CSE 311 Section 08

Induction, Regular Expressions, CFGs

Administrivia

Announcements & Reminders

- Midterm
    - Please don’t talk about the midterm!! Not everyone has taken it yet ☺
- HW5 Regrade Requests
    - Regrade request window open as usual
    - If something was regraded incorrectly, submit a regrade request
- HW6
    - Due Wednesday 11/22 @ 10pm (Wednesday before Thanksgiving)
    - Late due date Friday 11/24
- HW7
    - Will be released Wednesday 11/22 (Wednesday before Thanksgiving)
    - Due Friday 12/1 @ 10pm (Friday after Thanksgiving)

Recursively Defined Sets

Recursive Definition of Sets

Define a set S as follows:

Basis Step:
Describe the basic starting elements in your set
ex: $0 \in S$

Recursive Step:
Describe how to derive new elements of the set from previous elements
ex: If $x \in S$ then $x + 2 \in S$.

Exclusion Rule: Every element of S is in S from the basis step (alone) or a finite number of recursive steps starting from a basis step.

Problem 3 – Recursively Defined Sets

For each of the following, write a recursive definition of the sets satisfying the following properties. Briefly justify that your solution is correct.

a) Binary strings of even length.

b) Binary strings not containing 10.

c) Binary strings not containing 10 as a substring and having at least as many 1s as 0s.

d) Binary strings containing at most two 0s and at most two 1s.

Work on this problem with the people around you.

Problem 3 – Recursively Defined Sets

For each of the following, write a recursive definition of the sets satisfying the following properties. Briefly justify that your solution is correct.

a) Binary strings of even length.

Problem 3 – Recursively Defined Sets

For each of the following, write a recursive definition of the sets satisfying the following properties. Briefly justify that your solution is correct.

b) Binary strings not containing 10.

Problem 3 – Recursively Defined Sets

For each of the following, write a recursive definition of the sets satisfying the following properties. Briefly justify that your solution is correct.

c) Binary strings not containing 10 as a substring and having at least as many 1s as 0s.

Problem 3 – Recursively Defined Sets

For each of the following, write a recursive definition of the sets satisfying the following properties. Briefly justify that your solution is correct.

d) Binary strings containing at most two 0s and at most two 1s.

Structural Induction

Idea of Structural Induction

Every element is built up recursively…

So to show $P(s)$ for all $s \in S$…

Show $P(b)$ for all base case elements b.

Show for an arbitrary element not in the base case, if $P()$ holds for every named element in the recursive rule, then $P()$ holds for the new element (each recursive rule will be a case of this proof).

Structural Induction Template

Let $P(x)$ be. We show $P(x)$ holds for all $x \in S$ by structural induction.

Base Case: Show $P(x)$
[Do that for every base cases x in S.]
Let y be an arbitrary element of S not covered by the base cases. By the exclusion rule,
y = <recursive rules>

Inductive Hypothesis: Suppose $P(x)$
[Do that for every x listed as in S in the recursive rules.]

Inductive Step: Show $P()$ holds for y.
[You will need a separate case/step for every recursive rule.]

Therefore $P(x)$ holds for all $x \in S$ by the principle of induction.

Problem 4b – Structural Induction on Trees

Definition of Tree:
Basis Step: • is a Tree.
Recursive Step: If L is a Tree and R is a Tree then Tree(•, L, R) is a Tree

Definition of leaves():
leaves(•) = 1
leaves(Tree(•, L, R)) = leaves(L) + leaves(R)

Definition of size():
size(•) = 1
size(Tree(•, L, R)) =1 + size(L) + size(R)

Prove that leaves(T) ≥ size(T)/2 + 1/2 for all Trees T

Work on this problem with the people around you.

Problem 4b – Structural Induction on Trees

Let $P(\text{x})$ be "" for all elements $\text{x} \in S$.
We show $P(\text{x})$ holds for all elements $\text{x} \in S$ by structural induction.
Base Case: (x= <basis>):
Let y be an arbitrary element not covered by the base cases. By the exclusion rule, y=<recursive rule> for <building blocks of y>.
Inductive Hypothesis: Suppose P(<building blocks of y>) holds for <building blocks>
Inductive Step: Goal: Show $P(\text{y})$ holds:

Conclusion: Therefore $P(\text{x})$ holds for all elements $\text{x} \in S$ by the principle of induction.

Problem 4a – Structural Induction on Strings

Definition of string:
Basis Step: "" is a string.
Recursive Step: If X is a string and c is a character then append(c, X) is a string.

Definition of len():
len("") = 0
len(append(c, X)) = 1 + len(X)

Definition of double():
double("") = ""
double(append(c, X)) = append(c, append(c, double(X)))

Prove that for any string X, len(double(X)) = 2len(X).

Work on this problem with the people around you.

Problem 4a – Structural Induction on Strings

Let $P(\text{x})$ be “” for all elements x ∈ S.
We show $P(\text{x})$ holds for all elements x ∈ S by structural induction.
Base Case: (x= <basis>):
Let y be an arbitrary element not covered by the base cases. By the exclusion rule, y=<recursive rule> for <building blocks of y>.
Inductive Hypothesis: Suppose P(<building blocks of y>) holds for <building blocks>
Inductive Step: Goal: Show $P(\text{y})$ holds:

Conclusion: Therefore $P(\text{x})$ holds for all elements x ∈ S by the principle of induction.

Regular Expressions

Regular Expressions

Basis:

- ε is a regular expression. The empty string itself matches the pattern (and nothing else does).
- $\emptyset$ is a regular expression. No strings match this pattern.
- a is a regular expression, for any $a \in \Sigma$ (i.e. any character). The character itself matching this pattern.

Recursive:

- If A, B are regular expressions then $(A \cup B)$ is a regular expression. matched by any string that matches A or that matches B [or both]).
- If A, B are regular expressions then AB is a regular expression. matched by any string x such that $x = yz$, y matches A and z matches B.
- If A is a regular expression, then $A*$ is a regular expression. matched by any string that can be divided into 0 or more strings that match A.

Regular Expressions

A regular expression is a recursively defined set of strings that form a language.

A regular expression will generate all strings in a language, and won’t generate any strings that ARE NOT in the language

Hints:

- Come up with a few examples of strings that ARE and ARE NOT in your language
- Then, after you write your regex, check to make sure that it CAN generate all of your examples that are in the language, and it CAN’T generate those that are not

Problem 1 – Regular Expressions

a) Write a regular expression that matches base 10 numbers (e.g., there should be no leading zeroes).

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

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

d) Write a regular expression that matches all binary strings that do not have any consecutive 0’s or 1’s.

e) Write a regular expression that matches all binary strings of the form $1^{k}y$, where $k \geq 1$ and $y \in \{0,1\}^*$ has at least k 1’s.

Work on this problem with the people around you.

Problem 1 – Regular Expressions

a) Write a regular expression that matches base 10 numbers (e.g., there should be no leading zeroes).

Problem 1 – Regular Expressions

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

Problem 1 – Regular Expressions

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

Problem 1 – Regular Expressions

d) Write a regular expression that matches all binary strings that do not have any consecutive 0’s or 1’s.

Problem 1 – Regular Expressions

e) Write a regular expression that matches all binary strings of the form $1^k y$, where $k \geq 1$ and $y \in \{0,1\}^*$ has at least k 1's.

That’s All, Folks!

Thanks for coming to section this week!
Any questions?