CSE 390Z: Mathematics for Computation Workshop

Week 8 Workshop

Conceptual Review

(a) **Set Definitions**

Set Equality: $A = B := \forall x(x \in A \leftrightarrow x \in B)$
Subset: $A \subseteq B := \forall x(x \in Ax \in B)$
Union: $A \cup B := \{x \;:\; x \in A \vee x \in B\}$
Intersection: $A \cap B := \{x \;:\; x \in A \wedge x \in B\}$
Set Difference: $A \setminus B = A - B := \{x \;:\; x \in A \wedge x \notin B\}$
Set Complement: $\overline{A} = A^C := \{x \;:\; x \notin A\}$
Powerset: $\mathcal{P}(A) := \{B \;: B \subseteq A\}$
Cartesian Product: $A \times B := \{(a, b) \;: a \in A,\; b \in B\}$

(b) How do we prove that for sets A and B, $A \subseteq B$?

(c) How do we prove that for sets A and B, $A = B$?

1. A Basic Subset Proof

Prove that $A \cap B \subseteq A \cup B$.

2. Set Equality Proof

(a) Write an English proof to show that $A \cap (A \cup B) \subseteq A$ for any sets A, B.

(b) Write an English proof to show that $A \subseteq A \cap (A \cup B)$ for any sets A, B.

(c) Combine part (a) and (b) to conclude that $A \cap (A \cup B) = A$ for any sets A, B.

3. Subsets

Prove or disprove: for any sets A, B, and C, if $A \subseteq B$ and $B \subseteq C$, then $A \subseteq C$.

4. $\cup \to \cap$?

Prove or disprove: for all sets A and B, $A \cup B \subseteq A \cap B$.

5. Set Equality Proof II

We want to prove that $A \setminus (B \cap C) = (A \setminus B) \cup (A \setminus C)$.

(a) First prove this with a chain of logical equivalences proof.

(b) Now prove this with an English proof that is made of two subset proofs.

6. Cartesian Product Proof

Write an English proof to show that $A \times C \subseteq (A \cup B) \times (C \cup D)$.

7. Structural Induction: Divisible by 4

Define a set $\mathfrak{B}$ of numbers by:

- 4 and 12 are in $\mathfrak{B}$
- If $x \in \mathfrak{B}$ and $y \in \mathfrak{B}$, then $x + y \in \mathfrak{B}$ and $x - y \in \mathfrak{B}$

Prove by induction that every number in $\mathfrak{B}$ is divisible by 4.

Complete the proof below:

8. Structural Induction: CharTrees

Recursive Definition of CharTrees:

- Basis Step: `Null` is a **CharTree**
- Recursive Step: If L, R are **CharTree**s and $c \in \Sigma$, then $\texttt{CharTree}(L, c, R)$ is also a **CharTree**

Intuitively, a **CharTree** is a tree where the non-null nodes store a `char` data element.

Recursive functions on CharTrees:

- The preorder function returns the preorder traversal of all elements in a **CharTree**.

$$\begin{aligned} \text{preorder}(\texttt{Null}) &= \varepsilon \\ \text{preorder}(\texttt{CharTree}(L, c, R)) &= c \cdot \text{preorder}(L) \cdot \text{preorder}(R) \end{aligned}$$

- The postorder function returns the postorder traversal of all elements in a **CharTree**.

$$\begin{aligned} \text{postorder}(\texttt{Null}) &= \varepsilon \\ \text{postorder}(\texttt{CharTree}(L, c, R)) &= \text{postorder}(L) \cdot \text{postorder}(R) \cdot c \end{aligned}$$

- The mirror function produces the mirror image of a **CharTree**.

$$\begin{aligned} \text{mirror}(\texttt{Null}) &= \texttt{Null} \\ \text{mirror}(\texttt{CharTree}(L, c, R)) &= \texttt{CharTree}(\text{mirror}(R), c, \text{mirror}(L)) \end{aligned}$$

- Finally, for all strings x, let the "reversal" of x (in symbols x^R) produce the string in reverse order.

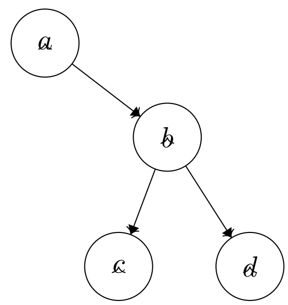
Additional Facts:
You may use the following facts:

- For any strings $x_1, ..., x_k$: $(x_1 \cdot ... \cdot x_k)^R = x_k^R \cdot ... \cdot x_1^R$
- For any character c, $c^R = c$

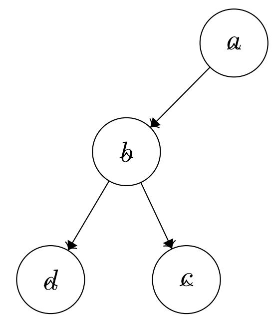
Statement to Prove:
Show that for every **CharTree** T, the reversal of the preorder traversal of T is the same as the postorder traversal of the mirror of T. In notation, you should prove that for every **CharTree**, T: $[\text{preorder}(T)]^R = \text{postorder}(\text{mirror}(T))$.

There is an example and space to work on the next page.

Example for Intuition:



Let T_i be the tree above.
preorder(T_i) =“abcd”.
T_i is built as $(\texttt{null}, a, U)$
Where U is (V, b, W),
$V = (\texttt{null}, c, \texttt{null}), W = (\texttt{null}, d, \texttt{null})$.



This tree is mirror(T_i).
postorder(mirror(T_i)) =“dcba”,
“dcba” is the reversal of “abcd” so
[preorder(T_i)]^R = postorder(mirror(T_i)) holds for T_i