CSE 390Z: Mathematics for Computing Workshop

Week 8 Workshop Solutions

0. Structural Induction: Strings

Recursive Definition of a String:

- Basis Step: ϵ is a string
- Recursive Step: If w is a string and a is a character, $w \bullet a$ is a string (the string w with the character a appended on to the end)

Recursive functions on String:

Length:

$$\begin{aligned}\mathsf{len}(\epsilon) &= 0 \\ \mathsf{len}(w \bullet a) &= \mathsf{len}(a \bullet w) = 1 + \mathsf{len}(w)\end{aligned}$$

Reverse:

$$\begin{aligned}\mathsf{rev}(\epsilon) &= \epsilon \\ \mathsf{rev}(w \bullet a) &= a \bullet \mathsf{rev}(w)\end{aligned}$$

Prove that for any string x, $\mathsf{len}(\mathsf{rev}(x)) = \mathsf{len}(x)$.

Solution:

For a string x, let $\mathsf{P}(x)$ be "$\mathsf{len}(\mathsf{rev}(x)) = \mathsf{len}(x)$". We will prove $\mathsf{P}(x)$ for all strings x by structural induction on the set of strings.

Base Case ($x = \epsilon$): By definition of reverse, $\mathsf{len}(\mathsf{rev}(\epsilon)) = \mathsf{len}(\epsilon)$. So $\mathsf{P}(\epsilon)$ holds.

Let s be an arbitrary string not covered by the base case. Then by the exclusion rule, $s = w \bullet a$ for some string w and some character a.

Inductive Hypothesis: Suppose $\mathsf{P}(w)$ holds. Then $\mathsf{len}(\mathsf{rev}(w)) = \mathsf{len}(w)$.

Inductive Step: $\boxed{\text{Goal: Show that } \mathsf{P}(w \bullet a) \text{ holds}}$

$$\begin{aligned}\mathsf{len}(\mathsf{rev}(w \bullet a)) &= \mathsf{len}(a \bullet \mathsf{rev}(w)) && \text{[By Definition of reverse]} \\ &= 1 + \mathsf{len}(\mathsf{rev}(w)) && \text{[By Definition of length]} \\ &= 1 + \mathsf{len}(w) && \text{[By IH]} \\ &= \mathsf{len}(w \bullet a) && \text{[By Definition of length]}\end{aligned}$$

This proves $\mathsf{P}(w \bullet a)$.

Conclusion: $\mathsf{P}(x)$ holds for all strings x by structural induction.

1. Structural Induction: CharTrees

Recursive Definition of CharTrees:

- Basis Step: `Null` is a **CharTree**
- Recursive Step: If L, R are **CharTrees** 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.

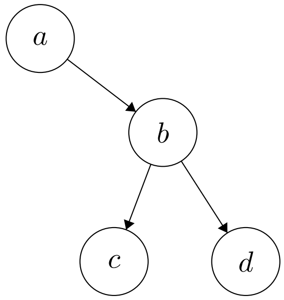
Additional Facts:
You may use the following facts:

- 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, $c^R = c$

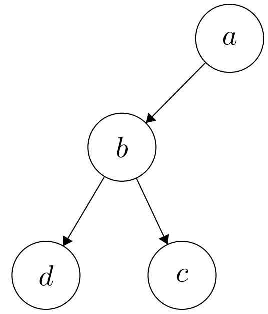
Statement to Prove:
Show that for every **CharTree** T, the reversal of the preorder traversal of T is the same as the postorder traversal of the mirror of T. In notation, you should prove that for every **CharTree**, T: $[\text{preorder}(T)]^R = \text{postorder}(\text{mirror}(T))$.

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

Solution:

Let $P(T)$ be “[preorder(T)]^R = postorder(mirror(T))”. We show $P(T)$ holds for all **CharTree**s T by structural induction.

Base case ($T = \texttt{Null}$): preorder$(T)^R = \varepsilon^R = \varepsilon =$ postorder(Null) = postorder(mirror(Null)), so $P(\texttt{Null})$ holds.

Let T be an arbitrary **CharTree** not covered by the base case. By the exclusion rule, $T = \texttt{CharTree}(L, c, R)$ for some **CharTrees** L, R.

Inductive hypothesis: Suppose $P(L) \wedge P(R)$.

Inductive step: $\boxed{\text{Goal: Show } P(T)\text{, i.e. [preorder}(T)]^R = \text{postorder(mirror}(T)).}$

$$
\begin{aligned}
\text{preorder}(T)^R &= \text{preorder}(\texttt{CharTree}(L, c, R))^R && \text{defn of } T\\
&= [c \cdot \text{preorder}(L) \cdot \text{preorder}(R)]^R && \text{defn of preorder}\\
&= \text{preorder}(R)^R \cdot \text{preorder}(L)^R \cdot c^R && \text{Fact 1}\\
&= \text{preorder}(R)^R \cdot \text{preorder}(L)^R \cdot c && \text{Fact 2}\\
&= \text{postorder}(\text{mirror}(R)) \cdot \text{postorder}(\text{mirror}(L)) \cdot c && \text{by I.H.}\\
&= \text{postorder}(\text{CharTree}(\text{mirror}(R), c, \text{mirror}(L)) && \text{recursive defn of postorder}\\
&= \text{postorder}(\text{mirror}(\text{CharTree}(L, c, R))) && \text{recursive defn of mirror}\\
&= \text{postorder}(\text{mirror}(T)) && \text{defn of } T
\end{aligned}
$$

So $P(\texttt{CharTree}(L, c, R))$ holds.
By the principle of induction, $P(T)$ holds for all **CharTree**s T.

2. Structural Induction: Dictionaries

Recursive definition of a Dictionary (i.e. a Map):

- Basis Case: [] is the empty dictionary
- Recursive Case: If D is a dictionary, and a and b are elements of the universe, then $(a \to b)$ :: D is a dictionary that maps a to b (in addition to the content of D).

Recursive functions on Dictionaries:

$$\begin{aligned} \text{AllKeys}([\,]) &= [\,] \\ \text{AllKeys}((a \to b) :: \text{D}) &= a :: \text{AllKeys}(\text{D}) \end{aligned}$$

$$\begin{aligned} \text{len}([\,]) &= 0 \\ \text{len}((a \to b) :: \text{D}) &= 1 + \text{len}(\text{D}) \end{aligned}$$

Recursive functions on Sets:

$$\begin{aligned} \text{len}([\,]) &= 0 \\ \text{len}(a :: \text{C}) &= 1 + \text{len}(\text{C}) \end{aligned}$$

Statement to prove:
Prove that len(D) = len(AllKeys(D)).

Solution:

Proof. Define P(D) to be len(D) = len(AllKeys(D)) for a Dictionary D. We will use structural induction to show P(D) for all dictionaries D.

Base Case: D = []:
len(D) = len([]) = 0 by definition of dictionary len.
Since AllKeys([]) = [] by definition of AllKeys, len(AllKeys(D)) = len([]) = 0 by definition of set len.
Since $0 = 0$, P([]) is true.

Let C be an arbitrary dictionary not covered by the base case. By the exclusion rule, C must be of the form $(a \to b$ :: B) for a dictionary B.
Inductive Hypothesis: Suppose P(B). That is, len(B) = len(AllKeys(B)).
Inductive Step: $\boxed{\text{Goal: Show P(C), i.e. len(C) = len(AllKeys(C))}}$

$$\begin{aligned} \text{len}(C) &= \text{len}((a \to b) :: \text{B}) && \text{[Definition of C]} \\ &= 1 + \text{len}(\text{B}) && \text{[Definition of Len]} \\ &= 1 + \text{len}(\text{AllKeys}(\text{B})) && \text{[IH]} \\ &= \text{len}(a :: \text{AllKeys}(\text{B})) && \text{[Definition of Len]} \\ &= \text{len}(\text{AllKeys}((a \to b) :: \text{B})) && \text{[Definition of AllKeys]} \\ &= \text{len}(\text{AllKeys}(\text{C})) && \text{[Definition of C]} \end{aligned}$$

So P(C) holds.
Conclusion: Thus, the claim holds for all dictionaries D by structural induction. □

3. Structural Induction: CFGs

Consider the following CFG:

$$S \to SS \mid 0S1 \mid 1S0 \mid \epsilon$$

Prove that every string generated by this CFG has an equal number of 1's and 0's.

Hint 1: Start by converting this CFG to a recursively defined set.
Hint 2: You may wish to define the functions $\#_0(x), \#_1(x)$ on a string x.

Solution:

First we observe that the language defined by this CFG can be represented by a recursively defined set. Define a set S as follows:
Basis Rule: $\epsilon \in S$
Recursive Rule: If $x, y \in S$, then $0x1, 1x0, xy \in S$.

Now we perform structural induction on the recursively defined set. Define the functions $\#_0(t), \#_1(t)$ to be the number of 0's and 1's respectively in the string t.

Proof. For a string t, let P(t) be defined as "$\#_0(t) = \#_1(t)$". We will prove P(t) is true for all strings $t \in S$ by structural induction.

Base Case ($t = \epsilon$): By definition, the empty string contains no characters, so $\#_0(t) = 0 = \#_1(t)$

Let s be an arbitrary string in S not covered by the base case. By the exclusion rule, $s = 0x1$ or $s = 1x0$ or $s = xy$ for some strings x, y.

Inductive Hypothesis: Suppose P(x) and P(y) hold.

Inductive Step: $\boxed{\text{Goal: Prove } P(s).}$
Case 1: $s = 0x1$
By the IH, $\#_0(x) = \#_1(x)$. Then observe that:

$$\#_0(0x1) = \#_0(x) + 1 = \#_1(x) + 1 = \#_1(0x1)$$

Therefore $\#_0(0x1) = \#_1(0x1)$. This proves P$(0x1)$.

Case 2: $s = (1x0)$
By the IH, $\#_0(x) = \#_1(x)$. Then observe that:

$$\#_0(1x0) = \#_0(x) + 1 = \#_1(x) + 1 = \#_1(1x0)$$

Therefore $\#_0(1x0) = \#_1(1x0)$. This proves P$(1x0)$.

Case 3: $s = xy$
By the IH, $\#_0(x) = \#_1(x)$ and $\#_0(y) = \#_1(y)$. Then observe that:

$$\#_0(xy) = \#_0(x) + \#_0(y) = \#_1(x) + \#_1(y) = \#_1(xy)$$

Therefore $\#_0(xy) = \#_1(xy)$. This proves P(xy).
In all cases, $P(s)$ hold.

So by structural induction, P(t) is true for all strings $t \in S$. □

Since the recursively defined set, S, is exactly the set of strings generated by the CFG, we have proved that the statement is true for every string generated by the CFG too.