CSE 311 Foundations of Computing I

Lecture 17
Recursive Definitions and Structural Induction
Autumn 2012

Announcements

- Reading assignments
  - Today:
    - 5.3 7th Edition
    - 4.3 6th Edition
    - 3.4 5th Edition (not all there)
- Midterm Friday, Nov 2
  - Closed book, closed notes
  - Practice midterm available on the Web
- Extra office hours Thursday (midterm review)
  - 3:30 pm, Dan Suciu, Gowen 201
  - 4:30 pm, Richard Anderson, Gowen 201

Highlights from last lecture

- Strong Induction

  $P(0)$
  $\underline{\forall k\, ((P(0) \wedge P(1) \wedge P(2) \wedge \ldots \wedge P(k)) \rightarrow P(k+1))}$
  $\therefore\ \forall n\, P(n)$
- Strong Induction proof layout:
  1. By induction we will show that P(n) is true for every n≥0
  2. Base Case: Prove P(0)
  3. Inductive Hypothesis: Assume that for some arbitrary integer k ≥ 0, P(j) is true for every j from 0 to k
  4. Inductive Step: Prove that P(k+1) is true using Inductive Hypothesis that P(j) is true for all values ≤ k
  5. Conclusion: Result follows by induction

Fibonacci Numbers

$f_0 = 0$; $f_1 = 1$; $f_n = f_{n-1} + f_{n-2}$

Bounding the Fibonacci Numbers

Theorem: $2^{n/2-1} \leq f_n$ for $n \geq 1$

<u>Base cases</u>: $2^{1/2-1} \leq 1 = f_1$; $2^{2/2-1} = 1 = f_2$

<u>Inductive step</u>:

Assume that for some $k \geq 2$, P(1), P(2), . . .,P(K)

$f_{k+1} = f_k + f_{k-1} \geq 2^{k/2-1} + 2^{(k-1)/2-1}$
$\geq 2^{(k-1)/2-1} + 2^{(k-1)/2-1}$
$= 2 \cdot 2^{(k-1)/2-1} = 2^{(k+1)/2-1}$

Recursive Definitions of Sets

- Recursive definition
  - Basis step: $0 \in S$
  - Recursive step: if $x \in S$, then $x + 2 \in S$
  - Exclusion rule: Every element in S follows from basis steps and a finite number of recursive steps

Recursive definitions of sets

Basis: 6 ∈ S; 15 ∈ S;
Recursive: if x, y ∈ S, then x + y ∈ S;

Basis: [1, 1, 0] ∈ S, [0, 1, 1] ∈ S;
Recursive:
if [x, y, z] ∈ S, α in R, then [α x, α y, α z] ∈ S
if $[x_1, y_1, z_1]$, $[x_2, y_2, z_2]$ ∈ S
then $[x_1 + x_2, y_1 + y_2, z_1 + z_2]$ ∈ S

Powers of 3

Recursive Definitions of Sets: General Form

- Recursive definition
  - *Basis step:* Some specific elements are in S
  - *Recursive step:* Given some existing named elements in S some new objects constructed from these named elements are also in S.
  - Exclusion rule: Every element in S follows from basis steps and a finite number of recursive steps

Strings

- An *alphabet* Σ is any finite set of characters.
- The set Σ* of *strings* over the alphabet Σ is defined by
  - Basis: λ ∈ S (λ is the empty string)
  - Recursive: if w ∈ Σ*, x ∈ Σ, then wx ∈ Σ*

Palindromes

- Palindromes are strings that are the same backwards and forwards
- Basis: λ is a palindrome and any a ∈ Σ is a palindrome
- Recursive step: If *p* is a palindrome then a*p*a is a palindrome for every a ∈ Σ

All binary strings with no 1's before 0's

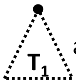
Function definitions on recursively defined sets

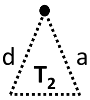
len (λ) = 0;
len (wa) = 1 + len(w); for w ∈ Σ*, a ∈ Σ

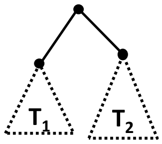
Reversal:
$\lambda^R = \lambda$
$(wa)^R = aw^R$ for w ∈ Σ*, a ∈ Σ

Concatenation:
x • λ = x for x ∈ Σ*
x • wa = (x • w)a for x, w ∈ Σ*, a ∈ Σ

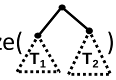
Rooted Binary trees

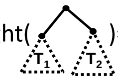
- Basis: • is a rooted binary tree
- Recursive Step: If 

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)}