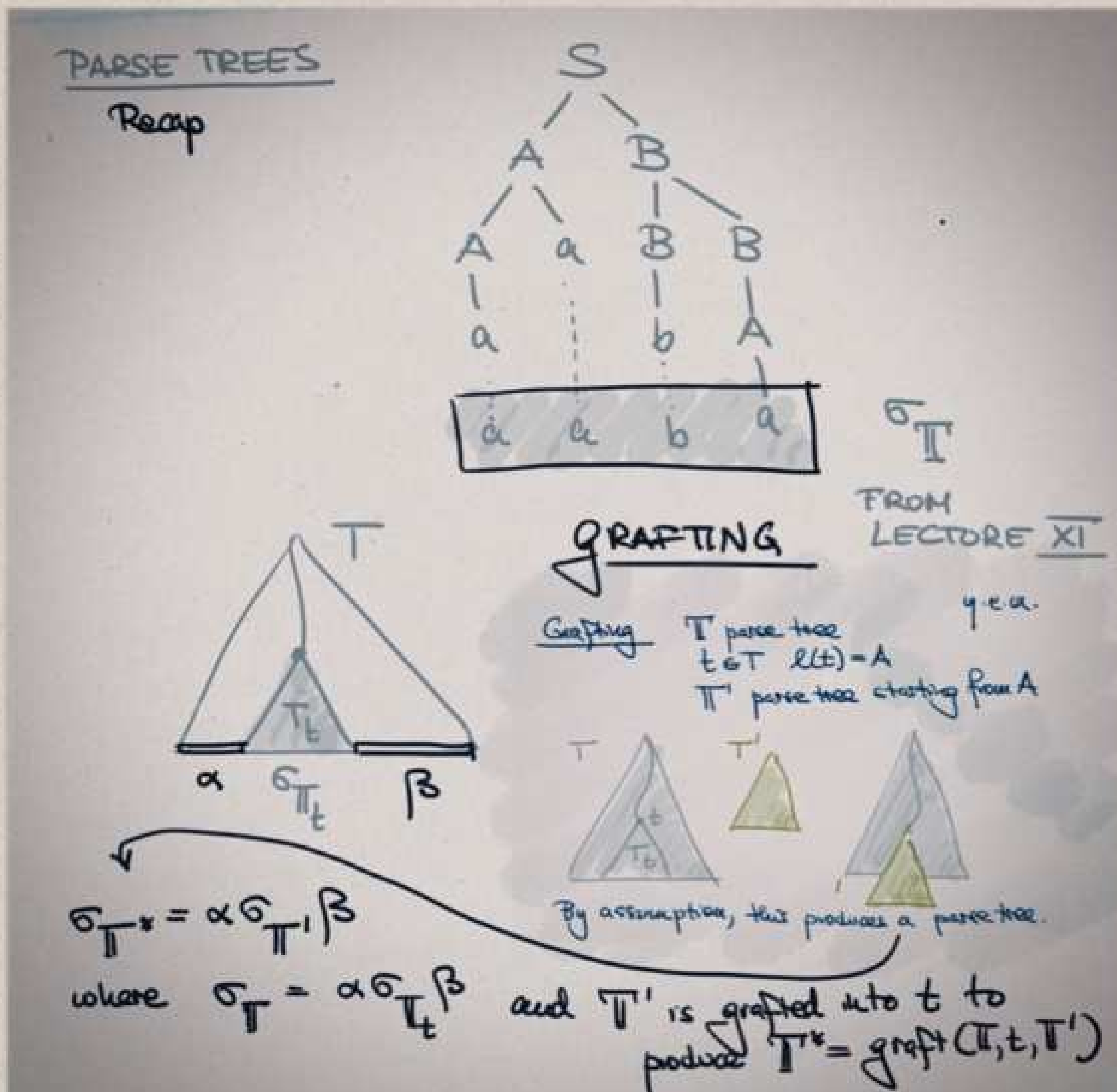
XIII

AUTOMATA & FORMAL LANGUAGES
THIRTEENTH LECTURE

3 November 2022

Re-recap

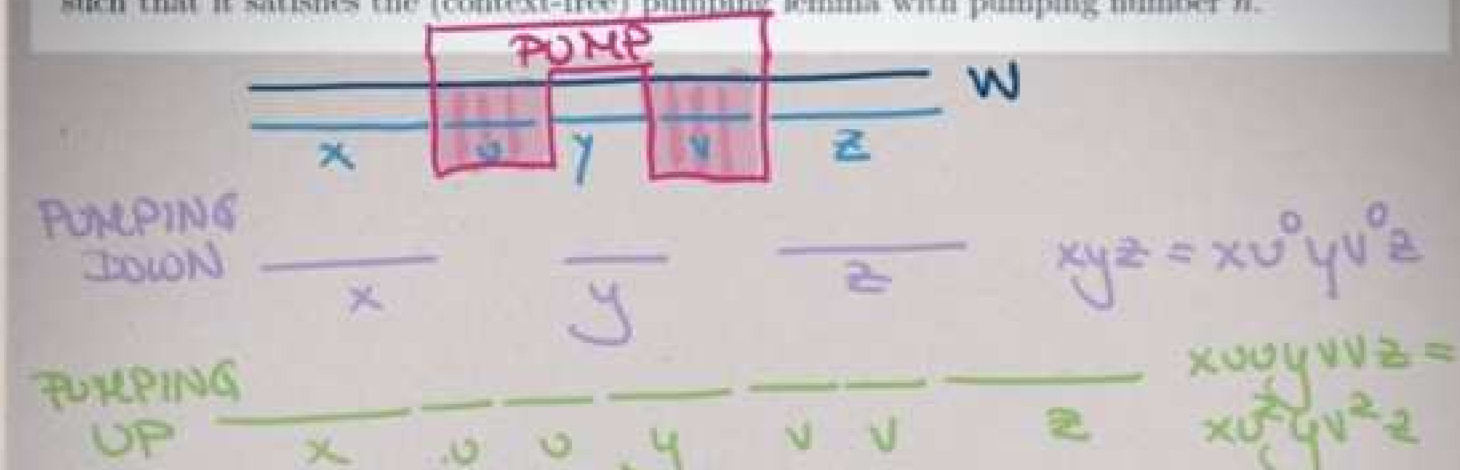


[Recap from Lecture XII]

Recap

Definition 3.9. Let $L \subseteq \mathbb{W}$ be a language. We say that L *satisfies the (context-free) pumping lemma with pumping number* n if for every word $w \in L$ such that $|w| \geq n$ there are words u, v, x, y, z such that $w = xuyvz$, $|uv| > 0$, $|uyv| \leq n$ and for all $k \in \mathbb{N}$, we have that $xu^kyv^kz \in L$. We say that L *satisfies the (context-free) pumping lemma* if there is some n such that it satisfies the (context-free) pumping lemma with pumping number n.

Definition 3.9. Let $L \subseteq \mathbb{W}$ be a language. We say that L *satisfies the (context-free) pumping lemma with pumping number* n if for every word $w \in L$ such that $|w| \geq n$ there are words u, v, x, y, z such that $w = xuyvz$, $|uv| > 0$, $|uyv| \leq n$ and for all $k \in \mathbb{N}$, we have that $xu^kyv^kz \in L$. We say that L *satisfies the (context-free) pumping lemma* if there is some n such that it satisfies the (context-free) pumping lemma with pumping number n.



REMINDER Regular Pumping lemma

Definition 2.10. Let $L \subseteq \mathbb{W}$ be a language. We say that L *satisfies the (regular) pumping lemma with pumping number* n if for every word $w \in L$ such that $|w| \geq n$ there are words x, y, z such that $w = xyz$, $|y| > 0$, $|xy| \leq n$ and for all $k \in \mathbb{N}$, we have that $xy^kz \in L$. We say that L *satisfies the (regular) pumping lemma* if there is some n such that it satisfies the (regular) pumping lemma with pumping number n.

If a language L satisfies the pumping lemma and we have written $w = xyz$ as in the definition, then $xz = xy^0z$, xy^2z, xy^3z, etc. are all in L. We call the transition from $w = xyz$ to xz *pumping down* and the transition to xy^kz (for $k > 1$) *pumping up*.

Theorem 2.11 (The regular pumping lemma). For every regular language L there is a number n such that L satisfies the regular pumping lemma with pumping number n.

CONTEXT-FREE

for all $w \in L$
$|w| \geq n$,
there are x, y, z, u, v
s.t.
$w = xuyvz$
$|uyv| \leq n$, $|uv| > 0$
& f.a. k $xu^kyv^kz \in L$.

for all $w \in L$ s.t. $|w| \geq n$, there are x, y, z s.t.
$w = xyz$, $|xy| \leq n$, $|y| > 0$
& for all k
$xy^kz \in L$

Theorem 3.11 For every context-free language L there is an $n \in \mathbb{N}$ s.t. L satisfies the context-free pumping lemma with pumping number n.

Proof. By Theorem 3.4, we find G in CNF s.t. $L = \mathcal{L}(G)$. $G = (\Sigma, V, P, S)$

Let $m := |V|$; $n := 2^m$.

CLAIM n is the pumping # of G.

Let us analyse what information CNF gives:

Claim If $\mathbb{T}$ is a G-parse tree and the height of $\mathbb{T}$ is $h+1$ and $\sigma_{\mathbb{T}} \in \mathbb{W}$, then

$$|\sigma_{\mathbb{T}}| \leq 2^h.$$

[Observe that the full binary tree of height $h+1$ has 2^{h+1} many leaves.

In order to have a letter in $\sigma_{\mathbb{T}}$ I need to use one unary rule per letter.

Every unary reduces # of leaves at least by one.

Thus the tree must have at most

$2^{h+1} - |w|$ leaves.

Therefore $|w| \leq 2^h$.]

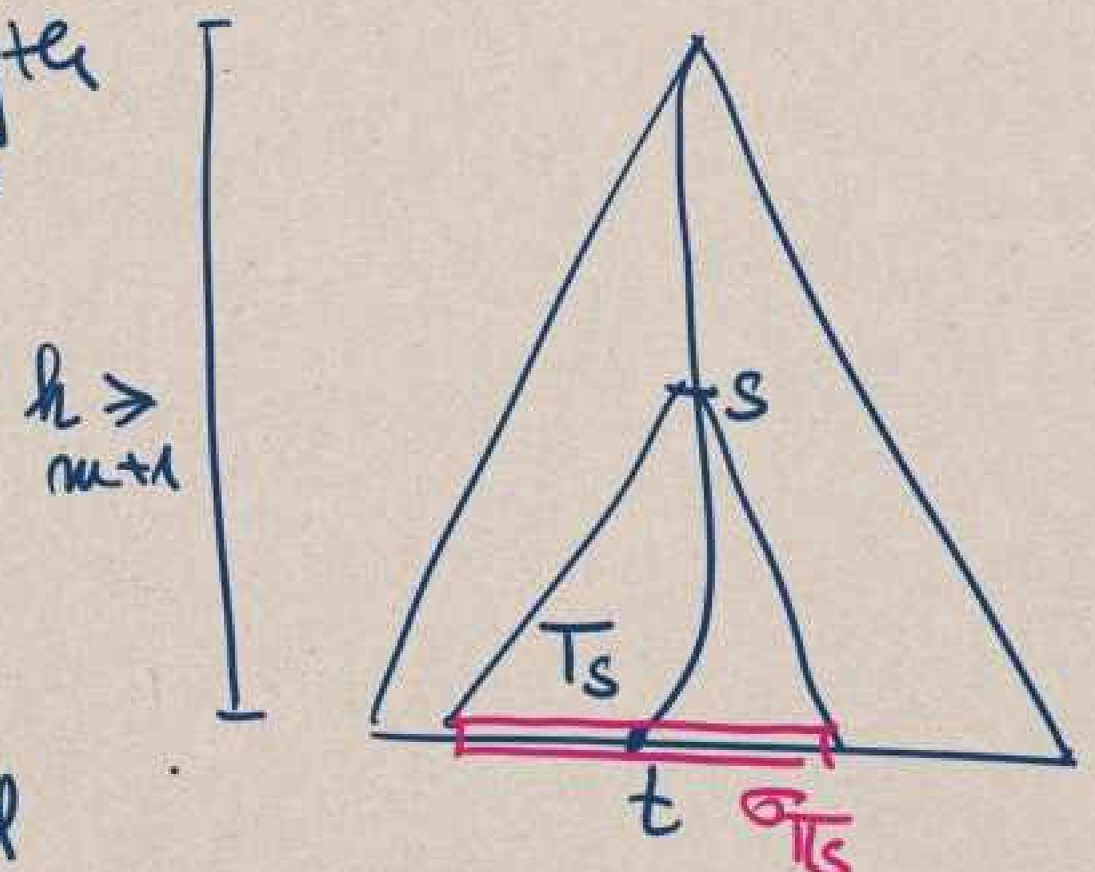
Now prove that 2^m is the pumping #:

Let $w \in \mathcal{L}(G)$ with $|w| \geq n = 2^m$.

If $\mathbb{T}$ is a G-parse tree of w, i.e., $\sigma_{\mathbb{T}} = w$, we know by previous claim height of $\mathbb{T} \geq m+1$.

Fix t s.t. the length of the branch to t is $h \geq m+1$.

The path from ε to t has $h+1$ many labels: h variables and one letter.



Find s on the branch s.t. height of T_s is exactly $m+1$.

Note that $|\sigma_{\mathbb{T}_s}| \leq 2^m = n$. (*)

The path from s to t has $(m+1)+1$ many labels:

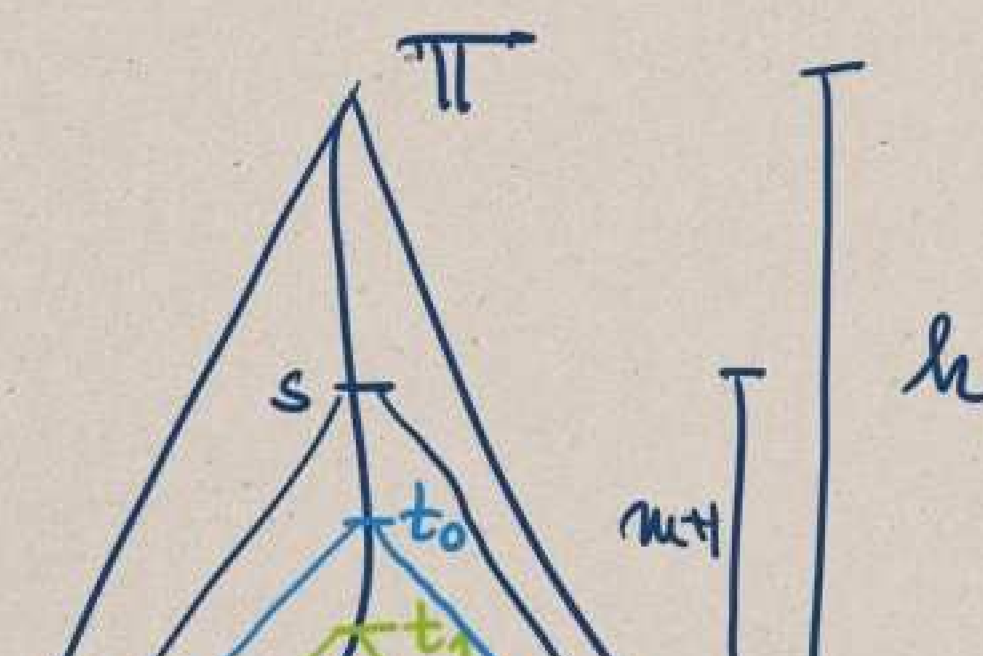
$m+1$ variables &

1 letter.

By pigeonhole, we find two nodes on the branch from s to t, say,

$$t_0 \subsetneqq t_1 \quad \text{and } A \in V$$

s.t. $\quad \ell(t_0) = A = \ell(t_1).$



$x := ax'$

$u := u$

$y := y$

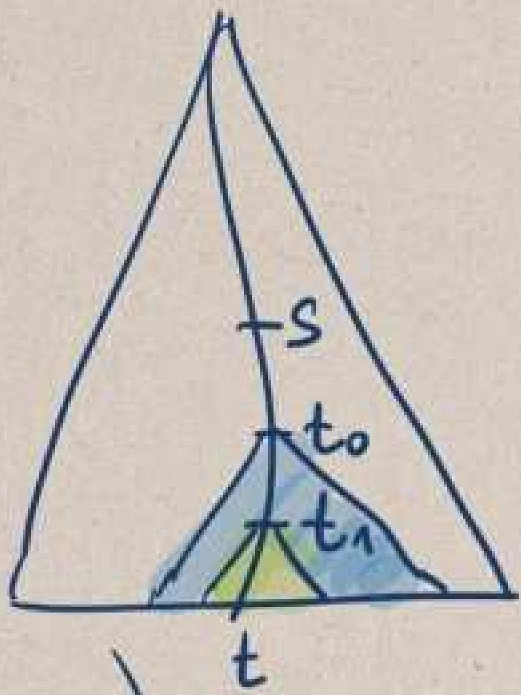
$v := v$

$z := z'b$

Clearly since $t_0 \neq t_1 \Longrightarrow |uv| > 0.$

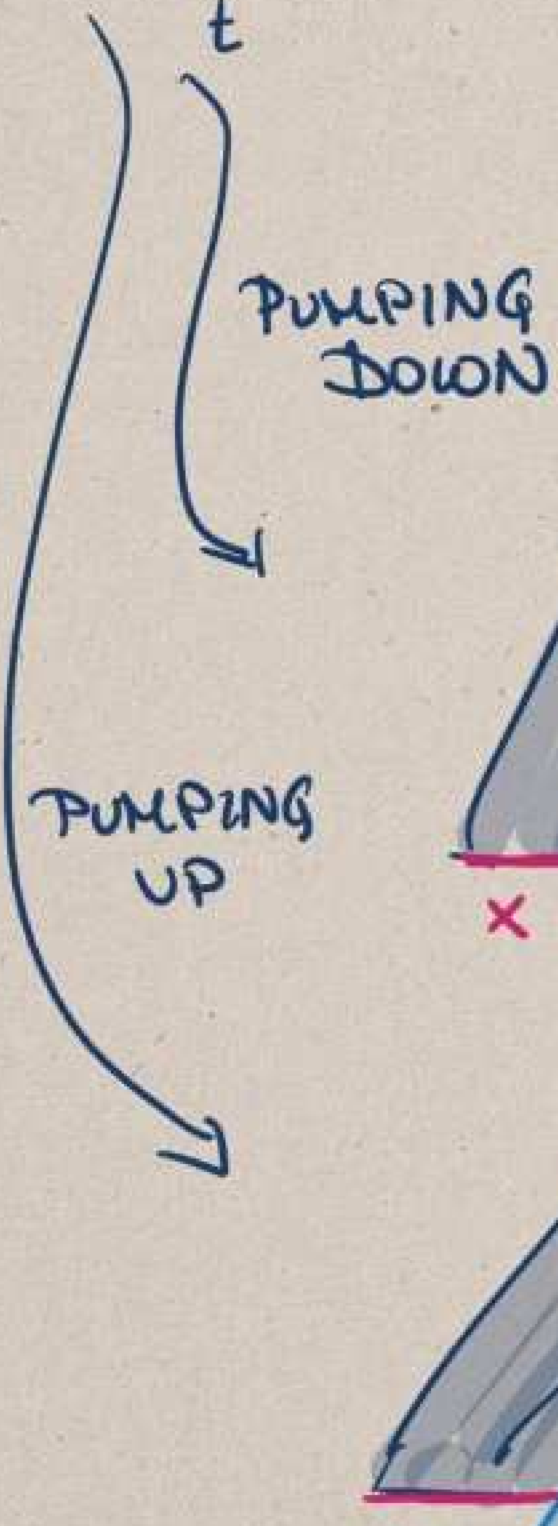
(*) last page

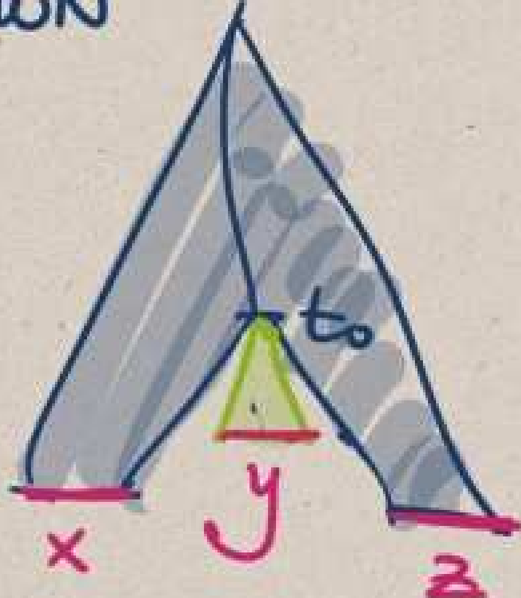
$$|uyv| = |\sigma_{\pi_{t_0}}| \leq |\sigma_{\pi_s}| \leq 2^m = n.$$



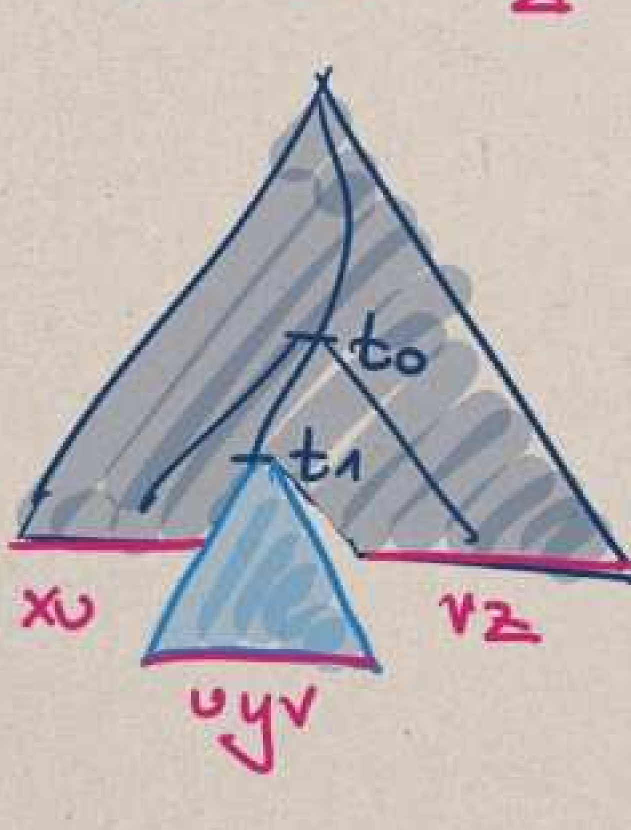
How do we pump?

Pumping down is grafting T_{t_1} into t_0;
pumping up is grafting (iteratively) T_{t_0} into t_1.





$\underbrace{xyz}=x\underbrace{u^0yv^0}z$



$x\underbrace{uuyvv}z =$
$x\underbrace{u^2yv^2}z$

Formally:

$$\Pi_{(0)} := \Pi_{t_1}$$

$$\Pi_{(i+1)} := \mathrm{graft}(\Pi_{t_0}, t_1, \Pi_{(i)})$$

$$\Pi_k := \mathrm{graft}(\Pi, t_1, \Pi_{(k)})$$

Then $\sigma_{\Pi_k} = x u^k y v^k z$.

This proves the pumping lemma.

q.e.d.

§ 3.4 Closure properties

$$L = \{a^n b^n c^n ;\ n > 0\}$$

is not context-free

However:

$$L_0 = \{a^m b^m c^k ;\ m, k > 0\}$$

$$L_1 = \{a^k b^m c^m ;\ m, k > 0\}$$

are context-free.

[$\{a^m b^m ;\ m > 0\}$, $\{b^m c^m ;\ m > 0\}$ are context-free ;

$\{c^k ;\ k > 0\}$, $\{a^k ;\ k > 0\}$ are regular, thus context-free ;

L_0, L_1 are concatenations of these]

but $L = L_0 \cap L_1$.

Therefore, the class of c-f languages cannot be closed under intersections.

So by basic set algebra, it can't be closed under either complementation or difference.

This means that any model of computation that corresponds to c-f grammar cannot have a product construction as in Chapter 2.

WITHOUT PROOF OR DETAILS:

Pushdown automata.

Deterministic automata with one storage unit, a so called STACK.

LIFO LAST-IN-FIRST-OUT storage with symbols pushed on the stack or removed.

Change transition function δ to read information from the top of the stack & modify the stack by removing something or pushing something onto the stack!

Theorem (w/o proof). L is context-free iff there is a pushdown automaton P s.t. $L = \mathcal{L}(P)$.

§ 3.5 Decision problems

Word problem ← SOLVED for Type 1 grammar

Emptiness problem

Equivalence problem

<u>EMPTINESS PROBLEM</u>

If L satisfies PL with pumping # n, then if $L \neq \emptyset$, then L accepts a word w with $|w| < n$.

Which pumping lemma we use doesn't matter in this argument.

So: the solution to the emptiness problem for c-f grammars:

1. Transform G to CNF.
2. Count variables: m.
3. Compute $n := 2^m$
4. Check all words of length $< n$.

Note (without proof):

The EQUIVALENCE PROBLEM for c-f grammars is NOT SOLVABLE, so there is no algorithm that determines on input G, G' c-f whether $\mathcal{L}(G) = \mathcal{L}(G')$.

However, how would one even show this without a precise notion of ALGORITHM?

We urgently need such a notion

$\rightsquigarrow$ Chapter 4!

SUMMARY

	regular (type 3)	context-free (type 2)
Closure properties.		
Concatenation	✓	✓
Union	✓	✓
Intersection	✓	×
Complementation	✓	×
Difference	✓	×
Decision problems.		
Word problem	✓	✓
Emptiness problem	✓	✓
Equivalence problem	✓	×