CSE 390Z: Mathematics of Computing Workshop

Week 8 Workshop Solutions

0. Conceptual Review

(a) Regular expression rules:
Basis: ϵ, a for $a \in \Sigma$
Recursive: If A, B are regular expressions, $(A \cup B), AB$, and A^* are regular expressions.

1. 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:

Solution:

Let $P(b)$ be the claim that $4 \mid b$. We will prove $P(b)$ is true for all numbers $b \in \mathfrak{B}$ by structural induction.
Base Case:

- $4 \mid 4$ is trivially true, so $P(4)$ holds.
- $12 = 3 \cdot 4$, so $4 \mid 12$ and $P(12)$ holds.

Inductive Hypothesis: Suppose $P(x)$ and $P(y)$ for some arbitrary $x, y \in \mathfrak{B}$.
Inductive Step:

Goal: Prove $P(x+y)$ and $P(x-y)$

Per the IH, $4 \mid x$ and $4 \mid y$. By the definition of divides, $x = 4k$ and $y = 4j$ for some integers k, j.

Case 1: Goal: Show $P(x + y)$
$x + y = 4k + 4j = 4(k + j)$. Since integers are closed under addition, $k + j$ is an integer, so $4 \mid x + y$ and $P(x + y)$ holds.

Case 2: Goal: Show $P(x - y)$
Similarly, $x - y = 4k - 4j = 4(k - j) = 4(k + (-1 \cdot j))$. Since integers are closed under addition and multiplication, and -1 is an integer, we see that $k - j$ must be an integer. Therefore, by the definition of divides, $4 \mid x - y$ and $P(x - y)$ holds.

So, $P(t)$ holds in both cases.
Conclusion: Therefore, $P(b)$ holds for all numbers $b \in \mathfrak{B}$.

2. 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.

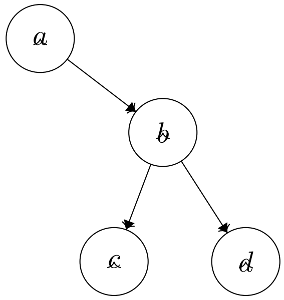
Additional Facts:
You may use the following facts:

- 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, $c^R = c$

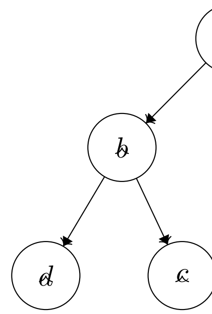
Statement to Prove:
Show that for every **CharTree** T, the reversal of the preorder traversal of T is the same as the postorder traversal of the mirror of T. In notation, you should prove that for every **CharTree**, T: $[\text{preorder}(T)]^R = \text{postorder}(\text{mirror}(T))$.

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
$[\text{preorder}(T_i)]^R = \text{postorder}(\text{mirror}(T_i))$ holds for T_i

Solution:

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

Base case ($T = \texttt{Null}$): $\text{preorder}(T)^R = \varepsilon^R = \varepsilon = \text{postorder}(\texttt{Null}) = \text{postorder}(\text{mirror}(\texttt{Null}))$, 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. $[\text{preorder}(\texttt{CharTree}(L, c, R))]^R = \text{postorder}(\text{mirror}(\texttt{CharTree}(L, c, R)))$.

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.

3. Regular Expressions Warmup

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 zero or more pairs of 45 or 54 in the middle.

4. Context Free Grammars Warmup

Consider the following CFG which generates strings from the language $V := \{0, 1, 2, 3, 4\}^\star$

$$\mathbf{S} \to 0\mathbf{X}4$$
$$\mathbf{X} \to 1\mathbf{X}3 \mid 2$$

List 5 strings generated by the CFG and 5 strings from V not generated by the CFG. Then, summarize this CFG in your own words.

Solution:

Accepted:

- 024
- 01234
- 0112334
- 011123334
- 01111233334

Rejected:

- ϵ
- 2
- 0244
- 011234
- 10234

This CFG is all strings of the form $0\ 1^m\ 2\ 3^m\ 4$, where $m \geq 0$. That is, it's all strings made of one 0, followed by zero or more 1's, followed by a 2, followed by the same number of 3's as 1's, followed by one 4.

5. Constructing RegExs and CFGs

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

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

Solution:

$$(aa)^*$$

$$\mathbf{S} \to aa\mathbf{S}|\varepsilon$$

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

Solution:

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

$$\mathbf{S} \to bS|aSaS|\epsilon$$

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

Solution:

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

$$\mathbf{S} \to \mathbf{CS}|a|b$$
$$\mathbf{C} \to aa\mathbf{C}|ab\mathbf{C}|ba\mathbf{C}|bb\mathbf{C}|\varepsilon$$

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

Solution:

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

$$\mathbf{S} \to \mathbf{E}|\mathbf{O}b\mathbf{O}$$
$$\mathbf{E} \to \mathbf{EE}|a\mathbf{E}a|b|\varepsilon$$
$$\mathbf{O} \to \mathbf{OO}|b\mathbf{O}b|a|\varepsilon$$

6. 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 $\#_0(t) = 0 = \#_1(t)$

Inductive Hypothesis: Suppose P(x) and P(y) hold for arbitrary strings $x, y \in S$.

Inductive Step:
Case 1: Goal: show $P(0x1)$.
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: Goal: show $P(1x0)$
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: Goal: show $P(xy)$
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).

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

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.

7. Bijections

Write a proof to show that both of these functions are a bijection from $\mathbb{R}$ to $\mathbb{R}$.

(a) $f(x) = 2x + 1$

Solution:

In order to prove bijectivity we must show that the function is both **one-to-one** and **onto**.

One-to-one: The function is one-to-one if $\forall x \forall y(f(x) = f(y) \rightarrow x = y)$. Let x, y be arbitrary elements of $\mathbb{R}$ such that $f(x) = f(y)$. By the function definition we have $2x + 1 = 2y + 1$. Subtracting one from both sides gives $2x = 2y$ and dividing by 2 results in $x = y$. Since x, y were arbitrary we have shown that f is one-to-one.

Onto: The function is onto if $\forall y \exists x(f(x) = y)$. Let y be an arbitrary element of $\mathbb{R}$. Consider the expression $x = \frac{y-1}{2}$, where $x \in \mathbb{R}$. Solving for y gives us $2x + 1 = y$. Thus, x is a value which gives $f(x) = y$. Since y was arbitrary we have shown that f is onto.

Since $f(x)$ is both one-to-one and onto, it is a bijection.

(b) $f(x) = x^3$

Solution:

In order to prove bijectivity we must show that the function is both **one-to-one** and **onto**.

One-to-one: The function is one-to-one if $\forall x \forall y(f(x) = f(y) \rightarrow x = y)$. Let x, y be arbitrary elements of $\mathbb{R}$ such that $f(x) = f(y)$. By the function definition we have $x^3 = y^3$. Taking the cube root of both sides gives us $x = y$. Since x, y were arbitrary we have shown that f is one-to-one.

Onto: The function is onto if $\forall y \exists x(f(x) = y)$. Let y be an arbitrary element of the co-domain. Consider the expression $x = \sqrt[3]{y}$, where $x \in \mathbb{R}$. Solving the equation for y gives us $x^3 = y$. Thus, x is a value which gives $f(x) = y$. Since y was arbitrary we have shown that f is onto.

Since $f(x)$ is both one-to-one and onto, it is a bijection.