CSE 311: Foundations of Computing

Fall 2013

Lecture 18: Structural induction, regular expressions

Announcements

Midterm back today

Graded Homework 5 back Friday

Homework 6 out later today

Review: Structural Induction

How to prove $\forall\, x \in S, P(x)$ is true:

Base Case: Show that $P(u)$ is true for all specific elements u of S mentioned in the *Basis step*

Inductive Hypothesis: Assume that P is true for some arbitrary values of *each* of the existing named elements mentioned in the *Recursive step*

Inductive Step: Prove that $P(w)$ holds for each of the new elements w constructed in the *Recursive step* using the named elements mentioned in the Inductive Hypothesis

Conclude that $\forall\, x \in S, P(x)$

Function Definitions on Recursively Defined Sets

Length:

len(ε) = 0

len(wa) = len(w) + 1 for w $\in \Sigma^*$, a $\in \Sigma$

Reversal:

$\varepsilon^R = \varepsilon$

$(wa)^R = aw^R$ for w $\in \Sigma^*$, a $\in \Sigma$

Concatenation:

x • ε = x for x $\in \Sigma^*$

x • wa = (x • w)a for x $\in \Sigma^*$, a $\in \Sigma$

len(x • y) = len(x) + len(y) for all x,y ∈ Σ^*

Let P(y) be "len(x•y) = len(x) + len(y) for all x ∈ Σ^*".
We prove P(y) for all y ∈ Σ^* by structural induction.

Base Case: y= ε. For any x ∈ Σ^*, len(x• ε) = len(x) = len(x) + len(ε) since len(ε)=0. Therefore P(ε) is true

Inductive Hypothesis: Assume that P(w) is true for some arbitrary w ∈ Σ^*

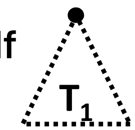
Inductive Step: Goal: Show that P(wa) is true for every a ∈ Σ

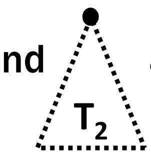
Let a ∈ Σ. Let x ∈ Σ^*. Then len(x•wa) = len((x•w)a) by defn of •
= len(x•w)+1 by defn of len
= len(x)+len(w)+1 by I.H.
= len(x)+len(wa) by defn of len

Therefore len(x•wa)= len(x)+len(wa) for all x ∈ Σ^*, so P(wa) is true.

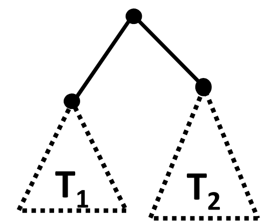
So, by induction len(x•y) = len(x) + len(y) for all x,y ∈ Σ^*

Rooted Binary Trees

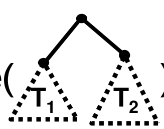
- **Basis:** • is a rooted binary tree
- **Recursive step:** If 

and 

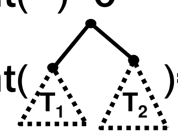
are rooted binary trees

then so is: 

Functions Defined on Rooted Binary Trees

- size(•)=1
- size(

) = 1+size(T_1)+size(T_2)

- height(•)=0
- height(

)=1+max{height(T_1),height(T_2)}

Claim: For every rooted binary tree T, size(T) ≤ $2^{\text{height}(T)+1} - 1$

Languages: sets of strings

- Sets of strings that satisfy special properties are called *languages*. Examples:
  - English sentences
  - Syntactically correct Java/C/C++ programs
  - Σ^* = All strings over alphabet Σ
  - Palindromes over Σ
  - Binary strings that don't have a 0 after a 1
  - Legal variable names. keywords in Java/C/C++
  - Binary strings with an equal # of 0's and 1's

Regular Expressions

Regular expressions over Σ

- **Basis**:
  - $\varnothing$, ε are regular expressions
  - ***a*** is a regular expression for any $a \in \Sigma$
- **Recursive step**:
  - If **A** and **B** are regular expressions then so are:
    - **(A $\cup$ B)**
    - **(AB)**
    - **A***

Each Regular Expression is a "pattern"

ε matches the **empty string**

a matches the one character string *a*

(A $\cup$ B) matches all strings that either **A** matches or **B** matches (or both)

(AB) matches all strings that have a first part that **A** matches followed by a second part that **B** matches

A* matches all strings that have any number of strings (even 0) that **A** matches, one after another

Examples

- 001^*
- 0^*1^*
- $(0 \cup 1)0(0 \cup 1)0$
- $(0^*1^*)^*$
- $(0 \cup 1)^*\ 0110\ (0 \cup 1)^*$
- $(00 \cup 11)^*\ (01010 \cup 10001)(0 \cup 1)^*$

Regular Expressions in Practice

- Used to define the "tokens": e.g., legal variable names, keywords in programming languages and compilers
- Used in **`grep`**, a program that does pattern matching searches in UNIX/LINUX
- Pattern matching using regular expressions is an essential feature of PHP
- We can use regular expressions in programs to process strings!

Regular Expressions in Java

- Pattern p = Pattern.compile("a*b");
- Matcher m = p.matcher("aaaaab");
- boolean b = m.matches();

  `[01]` a 0 or a 1 **^** start of string **$** end of string

  `[0-9]` any single digit **`\.`** period **`\,`** comma **`\-`** minus

  `.` any single character

ab	a followed by b	(**AB**)
(a|b)	a or b	($\mathbf{A} \cup \mathbf{B}$)
a**?**	zero or one of a	($\mathbf{A} \cup \varepsilon$)
a*****	zero or more of a	**A***
a**+**	one or more of a	**AA***

- e.g. **`^[\-+]?[0-9]*(\.|\,)?[0-9]+$`**

  General form of decimal number e.g. 9.12 or -9,8 (Europe)

More Examples

- **All binary strings that have an even # of 1's**

- **All binary strings that *don't* contain 101**