CSE 311: Foundations of Computing I

Section 8: Structural Induction, REs, and CFGs Solutions

1. Structural Induction I

Consider the following recursive definition of strings Σ^* over the alphabet Σ.

Basis Step: ε is a string

Recursive Step: If w is a string and $a \in \Sigma$ is a character, then wa is a string.

Recall the following recursive definition of the function len:

$$\begin{aligned} \mathsf{len}(\varepsilon) &= 0 \\ \mathsf{len}(wa) &= 1 + \mathsf{len}(w) \end{aligned}$$

Now, consider the following recursive definition:

$$\begin{aligned} \mathsf{double}(\varepsilon) &= \varepsilon \\ \mathsf{double}(wa) &= \mathsf{double}(w)aa. \end{aligned}$$

Prove that, for any string x, we have $\mathsf{len}(\mathsf{double}(x)) = 2\,\mathsf{len}(x)$.

Solution:

Let $\mathsf{P}(x)$ be "$\mathsf{len}(\mathsf{double}(x)) = 2\,\mathsf{len}(x)$". We prove $\mathsf{P}(x)$ for all strings $x \in \Sigma^*$ by structural induction.

Base Case. By definition, $\mathsf{len}(\mathsf{double}(\varepsilon)) = \mathsf{len}(\varepsilon) = 0 = 2 \cdot 0 = 2\,\mathsf{len}(\varepsilon)$, so $\mathsf{P}(\varepsilon)$ holds.

Induction Hypothesis. Suppose $\mathsf{P}(w)$ holds for some arbitrary string w.

Induction Step. We show that $\mathsf{P}(wa)$ holds, for any character $a \in \Sigma$, as follows:

$$\begin{aligned} \mathsf{len}(\mathsf{double}(wa)) &= \mathsf{len}(\mathsf{double}(w)aa) && \text{Def of double} \\ &= 1 + \mathsf{len}(\mathsf{double}(w)a) && \text{Def of len} \\ &= 1 + 1 + \mathsf{len}(\mathsf{double}(w)) && \text{Def of len} \\ &= 2 + 2\,\mathsf{len}(w) && \text{Inductive Hypothesis} \\ &= 2(1 + \mathsf{len}(w)) && \\ &= 2\,\mathsf{len}(wa) && \text{Def of len} \end{aligned}$$

Thus, $\mathsf{P}(x)$ holds for all strings $x \in \Sigma^*$ by structural induction.

2. Structural Induction II

Consider the following definition of a (binary) **Tree**:

Basis Step: $\bullet$ is a **Tree**.

Recursive Step: If L is a **Tree** and R is a **Tree** then $\texttt{Tree}(\bullet, L, R)$ is a **Tree**.

The function leaves returns the number of leaves of a **Tree**. It is defined as follows:

$$\begin{aligned} \mathsf{leaves}(\bullet) &= 1 \\ \mathsf{leaves}(\texttt{Tree}(\bullet, L, R)) &= \mathsf{leaves}(L) + \mathsf{leaves}(R) \end{aligned}$$

Also, recall the definition of size on trees:

$$\begin{aligned} \mathsf{size}(\bullet) &= 1 \\ \mathsf{size}(\texttt{Tree}(\bullet, L, R)) &= 1 + \mathsf{size}(L) + \mathsf{size}(R) \end{aligned}$$

Prove that $\mathsf{leaves}(T) \geq \mathsf{size}(T)/2$ for all $T \in$ **Trees**.

Solution:
In this problem, we define a strengthened predicate. For a tree T, let P be leaves$(T) \geq$ size$(T)/2 + 1/2$. We prove P for all trees T by structural induction.

Base Case. We show that P$(\cdot)$ holds. By definition of leaves(.), leaves$(\bullet) = 1$ and size$(\bullet) = 1$. So, leaves$(\bullet) = 1 \geq 1/2 + 1/2 =$ size$(\bullet)/2 + 1/2$.

Induction Hypothesis: Suppose P(L) and P(R) hold for some arbitrary trees L and R.

Induction Step: We prove that P$(\texttt{Tree}(\bullet, L, R))$ holds as follows:

$$\begin{aligned}
\text{leaves}(\texttt{Tree}(\bullet, L, R)) &= \text{leaves}(L) + \text{leaves}(R) && \text{Def of leaves} \\
&\geq (\text{size}(L)/2 + 1/2) + (\text{size}(R)/2 + 1/2) && \text{Inductive Hypothesis} \\
&= (\text{size}(L) + \text{size}(R) + 1)/2 + 1/2 \\
&= \text{size}(\texttt{Tree}(\bullet, L, R))/2 + 1/2 && \text{Def of size}
\end{aligned}$$

Thus, the P(T) holds for all trees T.

3. Regular Expressions

(a) Write a regular expression that matches base 10 non-negative numbers.
(Note that there should be no leading zeroes.)

Solution:

$$0 \cup ((1 \cup 2 \cup 3 \cup 4 \cup 5 \cup 6 \cup 7 \cup 8 \cup 9)(0 \cup 1 \cup 2 \cup 3 \cup 4 \cup 5 \cup 6 \cup 7 \cup 8 \cup 9)^*)$$

(b) Write a regular expression that matches all non-negative base-3 numbers that are divisible by 3.

Solution:

$$0 \cup ((1 \cup 2)(0 \cup 1 \cup 2)^* 0)$$

(c) Write a regular expression that matches all binary strings that contain the substring "111", but not the substring "000".

Solution:

$$(01 \cup 001 \cup 1^*)^*(0 \cup 00 \cup \varepsilon)111(01 \cup 001 \cup 1^*)^*(0 \cup 00 \cup \varepsilon)$$

(If you don't want the substring 000, the only way you can produce 0s is if there are only one or two 0s in a row, and they are immediately followed by a 1 or the end of the string.)

4. CFGs

Construct CFGs for the following languages:

(a) All binary strings that end in 00.

Solution:

$$\textbf{S} \to 0\textbf{S} \mid 1\textbf{S} \mid 00$$

(b) All binary strings that contain at least three 1's.

Solution:

$$\begin{aligned} \textbf{S} &\to \textbf{TTT} \\ \textbf{T} &\to 0\textbf{T} \mid \textbf{T}0 \mid 1\textbf{T} \mid 1 \end{aligned}$$

(c) Propositional logic statements using only variables from a fixed alphabet $\mathcal{A} = \{\dots, p, q, r, \dots\}$ and only the operators $\neg$, $\wedge$, and $\vee$ as well as parentheses "(..)". (Assume no space characters.)

Solution:

$$\begin{aligned} \textbf{S} &\to \textbf{F} \mid \textbf{S} \vee \textbf{F} \\ \textbf{F} &\to \textbf{P} \mid \textbf{F} \wedge \textbf{F} \\ \textbf{P} &\to \textbf{V} \mid (\textbf{S}) \mid \neg\textbf{P} \\ \textbf{V} &\to \cdots \mid p \mid q \mid r \mid \dots \end{aligned}$$

Note that this gives $\wedge$ higher precedence than $\vee$, as would be expected.

5. Structural Induction III

In this problem, we will prove De Morgan's Law for arbitrary propositions. For example, we will show that

$$\neg(p_1 \wedge p_2 \wedge \cdots \wedge p_n) \equiv \neg p_1 \vee \neg p_2 \vee \cdots \vee \neg p_n.$$

is true for any $n \geq 1$.

Let $\mathcal{A} = \{\dots, p, q, r, \dots\}$ be a fixed set of atomic propositions. We then define the set $\mathbf{Prop}$ as follows:

Basis Elements For any $p \in \mathcal{A}$, $\texttt{Atomic}(p) \in \mathbf{Prop}$.

Recursive Step If $A, B \in \mathbf{Prop}$, then $\text{Neg}(A), \text{Wedge}(A, B), \text{Vee}(A, B) \in \mathbf{Prop}$.

The set $\mathbf{Prop}$ represents parse trees of propositions. We allow the propositions to be combined using the operators, Wedge and Vee (the names of $\wedge$ and $\vee$ in LaTeX). We also allow negation of propositions with Neg.

Next, we define a function $\mathcal{T}$ that takes a parse tree (an element of $\mathbf{Prop}$) as input and returns the proposition that it represents.. Formally we define,

$$\begin{aligned} \mathcal{T}(\texttt{Atomic}(p)) &= p && \text{for any } p \in \mathcal{A} \\ \mathcal{T}(\texttt{Wedge}(A, B)) &= (\mathcal{T}(A)) \wedge (\mathcal{T}(B)) && \text{for any } A, B \in \mathbf{Prop} \\ \mathcal{T}(\texttt{Vee}(A, B)) &= (\mathcal{T}(A)) \vee (\mathcal{T}(B)) && \text{for any } A, B \in \mathbf{Prop} \\ \mathcal{T}(\texttt{Neg}(A)) &= \neg\mathcal{T}(A) && \text{for any } A \in \mathbf{Prop} \end{aligned}$$

The function flip takes a parse tree as input and returns another parse tree as follows:

$$\begin{aligned}
\mathsf{flip}(\mathsf{Atomic}(p)) &= \mathsf{Neg}(\mathsf{Atomic}(p)) && \text{for any } p \in \mathcal{A} \\
\mathsf{flip}(\mathsf{Wedge}(A, B)) &= \mathsf{Vee}(\mathsf{flip}(A), \mathsf{flip}(B)) && \text{for any } A, B \in \mathbf{Prop} \\
\mathsf{flip}(\mathsf{Vee}(A, B)) &= \mathsf{Wedge}(\mathsf{flip}(A), \mathsf{flip}(B)) && \text{for any } A, B \in \mathbf{Prop} \\
\mathsf{flip}(\mathsf{Neg}(A)) &= A && \text{for any } A \in \mathbf{Prop}
\end{aligned}$$

The function flip negates each atomic proposition and swaps $\vee$ with $\wedge$ (and vice versa) throughout the tree.

With those definitions in hand, use structural induction show that, for any $A \in \mathbf{Prop}$,

$$\mathcal{T}(\mathsf{Neg}(A)) \equiv \mathcal{T}(\mathsf{flip}(A)).$$

This proves that we can produce a proposition that is equivalent to negating the expression by, instead, flipping all $\wedge$s to $\vee$s (and vice versa) and negating atomic propositions recursively until we hit $\neg$s.

Solution:

Let $P(A)$ be "$\mathcal{T}(\mathsf{Neg}(A)) \equiv \mathcal{T}(\mathsf{flip}(A))$". We prove $P(A)$ for all $A \in \mathbf{Prop}$ by structural induction.

Base Case Let p be an arbitrary member of $\mathcal{A}$. In this case, $P(\mathsf{Atomic}(p))$ says

$$\mathcal{T}(\mathsf{Neg}(\mathsf{Atomic}(p))) = \mathcal{T}(\mathsf{flip}(\mathsf{Atomic}(p))),$$

which is immediate from the definition of flip (read right-to-left).

Induction Hypothesis Suppose $P(A)$ and $P(B)$ hold for some arbitrary A and B in $\mathbf{Prop}$.

Induction Step We show $P(\mathsf{Wedge}(A, B))$ as follows ($P(\mathsf{Vee}(A, B))$ is similar and left as an exercise):

$$\begin{aligned}
\mathcal{T}(\mathsf{Neg}(\mathsf{Wedge}(A, B)) &= \neg\mathcal{T}(\mathsf{Wedge}(A, B)) && \text{Def of } \mathcal{T} \\
&= \neg(\mathcal{T}(A) \wedge \mathcal{T}(B)) && \text{Def of } \mathcal{T} \\
&\equiv \neg\mathcal{T}(A) \vee \neg\mathcal{T}(B) && \text{De Morgan's Law} \\
&= \mathcal{T}(\mathsf{Neg}(A)) \vee \mathcal{T}(\mathsf{Neg}(B)) && \text{Def of } T \\
&\equiv \mathcal{T}(\mathsf{flip}(A)) \vee T(\mathsf{flip}(B)) && \text{Induction Hypothesis} \\
&= \mathcal{T}(\mathsf{Vee}(\mathsf{flip}(A), \mathsf{flip}(B))) && \text{Def of } T \\
&= \mathcal{T}(\mathsf{flip}(\mathsf{Wedge}(A, B))) && \text{Def of flip}
\end{aligned}$$

We can show $P(\mathsf{Neg}(A)$ as follows:

$$\begin{aligned}
\mathcal{T}(\mathsf{Neg}(\mathsf{Neg}(A)) &= \neg\mathcal{T}(\mathsf{Neg}(A)) && \text{Def of } \mathcal{T} \\
&= \neg\neg\mathcal{T}(A) && \text{Def of } \mathcal{T} \\
&\equiv \mathcal{T}(A) && \text{Double Negation} \\
&= \mathcal{T}(\mathsf{flip}(\mathsf{Neg}(A))) && \text{Def of flip}
\end{aligned}$$

Thus, $P(A)$ holds for all parse trees $A \in \mathbf{Prop}$, by structural induction.