0. Structural Induction: Strings

Recursive Definition of a String:
- Basis Step: $\epsilon$ is a string
- Recursive Step: If $w$ is a string and $a$ is a character, $w \cdot a$ is a string (the string $w$ with the character $a$ appended on to the end)

Recursive functions on String:
Length:
\[
\begin{align*}
\text{len}(\epsilon) &= 0 \\
\text{len}(w \cdot a) &= \text{len}(a \cdot w) = 1 + \text{len}(w)
\end{align*}
\]
Reverse:
\[
\begin{align*}
\text{rev}(\epsilon) &= \epsilon \\
\text{rev}(w \cdot a) &= a \cdot \text{rev}(w)
\end{align*}
\]

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

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

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

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

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

Inductive Step: Goal: Show that $P(w \cdot a)$ holds
\[
\begin{align*}
\text{len}(\text{rev}(w \cdot a)) &= \text{len}(a \cdot \text{rev}(w)) \quad \text{[By Definition of reverse]} \\
&= 1 + \text{len}(\text{rev}(w)) \quad \text{[By Definition of length]} \\
&= 1 + \text{len}(w) \quad \text{[By IH]} \\
&= \text{len}(w \cdot a) \quad \text{[By Definition of length]}
\end{align*}
\]
This proves $P(w \cdot a)$.

Conclusion: $P(x)$ 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 \( \text{CharTree}(L, c, R) \) 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.
  
  \[
  \text{preorder}(\text{Null}) = \epsilon \\
  \text{preorder}(\text{CharTree}(L, c, R)) = c \cdot \text{preorder}(L) \cdot \text{preorder}(R)
  \]

- The postorder function returns the postorder traversal of all elements in a CharTree.
  
  \[
  \text{postorder}(\text{Null}) = \epsilon \\
  \text{postorder}(\text{CharTree}(L, c, R)) = \text{postorder}(L) \cdot \text{postorder}(R) \cdot c
  \]

- The mirror function produces the mirror image of a CharTree.
  
  \[
  \text{mirror}(\text{Null}) = \text{Null} \\
  \text{mirror}(\text{CharTree}(L, c, R)) = \text{CharTree}(\text{mirror}(R), c, \text{mirror}(L))
  \]

- 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 \( x_1, \ldots, x_k \): \( (x_1 \cdot \ldots \cdot x_k)^R = x_k^R \cdot \ldots \cdot x_1^R \)
- 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.

$\text{preorder}(T_i) = \text{"abcd"}$.  
$T_i$ is built as $(\text{null}, a, U)$  
Where $U$ is $(V, b, W)$,  
$V = (\text{null}, c, \text{null})$, $W = (\text{null}, d, \text{null})$.

This tree is mirror$(T_i)$.  
$\text{postorder}(\text{mirror}(T_i)) = \text{"dcba"}$,  
"dcba" is the reversal of \text{"abcd"} so  
$[\text{preorder}(T_i)]^R = \text{postorder}(\text{mirror}(T_i))$ holds for $T_i$.

Solution:

Let $P(T)$ be \text{"[preorder(T)]^R = postorder(mirror(T))"}. We show $P(T)$ holds for all CharTrees $T$ by structural induction.

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

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

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

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

\[
\begin{align*}
\text{preorder}(T)^R &= \text{preorder}(\text{CharTree}(L, c, R))^R \\
&= [c \cdot \text{preorder}(L) \cdot \text{preorder}(R)]^R \\
&= \text{preorder}(R)^R \cdot \text{preorder}(L)^R \cdot c^R \\
&= \text{preorder}(R)^R \cdot \text{preorder}(L)^R \cdot c \\
&= \text{postorder}(\text{mirror}(R)) \cdot \text{postorder}(\text{mirror}(L)) \cdot c \\
&= \text{postorder}(\text{mirror} \text{CharTree}(\text{mirror}(R), c, \text{mirror}(L))) \text{ recursive defn of postorder} \\
&= \text{postorder}(\text{mirror}(\text{CharTree}(L, c, R))) \text{ recursive defn of mirror} \\
&= \text{postorder}(\text{mirror}(T)) \text{ defn of } T
\end{align*}
\]

So $P(\text{CharTree}(L, c, R))$ holds.

By the principle of induction, $P(T)$ holds for all CharTrees $T$. 

3
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 → b) :: D is a dictionary that maps a to b (in addition to the content of D).

Recursive functions on Dictionaries:

\[
\begin{align*}
\text{AllKeys}([ ]) &= [] \\
\text{AllKeys}((a \rightarrow b) :: D) &= a :: \text{AllKeys}(D) \\
\text{len}([ ]) &= 0 \\
\text{len}((a \rightarrow b) :: D) &= 1 + \text{len}(D)
\end{align*}
\]

Recursive functions on Sets:

\[
\begin{align*}
\text{len}([ ]) &= 0 \\
\text{len}(a :: C) &= 1 + \text{len}(C)
\end{align*}
\]

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 = []:
\[
\text{len}(D) = \text{len}([ ]) = 0 \text{ by definition of dictionary len.}
\]
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 → b) :: B for a dictionary B.

Inductive Hypothesis: Suppose P(B). That is, len(B) = len(AllKeys(B)).

Inductive Step: Goal: Show P(C), i.e. len(C) = len(AllKeys(C))

\[
\begin{align*}
\text{len}(C) &= \text{len}((a \rightarrow b) :: B) \quad \text{[Definition of C]} \\
&= 1 + \text{len}(B) \quad \text{[Definition of Len]} \\
&= 1 + \text{len}(\text{AllKeys}(B)) \quad \text{[IH]} \\
&= \text{len}(a :: \text{AllKeys}(B)) \quad \text{[Definition of Len]} \\
&= \text{len}(\text{AllKeys}((a \rightarrow b) :: B)) \quad \text{[Definition of AllKeys]} \\
&= \text{len}(\text{AllKeys}(C)) \quad \text{[Definition of C]}
\end{align*}
\]

So P(C) holds.

Conclusion: Thus, the claim holds for all dictionaries D by structural induction. □
3. Structural Induction: CFGs

Consider the following CFG:

\[ S \rightarrow SS \mid 0S1 \mid 1S0 \mid \epsilon \]

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 \( \#_0(\epsilon) = 0 = \#_1(\epsilon) \)

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:** 

**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.