Spring 2015

Lecture 19: Structural induction and regular expressions

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

  e.g. $\Sigma = \{0,1\}$ or $\Sigma = \{A, B, C, \dots X, Y, Z\}$ or

  $\Sigma =$

1		28	∟	95	_	153	Ö	186	║	219	█
2	☻	29	↔	96	`	154	Ü	187	╗	220	▄
3	♥	30	▲	97-122	a-z	155	¢	188	╝	221	▌
4	♦	31	▼	123	{	156	£	189	╜	222	▐
5	♣	32	(space)	124	|	157	¥	190	╛	223	▀
6	♠	33	!	125	}	158	₧	191	┐	224	α
7	•	34	"	126	~	159	ƒ	192	└	225	ß
8	◘	35	#	127	⌂	160	á	193	┴	226	Γ
9	○	36	$	128	Ç	161	í	194	┬	227	π
10	◙	37	%	129	ü	162	ó	195	├	228	Σ
11	♂	38	&	130	é	163	ú	196	─	229	σ

- The set Σ^* of *strings* over the alphabet Σ is defined by
  - **Basis:** $\varepsilon \in \Sigma^*$ (ε is the empty string)
  - **Recursive:** if $w \in \Sigma^*$, $a \in \Sigma$, then $wa \in \Sigma^*$

$\overline{\Sigma_1^*} = \Sigma_1^* \setminus \varepsilon$

Basis: $\forall a \in \Sigma_1$, $a \in \overline{\Sigma_1^*}$

Recur: if $w \in \overline{\Sigma_1^*}$ and $a \in \Sigma_1 \rightarrow wa \in \overline{\Sigma_1^*}$

function definitions on recursively defined sets

Length:

len (ε) = 0;

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

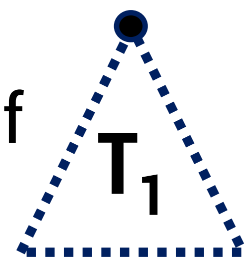
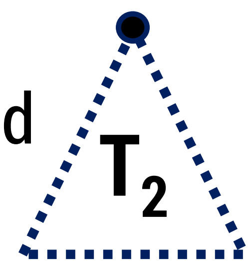
$x \cdot (a_1 \cdots a_k)$

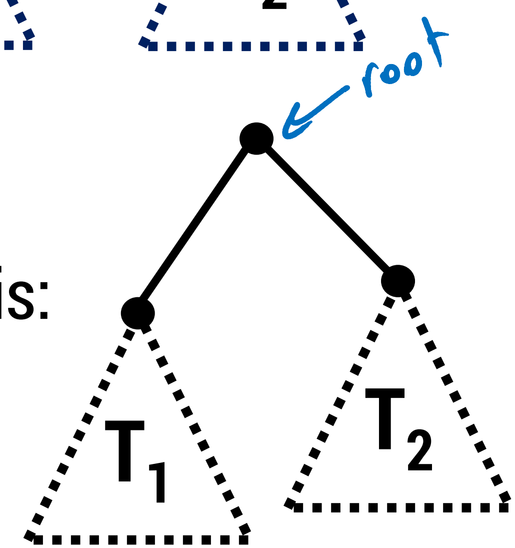
$= (x \cdot a_1 \cdots a_{k-1})\, a_k$

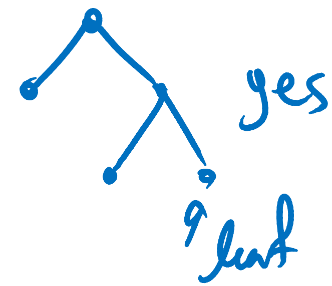
$= \cdots = x\, a_1 \cdots a_k$

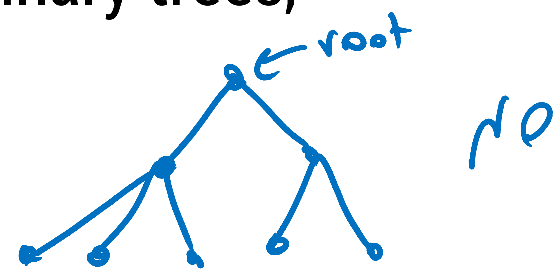
rooted binary trees

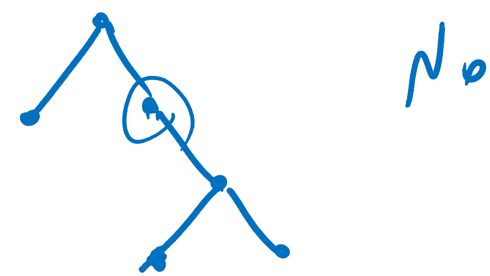
- Basis: • is a rooted binary tree
- Recursive step:

If  and  are rooted binary trees,

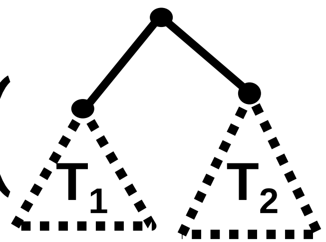
then so is: 

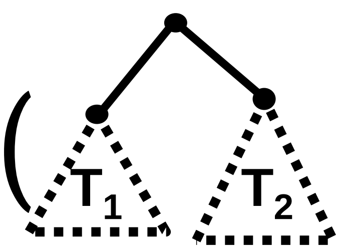






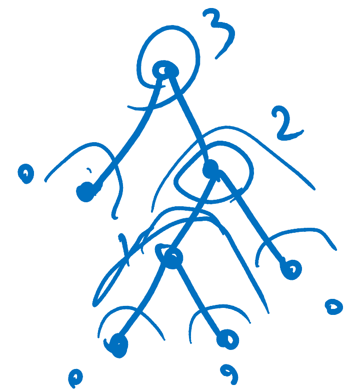
defining a function on rooted binary trees

- size(•) = 1
- size (

) = 1 + size(T_1) + size(T_2)
- height(•) = 0
- height (

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

size = 7
height = 3



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

Let S be a set of strings over $\Sigma = \{a, b\}$ defined by

Basis: $a \in S$

Recursive:

If $w \in S$ then $wa \in S$ and $wba \in S$

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

$ba \notin S$.

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

$P(w)$ = "w has more a's than b's."

Base Case: Show $P(a)$ holds.

$\#_a(a) = 1 > \#_b(a)$ ✓

IH: For some $w, u, v \in S$, $P(u), P(w), P(v)$ hold.

IS: $\#_a(wa) = \#_a(w) + 1 \underset{\text{IH}}{>} \#_b(w) + 1 > \#_b(wa) \Rightarrow P(wa)$ holds.

$\#_a(wba) = \#_a(w) + 1 \underset{\text{IH}}{>} \#_b(w) + 1 = \#_b(wba) \Rightarrow P(wba)$ holds

proof continued?

$$\#_a(uv) = \#_a(u) + \#_a(v) \underset{\uparrow\,\mathrm{IH}}{>} \#_b(u) + \#_b(v)$$

$$= \#_b(uv)$$

$\Rightarrow P(uv)$ holds.

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

Let $P(y)$ be "$\text{len}(x \cdot y) = \text{len}(x) + \text{len}(y)$ for all $x \in \Sigma^*$

Base Case: $P(\varepsilon)$ holds.

$$\text{len}(x \cdot \varepsilon) \underset{\text{def.}}{=} \text{len}(x) \underset{\text{def len}}{=} \text{len}(x) + \text{len}(\varepsilon) \quad ✓$$

IH: for some $w \in \Sigma^*$, $P(w)$ holds.

IS: Goal. $P(wa)$ holds for any $a \in \Sigma$.

Fix $x \in \Sigma^*$

$$\text{len}(x \cdot wa) \underset{\text{def of .}}{=} \text{len}((x \cdot w)a) \underset{\text{def len}}{=} \text{len}(x \cdot w) + 1$$

$$\underset{\text{IH}}{=} \text{len}(x) + \text{len}(w) + 1$$

$$\underset{\text{def. len}}{=} \text{len}(x) + \text{len}(wa)$$

$\Rightarrow P(wa)$ holds

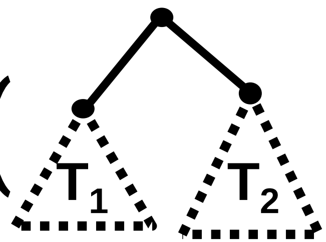
Conclu: $P(y)$ holds $\forall y \in \Sigma^*$

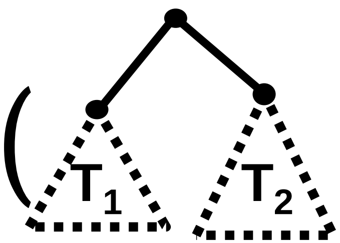
Length:

len $(\varepsilon) = 0$;

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

defining a function 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)}

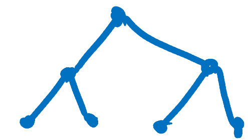
size vs. height

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

$P(T) =$ "$\text{size}(T) \le 2^{\text{height}(T)+1} - 1$".

size $= 7$

height $= 2$



$7 \le 2^{2+1} - 1$

Base Case: $P(\cdot)$ holds

$\text{size}(\cdot) = 1 \le 2^{0+1} - 1 = 2^{\text{height}(\cdot)+1} - 1$

IH: $P(T_1)$, $P(T_2)$ hold for some T_1, T_2

IS: Goal: $P($  $)$ holds

$\text{size}(T_3) = 1 + \text{size}(T_1) + \text{size}(T_2) \le 1 + 2^{\text{height}(T_1)+1} - 1 + 2^{\text{height}(T_2)+1} - 1$

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

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

$= 2 \cdot 2^{\text{height}(T_3)} - 1 = 2^{\text{height}(T_3)+1} - 1 \Rightarrow P(T_3)$

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