P}(A \cap B) \cup \mathcal{P}(A \cap C))$. Check whether the claim holds.

c) Write an **English proof** that the claim holds given $(A \cap B) \subseteq (A \cap C)$ hold.

(This updated claim describes the situation in part (b) but not part (a). The claim holds when either $(A \cap B) \subseteq (A \cap C)$ or $(A \cap C) \subseteq (A \cap B)$ hold, but the proof is the same for both cases thus one was omitted. Think about why that would intuitively make sense!)

Follow the structure of our template for subset proofs.

In your proof, you are free to use (cite or apply) the following theorems about sets:

Transitivity of Subset: $\forall A \, \forall B \, \forall C \, (((A \subseteq B) \land (B \subseteq C)) \to (A \subseteq C))$

Distributivity of Set: $\forall A \, \forall B \, \forall C \, (A \cap (B \cup C) = (A \cap B) \cup (A \cap C))$

Note: even though we want you to write your proof directly in English, it must still look like the translation of a formal proof. In particular, you must include all steps that would be required of a formal proof, excepting only those that we have explicitly said are okay to skip in English proofs.

Task 4 – Parmesian, Romano, and Meta [12 pts]

Let A, B, and C be sets. For each of the following claims:

1. **State** whether the the claim is true or false.
2. If the claim is true, write an **English proof** that the claim holds following the Meta Theorem *template*. (In your equivalence chain, you can skip steps showing commutativity or associativity, as long as each step is easy to follow.)
3. If it the claim false, give a **counterexample**. Provide specific finite sets for A, B, and C, and then calculate the value of each side of the claim, showing that they do not produce the same set. (Be sure to show the value of each intermediate expression, when calculating each side.)

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

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

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

Task 5 – List Me By a Mile [20 pts]

Recall the definition of lists of numbers from lecture:

Basis Step: $\mathsf{nil} \in$ **List**
Recursive Step: for any $a \in \mathbb{Z}$, if $L \in$ **List**, then $a :: L \in$ **List**.

For example, the list $[1, 2, 3]$ would be created recursively from the empty list as $1 :: (2 :: (3 :: \mathsf{nil}))$. We will consider "::" to associate to the right, so $1 :: 2 :: 3 :: \mathsf{nil}$ means the same thing.

The parts below use two recursively-defined functions. The first is sum, which calculates the sum of the list. It is defined recursively as follows:

$$\begin{array}{rcll} \mathsf{sum}(\mathsf{nil}) & := & 0 & \\ \mathsf{sum}(a :: L) & := & a + \mathsf{sum}(L) & \forall a \in \mathbb{Z}, \forall L \in \textbf{List} \end{array}$$

The second function, positives, which returns only the positive numbers in the list, is defined by:

$$\begin{array}{rcll} \mathsf{positives}(\mathsf{nil}) & := & \mathsf{nil} & \\ \mathsf{positives}(a :: L) & := & \mathsf{positives}(L) \quad \text{if } a \leqslant 0 & \forall a \in \mathbb{Z}, \forall L \in \textbf{List} \\ \mathsf{positives}(a :: L) & := & a :: \mathsf{positives}(L) \quad \text{if } a > 0 & \forall a \in \mathbb{Z}, \forall L \in \textbf{List} \end{array}$$

For example, from these definitions, we get $\mathsf{positives}(-1 :: 2 :: -3 :: \mathsf{nil}) = 2 :: \mathsf{nil}$.

a) Write a calculation block, citing the appropriate definitions, showing that

$$\mathsf{positives}(2 :: -4 :: 6 :: \mathsf{nil}) = 2 :: 6 :: \mathsf{nil}$$

b) Write a calculation block, citing the appropriate definitions, showing that

$$\mathsf{sum}(-1 :: 3 :: -5 :: \mathsf{nil}) = -3$$

c) Use structural induction to prove that

$$\forall L \in \textbf{List}\, (\mathsf{sum}(\mathsf{positives}(L)) \geqslant \mathsf{sum}(L))$$

Task 6 – Node to Self: Bound the Leaves! [16 pts]

Recall the definition of rooted binary trees (with no data) from lecture:

Basis Step: $\bullet \in$ **Tree**
Recursive Step: if $L \in$ **Tree** and $R \in$ **Tree**, then $\mathsf{Tree}(L, R) \in$ **Tree**.

Note that, in lecture, we drew "$\mathsf{Tree}(L, R)$" as a picture of a tree, whereas here we are using more normal (functional) notation, which should be easier for calculations.

Define the height of a tree as follows:

$$\begin{aligned} \mathsf{height}(\bullet) &= 0 \\ \mathsf{height}(\mathsf{Tree}(L, R)) &= 1 + \max(\mathsf{height}(L), \mathsf{height}(R)) \end{aligned}$$

Define the number of leaves in a tree as follows:

$$\begin{aligned} \mathsf{leaves}(\bullet) &= 1 \\ \mathsf{leaves}(\mathsf{Tree}(L, R)) &= \mathsf{leaves}(L) + \mathsf{leaves}(R) \end{aligned}$$

You are recommended to draw out some binary trees of your own, write their functional notation representations, calculate the height and leaves counts in these representations, and verify the results match the height and leaves of the trees you drew. This will help you build intuition for why the inductive definitions of height and leaves works.

Now, consider the following claim about leaves and height:

$$\forall\, T \in \textbf{Tree}\, (\mathsf{leaves}(T) \leqslant 2^{\mathsf{height}(T)})$$

This places an exponential bound on the number of leaves a binary tree can have with respect to its height. Prove this claim by structural induction.

Task 7 – Extra Credit: Walk Like an Encryption [0 pts]

As we discussed in Topic 4, an important advantage of modular arithmetic over ordinary arithmetic is that the size of the output is fixed. The output of $(a+b) \bmod m$ or $ab \bmod m$ is always between 0 and $m-1$. This size limitation is even more important for exponentiation: a^k could require exponentially more bits than a or k to store, whereas $a^k \bmod m$ is, once again, between 0 and $m-1$.

It turns out that modular exponentiation is also extremely important in practice. In particular, it is the key step of the RSA public-key encryption system. Briefly, if Bob wishes to send a secret number a to Alice, he will calculate $b = a^k \bmod m$, where k and m are public numbers published by Alice. Alice receives b and then calculates $c = b^\ell \bmod m$ for where ℓ is a private number known only to her. The numbers k and ℓ are chosen so that we always have $c = a$. Furthermore, calculating a without knowing the number ℓ is believed to be exponentially expensive on current computers.

How do we choose k and ℓ to have this property? To figure this out, we need some facts about modular exponentiation....

We know that we can reduce the *base* of an exponent modulo m : $a^k \equiv_m (a \bmod m)^k$. But the same is not true of the exponent! That is, we cannot write $a^k \equiv_m a^{k \bmod m}$. This is easily seen to be false in general. Consider, for instance, that $2^{10} \bmod 3 = 1$ but $2^{10 \bmod 3} \bmod 3 = 2^1 \bmod 3 = 2$.

The correct law for the exponent is more subtle. We will prove it in steps....

(a) Let $R = \{n \in \mathbb{Z} : 1 \leqslant n \leqslant m-1 \wedge \gcd(n,m) = 1\}$. Define the set $aR = \{ax \bmod m : x \in R\}$. Prove that $aR = R$ for every integer $a > 0$ with $\gcd(a,m) = 1$.

(b) Consider the product of all the elements in R modulo m and the elements in aR modulo m. By comparing those two expressions, conclude that, for all $a \in R$, we have $a^{\phi(m)} \equiv_m 1$, where $\phi(m) = |R|$.

(c) Use the last result to show that, for any $b \geqslant 0$ and $a \in R$, we have $a^b \equiv_m a^{b \bmod \phi(m)}$.

(d) Finally, prove the following two facts about the function ϕ above. First, if p is prime, then $\phi(p) = p-1$. Second, for any primes a and b with $a \neq b$, we have $\phi(ab) = \phi(a)\phi(b)$. (Or slightly more challenging: show this second claim for *all positive integers* a and b with $\gcd(a,b) = 1$.)

The second fact of part (d) implies that, if p and q are primes, then $\phi(pq) = (p-1)(q-1)$.

How does that help us choose k and ℓ? Since $c = b^\ell \bmod m$ and $b = a^k \bmod m$, we can see that $c \equiv_m (a^k)^\ell \equiv_m a^{k\ell}$. By what we just learned, we can see that $a^{k\ell} \equiv_m a^{k\ell \bmod \varphi(m)}$. Thus, if we choose k and ℓ to be multiplicative inverses modulo $\varphi(m)$, so that we have $k\ell \equiv_{\varphi(m)} 1$, then we have $c \equiv_m a^{k\ell} \equiv_m a^{k\ell \bmod \varphi(m)} \equiv_m a^1 = a$, showing that Alice does indeed calculate the secret number that Bob was hoping to send her!