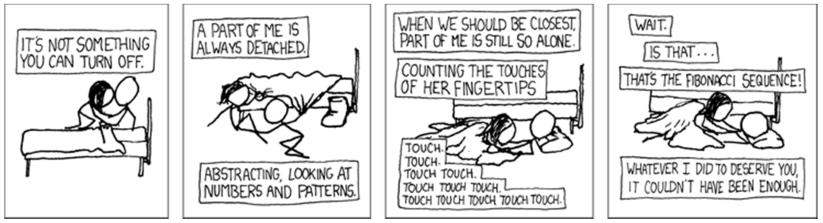
CSE 311: Foundations of Computing

Fall 2013

Lecture 17: Recursive definitions and structural induction



announcements

- Reading assignment
  - Induction
    - 5.3, 7th edition
    - 4.3, 6th edition
- Homework 5 solutions out today
- Midterm
  - Monday, November 4th, IN CLASS
  - Topics: Everything up to ordinary induction and recursive definition of functions
  - Sample questions from old midterms now posted.
  - Solutions posted later today.
- Review sessions: Today 3:30, Sunday 4:00 in EEB 125

review: recursive definition 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

review: recursive definitions of sets

Basis: $6 \in S$; $15 \in S$;
Recursive: if $x, y \in S$, then $x + y \in S$;

Basis: $[1,1,0] \in S, [0,1,1] \in S$;
Recursive:

if $[x,y,z] \in S$, $\alpha \in \mathbb{R}$, then $[\alpha x, \alpha y, \alpha z] \in S$
if $[x_1,y_1,z_1], [x_2,y_2,z_2] \in S$
then $[x_1 + x_2,\ y_1 + y_2,\ z_1 + z_2] \in S$

Powers of 3: $1 \in S$ and $x \in S \Rightarrow 3x \in S$

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

review: strings

- An *alphabet* Σ is any finite set of characters
- The set Σ^* of *strings* over the alphabet Σ is defined by
  - **Basis:** $\lambda \in \Sigma^*$ (λ is the empty string)
  - **Recursive:** if $w \in \Sigma^*$, $a \in \Sigma$, then $wa \in \Sigma^*$

review: palindromes

Palindromes are strings that are the same backwards and forwards

Basis:

λ is a palindrome and any $a \in \Sigma$ is a palindrome

Recursive step:

If $p \in \Sigma^*$ is a palindrome then apa is a palindrome for every $a \in \Sigma$

review: 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 $w \in \Sigma^*, a \in \Sigma$

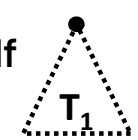
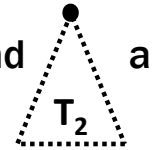
Reversal:

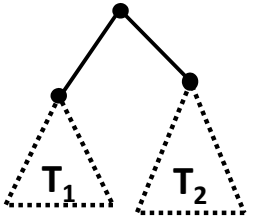
$\lambda^R = \lambda$
$(wa)^R = aw^R$ for $w \in \Sigma^*, a \in \Sigma$

Concatenation:

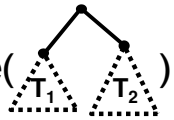
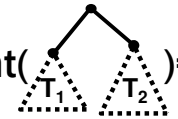
$x \bullet \lambda = x$ for $x \in \Sigma^*$
$x \bullet wa = (x \bullet w)a$ for $x, w \in \Sigma^*, a \in \Sigma$

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)}

structural induction

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

– **Base Case:** Show that P is true for all specific elements 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 holds for each of the new elements constructed in the ***Recursive step*** using the named elements mentioned in the Inductive Hypothesis
– **Conclude** that $\forall x \in S, P(x)$

structural induction vs. ordinary induction

Ordinary induction is a special case of structural induction:

Recursive definition of $\mathbb{N}$

Basis: $0 \in \mathbb{N}$

Recursive Step: If k $\in \mathbb{N}$ then k+1 $\in \mathbb{N}$

Structural induction follows from ordinary induction:

Let $Q(n)$ be true iff for all $x \in S$ that take n recursive steps to be constructed, $P(x)$ is true.

using structural induction

- Let S be given by
  - **Basis:** $6 \in S$; $15 \in S$;
  - **Recursive:** if $x, y \in S$, then $x + y \in S$.
- **Claim:** Every element of S is divisible by 3.

Claim: Every element of S is divisible by 3.

structural induction for strings

- Let S be a set of strings over $\{a, b\}$ defined as follows
  - **Basis:** $a \in S$
  - **Recursive:**

    If $w \in S$ then $aw \in S$ and $baw \in S$

    If $u \in S$ and $v \in S$ then $uv \in S$
- **Claim:** If $w \in S$ then w has more a's than b's

Claim: If $w \in S$ then w has more a's than b's

Basis: $a \in S$

Recursive: If $w \in S$ then $aw \in S$ and $baw \in S$

If $u \in S$ and $v \in S$ then $uv \in S$

function definitions on recursively defined sets

len (λ) = 0;

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

Reversal:

$\lambda^R = \lambda$

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

Concatenation:

$x \bullet \lambda = x$ for $x \in \Sigma^*$

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

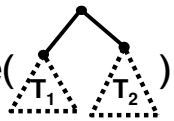
$\text{len}(x \bullet y) = \text{len}(x) + \text{len}(y)$ for all strings x and y

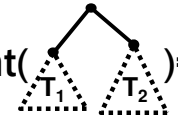
Let $P(y)$ be "$\text{len}(x \bullet y) = \text{len}(x) + \text{len}(y)$ for all strings x"

$\text{len}(x \bullet y) = \text{len}(x) + \text{len}(y)$ for all strings x and y

Let $P(y)$ be "$\text{len}(x \bullet y) = \text{len}(x) + \text{len}(y)$ for all strings x"

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)}

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

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