CSE 311: Foundations of Computing

Lecture 18: Structural Induction, Regular expressions

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 the basis step and a finite number of recursive steps

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 $\forall\ x \in S, P(x)$**

Strings

- An *alphabet* Σ is any finite set of characters

- The set Σ^* of *strings* over the alphabet Σ is defined by
  - **Basis:** $\varepsilon \in \Sigma^*$ (ε is the empty string w/ no chars)
  - **Recursive:** if $w \in \Sigma^*$, $a \in \Sigma$, then $wa \in \Sigma^*$

Functions on Recursively Defined Sets (on Σ^*)

Length:

$\text{len}(\varepsilon) = 0$

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

Reversal:

$\varepsilon^R = \varepsilon$

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

Concatenation:

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

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

Number of c's in a string:

$\#_c(\varepsilon) = 0$

$\#_c(wc) = \#_c(w) + 1$ for $w \in \Sigma^*$

$\#_c(wa) = \#_c(w)$ for $w \in \Sigma^*$, $a \in \Sigma$, $a \neq c$

Claim: 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.**

Claim: len(x•y) = len(x) + len(y) **for all** x,y $\in \Sigma^*$

Let P(y) **be** "len(x•y) = len(x) + len(y) for all x $\in \Sigma^*$" .
We prove P(y) **for all** y $\in \Sigma^*$ **by structural induction.**

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

Claim: len(x•y) = len(x) + len(y) **for all** x,y $\in \Sigma^*$

Let P(y) **be** "len(x•y) = len(x) + len(y) for all x $\in \Sigma^*$" .
We prove P(y) **for all** y $\in \Sigma^*$ **by structural induction.**

Base Case: y= ε. **For any** x $\in \Sigma^*$, 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 $\in \Sigma^*$

Inductive Step: **Goal: Show that** P(wa) **is true for every** a $\in \Sigma$

Claim: $\text{len}(x\bullet y) = \text{len}(x) + \text{len}(y)$ **for all** $x,y \in \Sigma^*$

Let $P(y)$ **be** "$\text{len}(x\bullet y) = \text{len}(x) + \text{len}(y)$ for all $x \in \Sigma^*$" .
We prove $P(y)$ **for all** $y \in \Sigma^*$ **by structural induction.**

Base Case: $y = \varepsilon$. **For any** $x \in \Sigma^*$, $\text{len}(x\bullet \varepsilon) = \text{len}(x) = \text{len}(x) + \text{len}(\varepsilon)$
since $\text{len}(\varepsilon)=0$. **Therefore** $P(\varepsilon)$ **is true**

Inductive Hypothesis: Assume that $P(w)$ **is true for some arbitrary** $w \in \Sigma^*$

Inductive Step: **Goal: Show that** $P(wa)$ **is true for every** $a \in \Sigma$

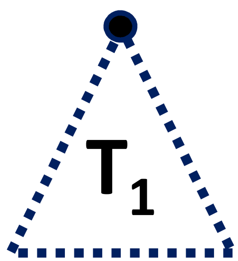
Let $a \in \Sigma$**. Let** $x \in \Sigma^*$. **Then** $\text{len}(x\bullet wa) = \text{len}((x\bullet w)a)$ **by defn of** $\bullet$
$= \text{len}(x\bullet w)+1$ **by defn of len**
$= \text{len}(x)+\text{len}(w)+1$ **by I.H.**
$= \text{len}(x)+\text{len}(wa)$ **by defn of len**

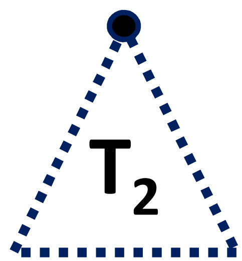
Therefore $\text{len}(x\bullet wa)= \text{len}(x)+\text{len}(wa)$ **for all** $x \in \Sigma^*$**, so** $P(wa)$ **is true.**

So, by induction $\text{len}(x\bullet y) = \text{len}(x) + \text{len}(y)$ **for all** $x,y \in \Sigma^*$

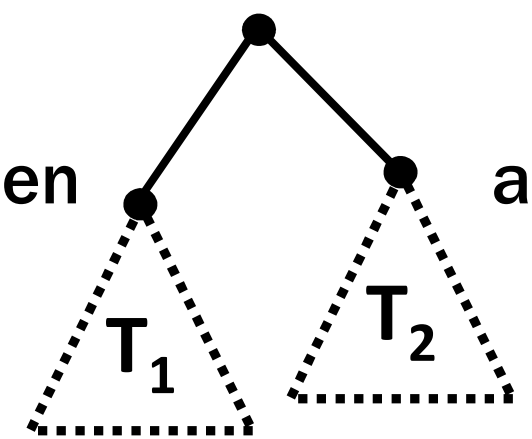
Rooted Binary Trees

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

If 

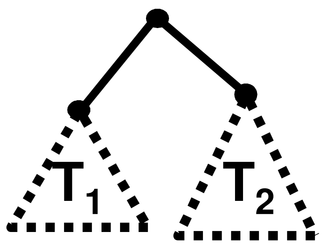
and 

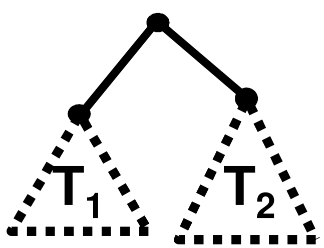
are rooted binary trees,

then 

also is a rooted binary tree.

Defining Functions on Rooted Binary Trees

- size(•) = 1
- size (

) = 1 + size($\mathbf{T_1}$) + size($\mathbf{T_2}$)
- height(•) = 0
- height (

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

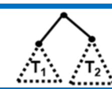
Claim: **For every rooted binary tree T,** size(**T**) $\leq 2^{\text{height}(\mathbf{T})+1} - 1$

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

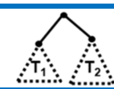
1. **Let** P(**T**) **be** "size(T) $\leq 2^{\text{height}(T)+1}-1$". **We prove** P(**T**) **for all rooted binary trees T by structural induction.**
2. **Base Case:** size(•)=1, height(•)=0 **and** $1=2^1-1=2^{0+1}-1$ **so** P(•) **is true.**

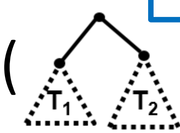
Claim: For every rooted binary tree T, $\text{size}(T) \le 2^{\text{height}(T) + 1} - 1$

1. Let P(**T**) be "$\text{size}(T) \le 2^{\text{height}(T)+1} - 1$". We prove P(**T**) for all rooted binary trees T by structural induction.
2. **Base Case:** $\text{size}(\bullet)=1$, $\text{height}(\bullet)=0$ and $1=2^1-1=2^{0+1}-1$ so P(•) is true.
3. **Inductive Hypothesis:** Suppose that $P(T_1)$ and $P(T_2)$ are true for some rooted binary trees T_1 and T_2.
4. **Inductive Step:**

Goal: Prove P().

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

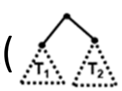
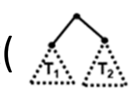
1. Let P(**T**) be "size(T) ≤ $2^{\text{height}(T)+1}-1$". We prove P(**T**) for all rooted binary trees T by structural induction.
2. **Base Case:** size(•)=1, height(•)=0 and $1=2^1-1=2^{0+1}-1$ so P(•) is true.
3. **Inductive Hypothesis:** Suppose that P($\mathbf{T_1}$) and P($\mathbf{T_2}$) are true for some rooted binary trees $\mathbf{T_1}$ and $\mathbf{T_2}$.
4. **Inductive Step:** Goal: Prove P().

By defn, size() $=1+\text{size}(\mathbf{T_1})+\text{size}(\mathbf{T_2})$

$\le 1+2^{\text{height}(\mathbf{T_1})+1}-1+2^{\text{height}(\mathbf{T_2})+1}-1$ by IH for $\mathbf{T_1}$ and $\mathbf{T_2}$

$= 2^{\text{height}(\mathbf{T_1})+1}+2^{\text{height}(\mathbf{T_2})+1}-1$

$\le 2(2^{\max(\text{height}(\mathbf{T_1}),\text{height}(\mathbf{T_2}))+1})-1$

$= 2(2^{\text{height}(}$  $^{)})-1 = 2^{\text{height}(}$  $^{)+1}-1$

which is what we wanted to show.

5. So, the P(T) is true for all rooted bin. trees by structural induction.

Languages: Sets of Strings

- Sets of strings that satisfy special properties are called *languages*. Examples:
  - English sentences
  - Syntactically correct Java/C/C++ programs
  - $\Sigma^* =$ 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 ∪ 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*

Examples

001*

{00, 001, 0011, 00111, ...}

0*1*

Any number of 0's followed by any number of 1's

Examples

$(0 \cup 1)\ 0\ (0 \cup 1)\ 0$

$(0^*1^*)^*$

Examples

$(0 \cup 1)\, 0\, (0 \cup 1)\, 0$

{0000, 0010, 1000, 1010}

$(0^*1^*)^*$

All binary strings

Examples

$(0 \cup 1)^* \, 0110 \, (0 \cup 1)^*$

$(00 \cup 11)^* \, (01010 \cup 10001) \, (0 \cup 1)^*$

Examples

$(0 \cup 1)^* \, 0110 \, (0 \cup 1)^*$

Binary strings that contain “0110”

$(00 \cup 11)^* \, (01010 \cup 10001) \, (0 \cup 1)^*$

Binary strings that begin with pairs of characters followed by “01010” or “10001”

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	$\mathbf{A}^*$
a**+**	one or more of a	$\mathbf{AA}^*$

- e.g. `^[\−+]?[0−9]*(\.|\,)?[0−9]+$`

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