CSE 390Z: Mathematics of Computing

Week 8 Workshop Solutions

0. 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.

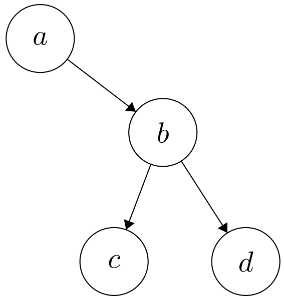
Additional Facts:
You may use the following facts:

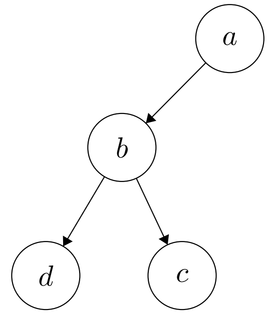
- For any strings $x_1, ..., x_k$: $(x_1 \cdot ... \cdot x_k)^R = x_k^R \cdot ... \cdot x_1^R$
- For any character c, $c^R = c$

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

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($\texttt{Null}$) = postorder(mirror($\texttt{Null}$)), so $P(\texttt{Null})$ holds.

Inductive hypothesis: Suppose $P(L) \wedge P(R)$ for arbitrary **CharTree**s L, R.

Inductive step: We want to show $P(\texttt{CharTree}(L, c, R))$,
i.e. [preorder($\texttt{CharTree}(L, c, R)$)]^R = postorder(mirror($\texttt{CharTree}(L, c, R)$)).
Let c be an arbitrary element in Σ, and let $T = \texttt{CharTree}(L, c, R)$

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

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

1. 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:

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

Reverse:

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

Statement to Prove:

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 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.

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

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

Let a be an arbitrary character.

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

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

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

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:

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

Recursive functions on Sets:

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

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 go by structural induction to show P(D) for all dictionaries D.
Base Case: D = []: Note that:

$$\begin{aligned} \mathsf{len}(\mathsf{D}) &= \mathsf{len}(\texttt{[]}) \\ &= \mathsf{len}(\mathsf{AllKeys}(\texttt{[]})) && \text{[Definition of AllKeys]} \\ &= \mathsf{len}(\mathsf{AllKeys}(\mathsf{D})) \end{aligned}$$

Inductive Hypothesis: Suppose P(C) to be true for an arbitrary dictionary C.
Inductive Step:
Let D' = $(a \to b)$ :: C. Note that:

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

So P(D') 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: 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 $\mathsf{P}(t)$ be defined as "$\#_0(t) = \#_1(t)$". We will prove $\mathsf{P}(t)$ is true for all strings $t \in S$ by structural induction.

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

Inductive Hypothesis: Suppose $\mathsf{P}(x)$, $\mathsf{P}(y)$ hold for some arbitrary strings x, y.

Inductive Step:
Case 1: Goal is to show $\mathsf{P}(0x1)$ holds.
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 $\mathsf{P}(0x1)$.

Case 2: Goal is to show $\mathsf{P}(1x0)$ holds.
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 $\mathsf{P}(1x0)$.

Case 3: Goal is to show $\mathsf{P}(xy)$ holds.
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 $\mathsf{P}(xy)$.

So by structural induction, $\mathsf{P}(t)$ is true for all strings $t \in S$. $\square$

4. Regular Expressions

(a) Consider the following Regular Expression (RegEx):

$$1(45 \cup 54)^{\star}1$$

List 5 strings accepted by the RegEx and 5 strings from $T := \{1, 4, 5\}^{\star}$ rejected by the RegEx. Then, summarize this RegEx in your own words.

Solution:

Accepted:

- 1451
- 1541
- 145541
- 1454545451
- 11

Rejected:

- 1
- 1441
- 45
- 14451
- 111

This RegEx accepts exactly those strings that start and end with a 1, and have one or more pairs of 45 or 54 in the middle.

(b) Consider the following Regular Expression (RegEx):

$$0^{\star}(0 \cup 1)^{\star}((01) \cup (11) \cup (10) \cup (00))1^{\star}(0 \cup 1)^{\star}$$

List 3 strings accepted by the RegEx and 3 strings from $S := \{0, 1\}^{\star}$ rejected by the RegEx. Then, summarize this RegEx in your own words and write a simpler RegEx that accepts exactly the same set of strings.

Solution:

Accepted:

- 01
- 10
- 10100100101

Rejected:

- ϵ
- 0
- 1

This RegEx accepts all binary strings that are 2 or more characters long. A simpler RegEx for this is $(0 \cup 1)(0 \cup 1)(0 \cup 1)^{\star}$.

5. Constructing Regular Expressions

For each of the following, construct a regular expression for the specified language.

(a) Strings from the language $S := \{a\}^*$ with an even number of a's.

Solution:

$(aa)^*$

(b) Strings from the language $S := \{a, b\}^*$ with an even number of a's.

Solution:

$b^*(b^*ab^*ab^*)^*$

(c) Strings from the language $S := \{a, b\}^*$ with odd length.

Solution:

$(aa \cup ab \cup ba \cup bb)^*(a \cup b)$

(d) (Challenge) Strings from the language $S := \{a, b\}^*$ with an even number of a's or an odd number of b's.

Solution:

$b^*(b^*ab^*ab^*)^* \cup (a^* \cup a^*ba^*ba^*)^*b(a^* \cup a^*ba^*ba^*)^*$