Theory of Computation
Week 8

Much of the material on this slides comes from the recommended textbook by Elaine Rich

Announcement

Assignment 2 Released

- Due: 17/05/2020

Midterm 1 score released

- Check your score/feedback
- Contact Mohammad if you have a query

Detailed content

Weekly program

- ✓ Week 1 – Background knowledge revision: logic, sets, proof techniques
- ✓ Week 2 – Languages and strings. Hierarchies. Computation. Closure properties
- ✓ Week 3 – Finite State Machines: non-determinism vs. determinism
- ✓ Week 4 – Regular languages: expressions and grammars
- ✓ Week 5 – Non regular languages: pumping lemma. Closure
- ✓ Week 6 – Context-free languages: grammars and parse trees
- ✓ Week 7 – Pushdown automata

➡ **Week 8 – Non context-free languages: pumping lemma and decidability. Closure**

- ❑ Week 9 – Decidable languages: Turing Machines
- ❑ Week 10 – Church-Turing thesis and the unsolvability of the Halting Problem
- ❑ Week 11 – Decidable, semi-decidable and undecidable languages (and proofs)
- ❑ Week 12 – Revision of the hierarchy. Safety-critical systems
- ❑ Week 13 – Extra revision (if needed)

Week 08 Videos

You already know

- What is Pumping Lemma for CFL
  - What is a long string for CFG
  - A intuitive explanation
  - Example

Videos to watch before lecture

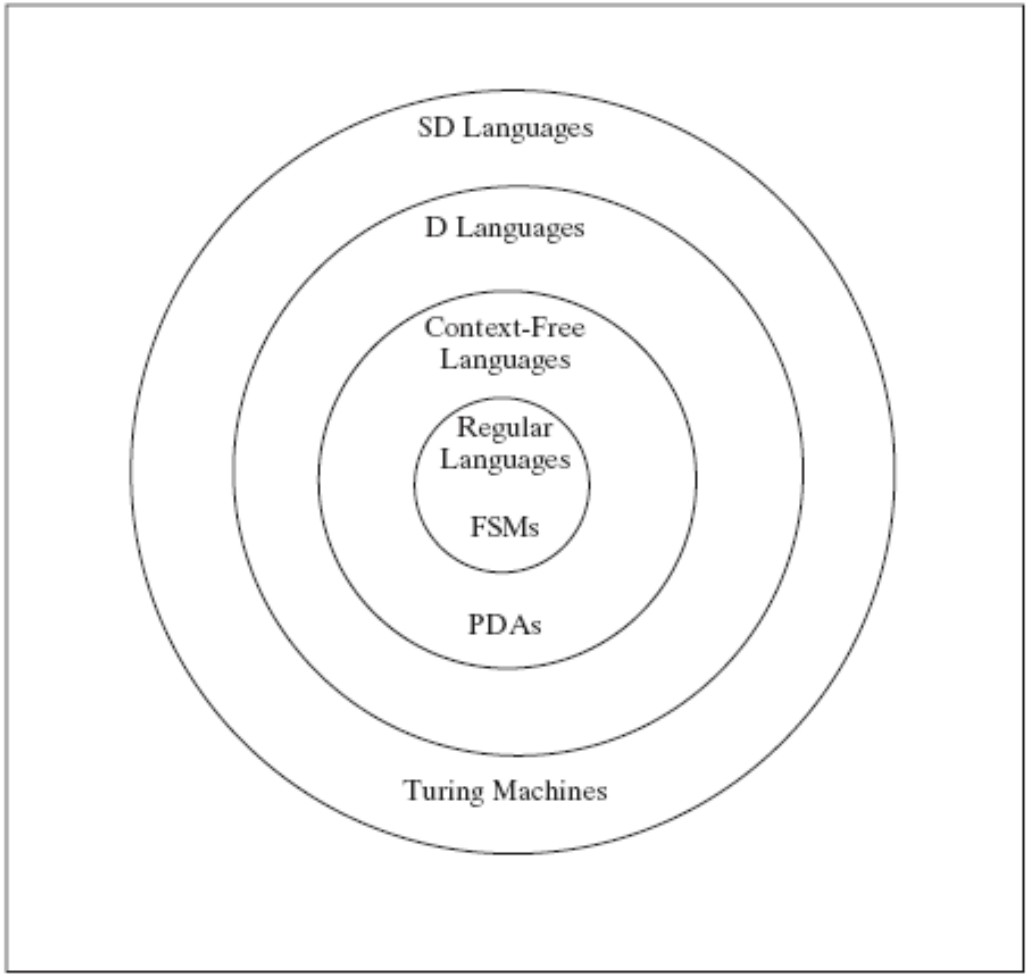
Additional videos to watch for this week

Week 08 Lecture

Pumping Lemma for CFL

- ❑ Relation between CFL and RL
- ❑ How many CFL out there
- ❑ Pumping lemma for CFL
- ❑ Closure properties of CFL
- ❑ Closure on CFL and RL
- ❑ Closure with pumping lemma
- ❑ Deterministic CFL
- ❑ CFL hierarchy

THE HIERARCHY



REGULAR LANGUAGES

A language is ***regular*** iff it is accepted by some FSM

THE PUMPING THEOREM FOR REGULAR LANGUAGES

If L is regular, then every **"long"** string in L is pumpable.

To show that L is not regular, we find one that isn't.

To use the Pumping Theorem to show that a language L is not regular, we must:

1. Choose a string w where $|w| \geq k$. Since we do not know what k is, we must state w in terms of k.
2. Divide the possibilities for y into a set of equivalence classes that can be considered together.
3. For each such class of possible y values where $|xy| \leq k$ and $y \neq \varepsilon$:

   Choose a value for q such that xy^qz is not in L.

CONTEXT-FREE LANGUAGES

A language *L* is ***context-free*** if and only if it is generated by some context-free grammar *G*.

LANGUAGES THAT ARE AND ARE NOT CONTEXT-FREE

a^*b^* is regular.

$A^nB^n = \{a^n b^n : n \geq 0\}$ is context-free but not regular.

$A^nB^nC^n = \{a^n b^n c^n : n \geq 0\}$ is not context-free.

THE REGULAR AND THE CF LANGUAGES

Theorem: The regular languages are a proper subset of the context-free languages.

Proof:

In two parts:

- Every regular language is CF.
- There exists at least one language that is CF but not regular.

THE REGULAR AND THE CF LANGUAGES

Lemma -1 : Every regular language is CF.

Proof: Every FSM is (trivially) a PDA:

Given an FSM $M = (K, \Sigma, \Delta, s, A)$ and elements of δ of the form:

$(\ p,\ c,\ q\ )$

old state, input, new state

Construct a PDA $M' = (K, \Sigma, \{\varnothing\}, \Delta, s, A)$. Each (p, c, q) becomes:

$((\ p,\ c,\ \varepsilon\ ),\ (\ q,\ \varepsilon\ ))$

old state, input, don't look at stack — new state, don't push on stack

In other words, we just don't use the stack.

THE REGULAR AND THE CF LANGUAGES

Lemma -2 : There exists at least one language that is CF but not regular

Proof: $\{a^n b^n, n \geq 0\}$ is context-free but not regular.

So the regular languages are a proper subset of the context-free languages.

THE REGULAR AND THE CF LANGUAGES

There is an uncountable number of languages.

The number of CF languages in **countable infinite.**

Thus there are more languages than there are context-free languages.

So there must exist some languages that are not context-free.

Example: $\{a^n b^n c^n : n \geq 0\}$

SHOWING THAT L IS CONTEXT-FREE

Techniques for showing that a language L is context-free:

1. Exhibit a context-free grammar for L.

2. Exhibit a PDA for L.

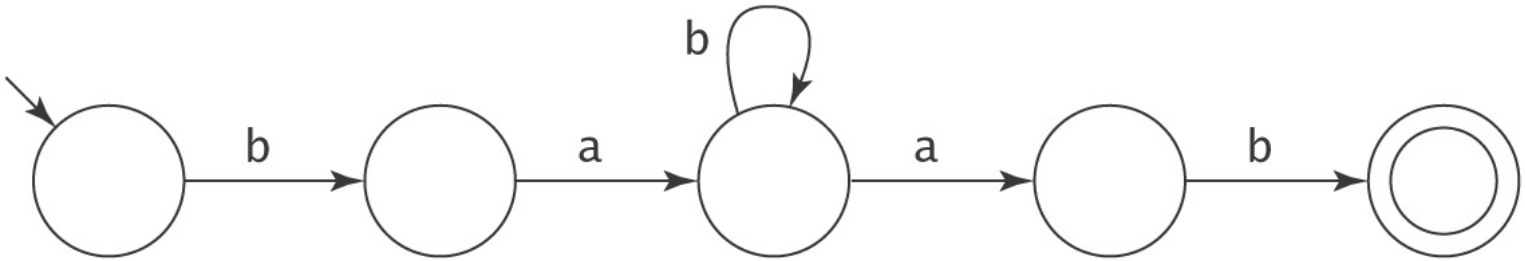
3. Use the closure properties of context-free languages.

Unfortunately, these are weaker than they are for regular languages.

Remember for RL
1. Show that L is finite.
2. Exhibit an FSM for L.
3. Exhibit a regular expression for L.
4. Show that the number of equivalence classes of $\approx_L$ is finite.
5. Exhibit a regular grammar for L.
6. Exploit the closure theorems.

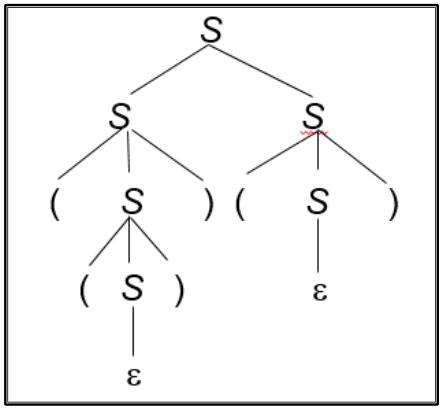
SHOWING THAT L IS NOT CONTEXT-FREE

Remember the pumping argument for regular languages:



A REVIEW OF PARSE TREES

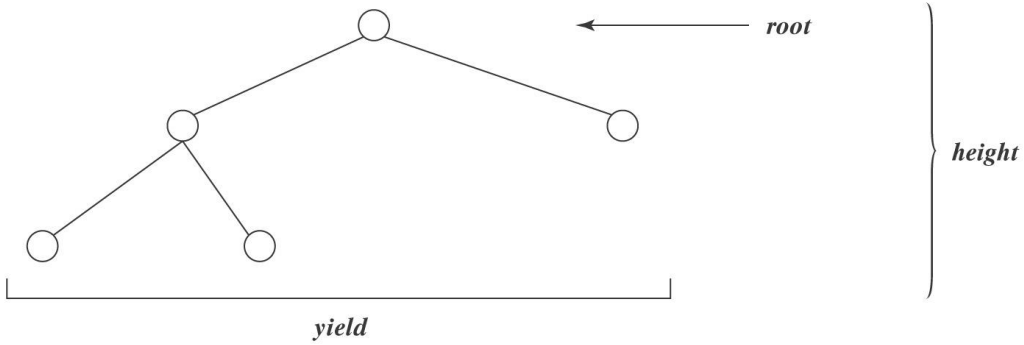
- A parse tree, derived by a grammar $G = (V, \Sigma, R, S)$, is a rooted, ordered tree in which:
- Every leaf node is labeled with an element of $\Sigma \cup \{\varepsilon\}$,
- The root node is labeled S,
- Every other node is labeled with some element of $V - \Sigma$,
- If m is a nonleaf node labeled X and the children of m are labeled $x_1, x_2, \ldots, x_n$, then the rule $X \rightarrow x_1, x_2, \ldots, x_n$ is in R.

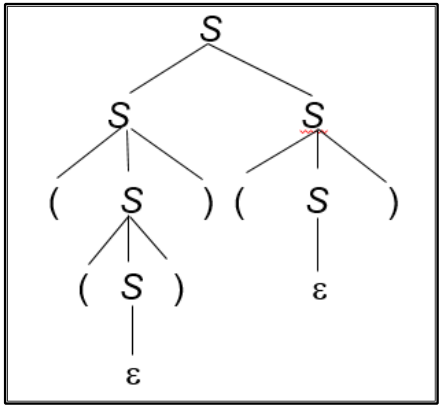


A REVIEW OF PARSE TREES

The ***height*** of a tree is the length of the longest path from the root to any leaf.

The ***branching factor*** of a tree is the largest number of daughter nodes associated with any node in the tree.





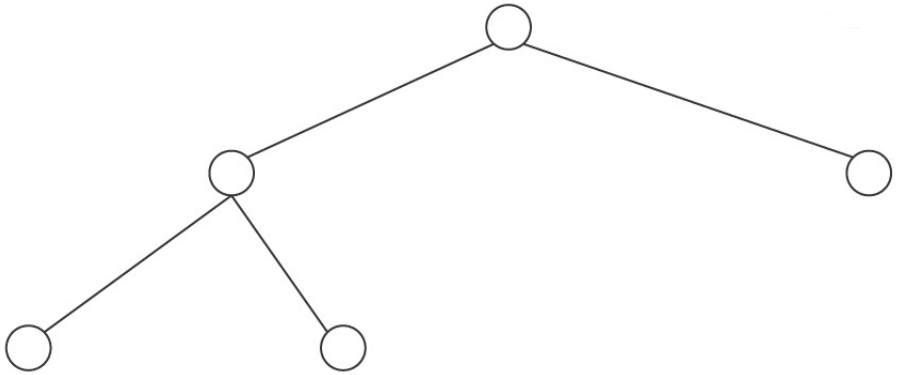
Theorem: The length of the yield of any tree *T* with height *h* and branching factor *b* is $\leq b^h$.

A REVIEW OF PARSE TREES

Given a context-free grammar *G*:

- Let *n* be the number of nonterminal symbols in *G*.
- Let *b* be the branching factor of *G*

Suppose that *T* is generated by *G* and no nonterminal appears more than once on any path:



The maximum height of *T* is:

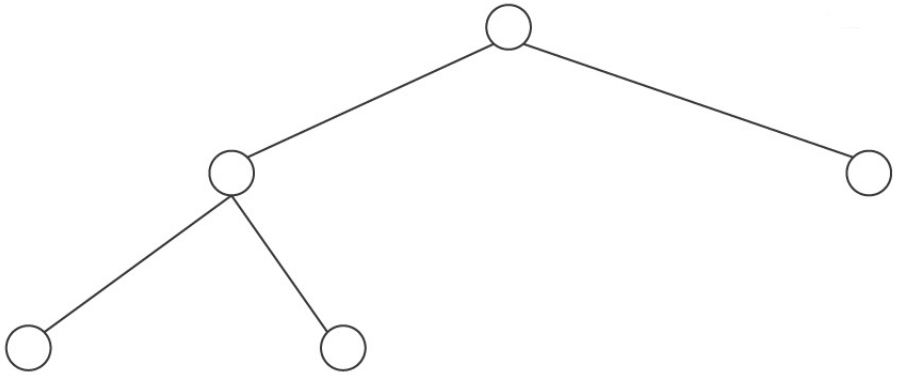
The maximum length of *T*'s yield is:

A REVIEW OF PARSE TREES

Given a context-free grammar *G*:

- Let *n* be the number of nonterminal symbols in *G*.
- Let *b* be the branching factor of *G*

Suppose that *T* is generated by *G* and no nonterminal appears more than once on any path:



The maximum height of *T* is: n

The maximum length of *T*'s yield is: b^n

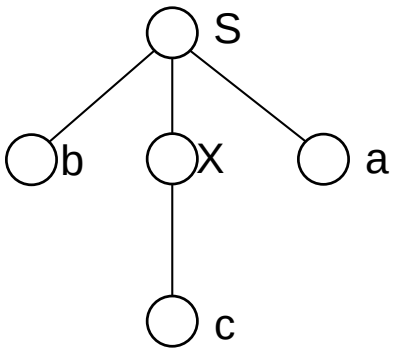
A REVIEW OF PARSE TREES

Given a context-free grammar *G* with the rules

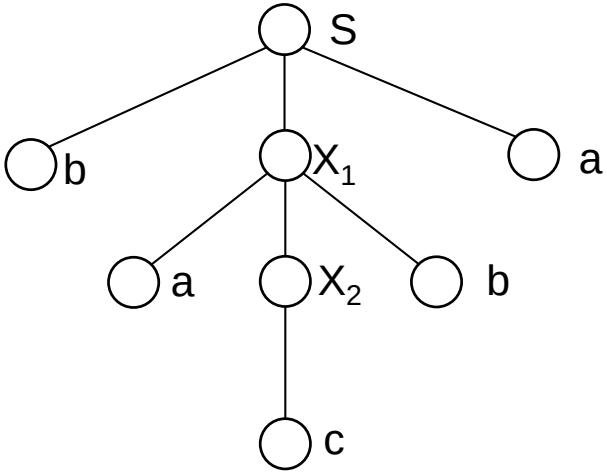
$S \rightarrow bXa$

$X \rightarrow aXb$

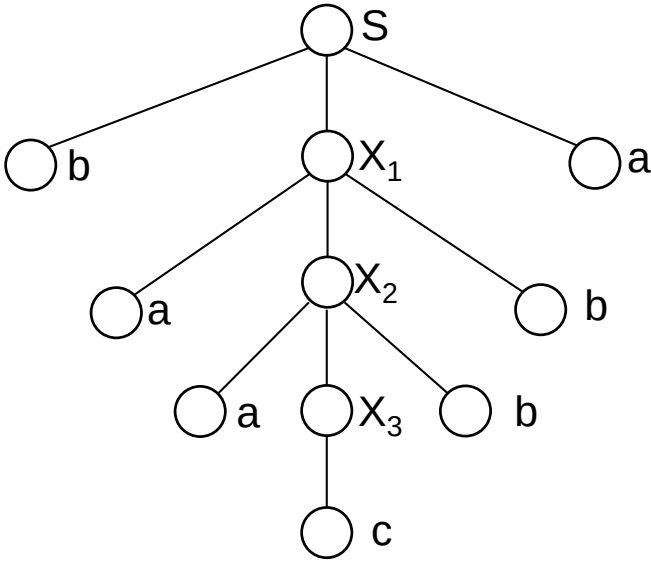
$X \rightarrow c$



yield= bca



yield= bacba

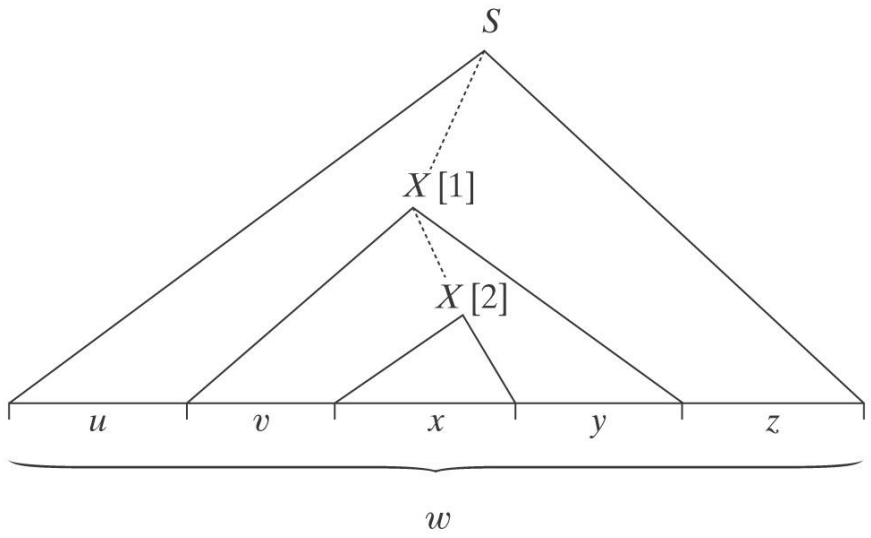


yield= ba^2cb^2a

THE CONTEXT-FREE PUMPING THEOREM

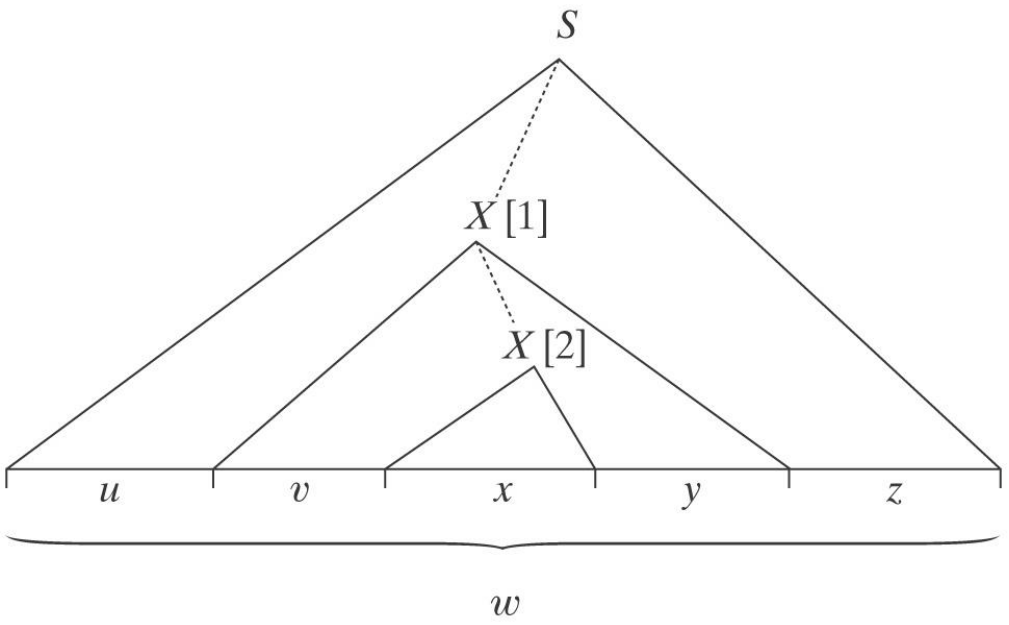
This time we use parse trees, not automata as the basis for our argument.

If w is **"long"**, *i.e.*, $|w| > b^n$
Then its parse trees must look like:

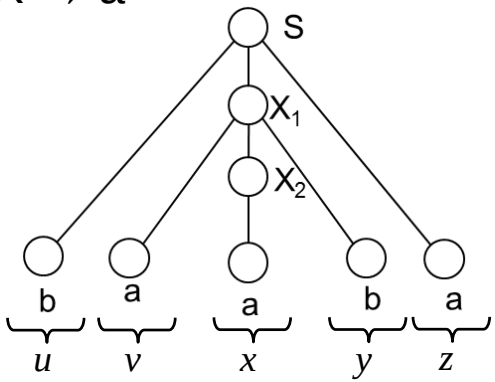


It may be possible that w is generated by more than one parse tree. Choose one such tree such that there's no other with fewer nodes.

THE CONTEXT-FREE PUMPING THEOREM



S → bXa
X → aXb
X → a

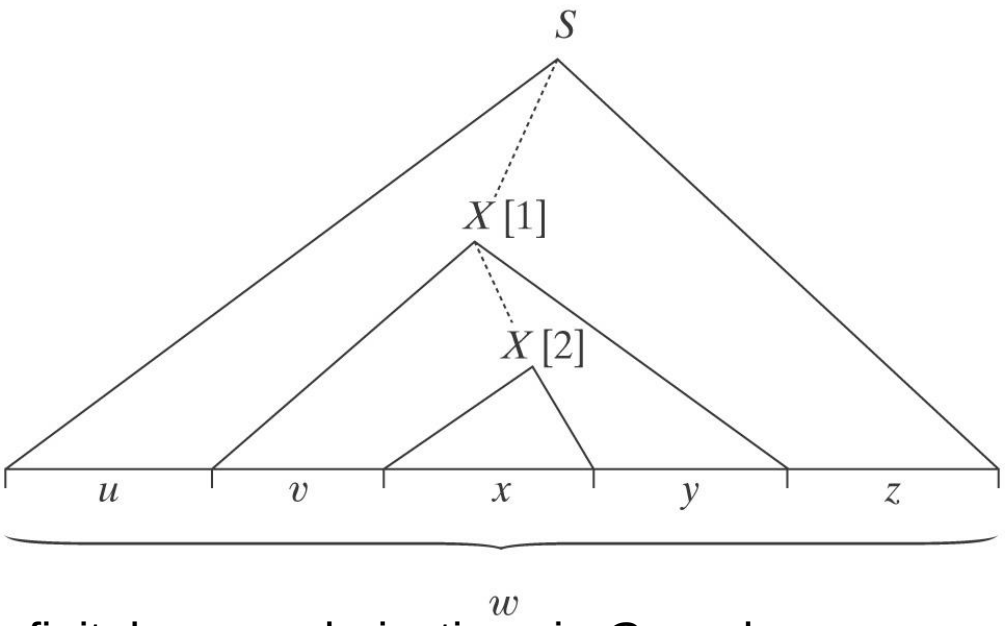


Sketch of derivation: $S \Rightarrow^* uXz \Rightarrow^* uvXyz, \Rightarrow^* uvxyz,$

But note that there is another derivation in G: $S \Rightarrow^* uXz \Rightarrow^* uxz,$

if at the point labeled [1], the nonrecursive $rule_2$ is used.
So uxz is also in $L(G)$.

THE CONTEXT-FREE PUMPING THEOREM



There are infinitely many derivations in G, such as:

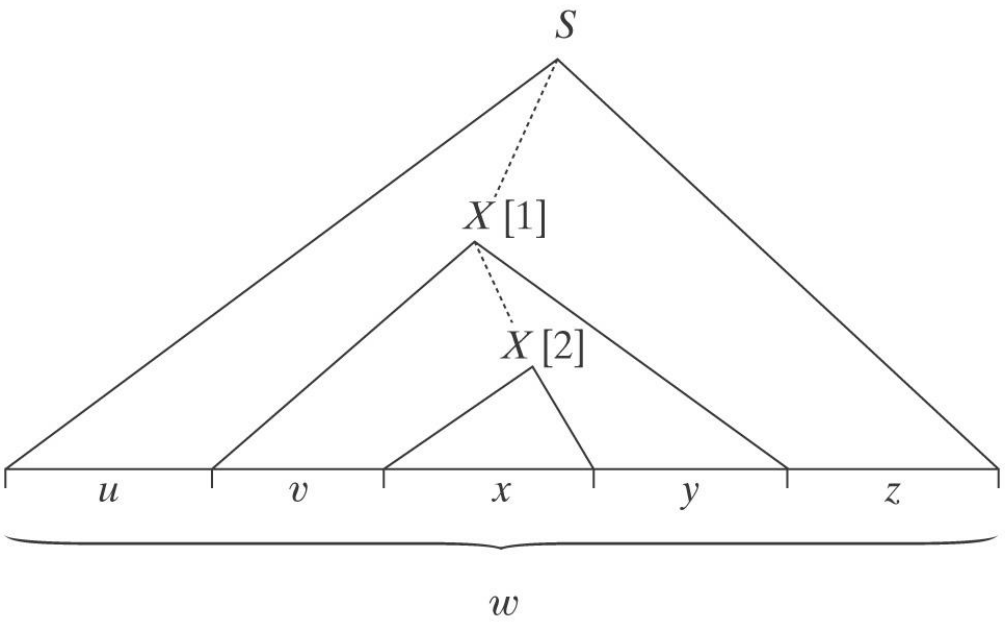
$$S \Rightarrow^* uXz \Rightarrow^* uvXyz \Rightarrow^* uvvXyyz \Rightarrow^* uvvxyyz$$

Those derivations produce the strings:

uv^2xy^2z, uv^3xy^3z, …

So all of those strings are also in $L(G)$.

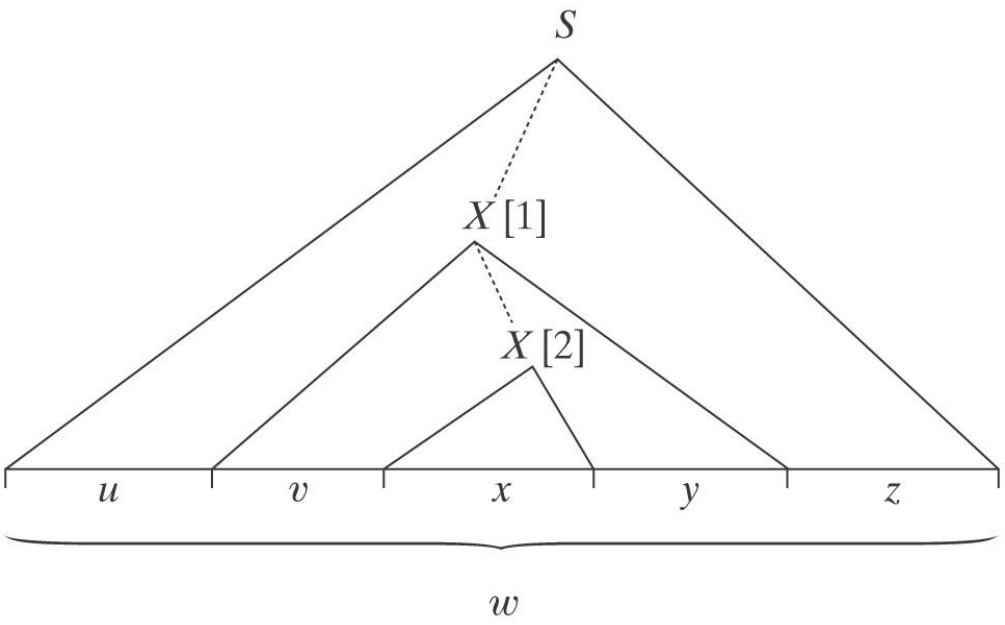
THE CONTEXT-FREE PUMPING THEOREM



If $rule_1 = X \rightarrow X\mathtt{a}$, we could get $v = \varepsilon$.

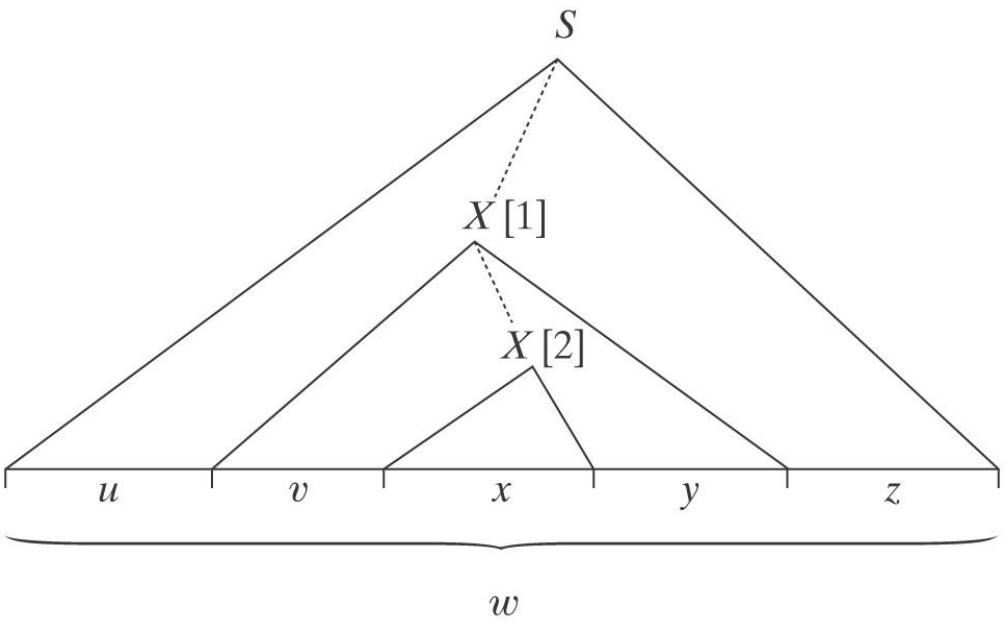
If $rule_1 = X \rightarrow \mathtt{a}X$, we could get $y = \varepsilon$.

THE CONTEXT-FREE PUMPING THEOREM



But it is not possible that both v and y are ε. If they were, then the derivation $S \Rightarrow^* uXz \Rightarrow^* uxz$ would also yield w and it would create a parse tree with fewer nodes. But that contradicts the assumption that we started with a tree with the smallest possible number of nodes.

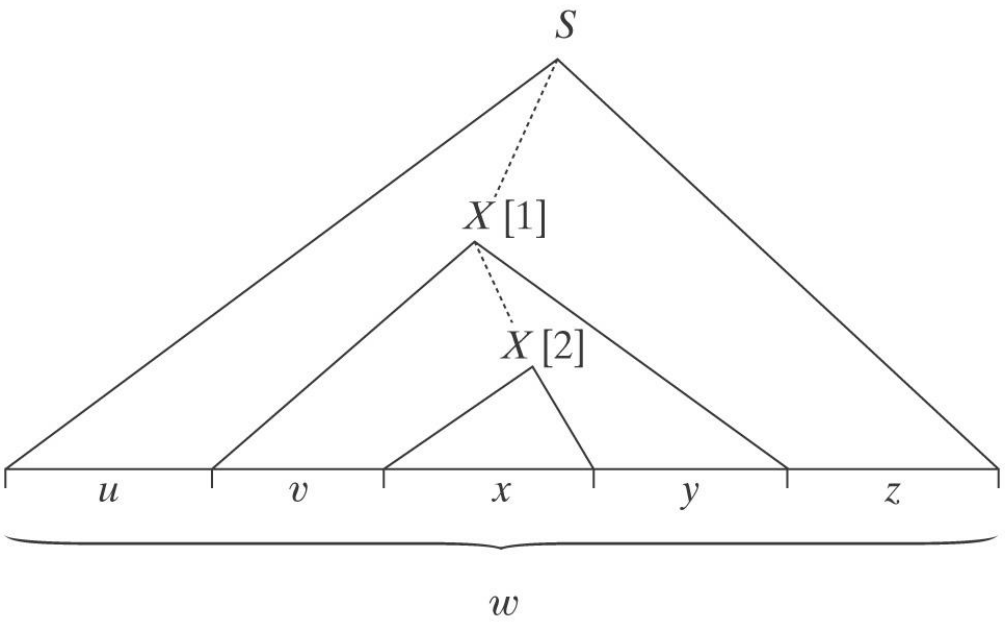
THE CONTEXT-FREE PUMPING THEOREM



The height of the subtree rooted at [1] is at most:

So the yield of the subtree rooted at [1] is:

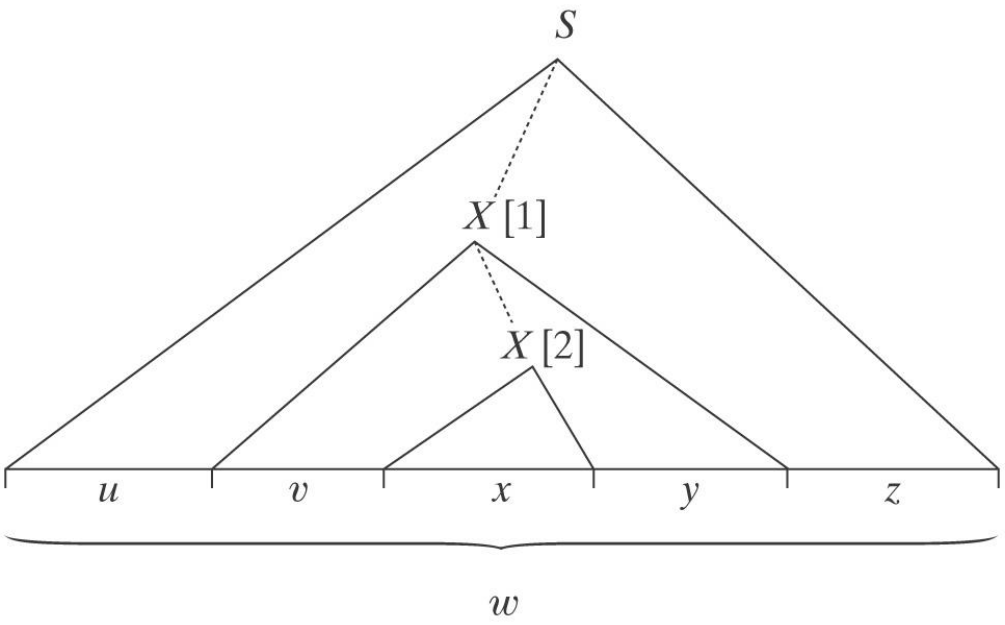
THE CONTEXT-FREE PUMPING THEOREM



The height of the subtree rooted at [1] is at most: $n + 1$

So the yield of the subtree rooted at [1] is: $|vxy| \leq b^{n+1}$.

THE CONTEXT-FREE PUMPING THEOREM



If L is a context-free language, then

$\exists k \geq 1$ ($\forall$ strings $w \in L$, where $|w| \geq k$

($\exists u, v, x, y, z$ ($w = uvxyz$,

$vy \neq \varepsilon$,

$|vxy| \leq k$ and

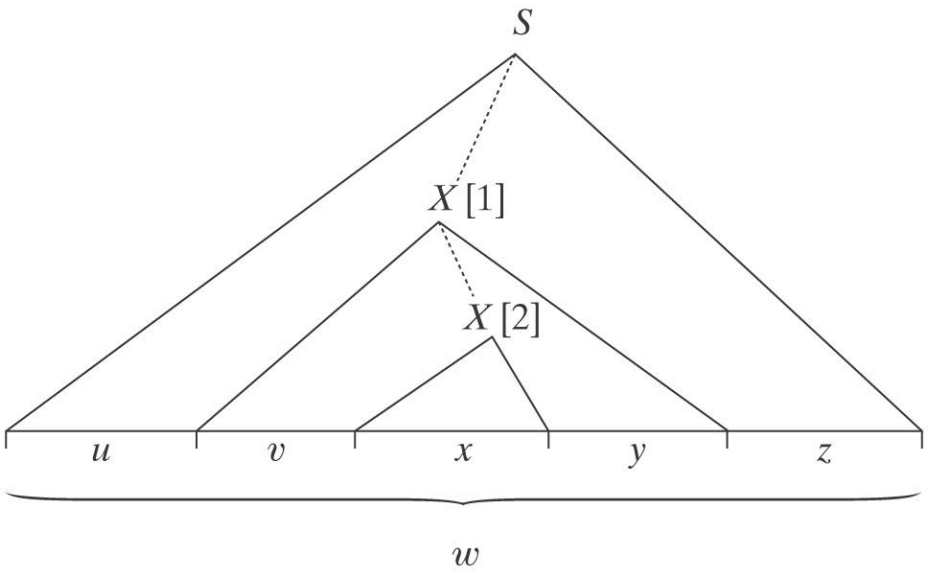
$\forall q \geq 0$ ($uv^q xy^q z$ is in L)))).

THE CONTEXT-FREE PUMPING THEOREM

What is k:

k serves two roles:

- How long must *w* be to guarantee it is pumpable?
- What's the bound on |*vxy*|?

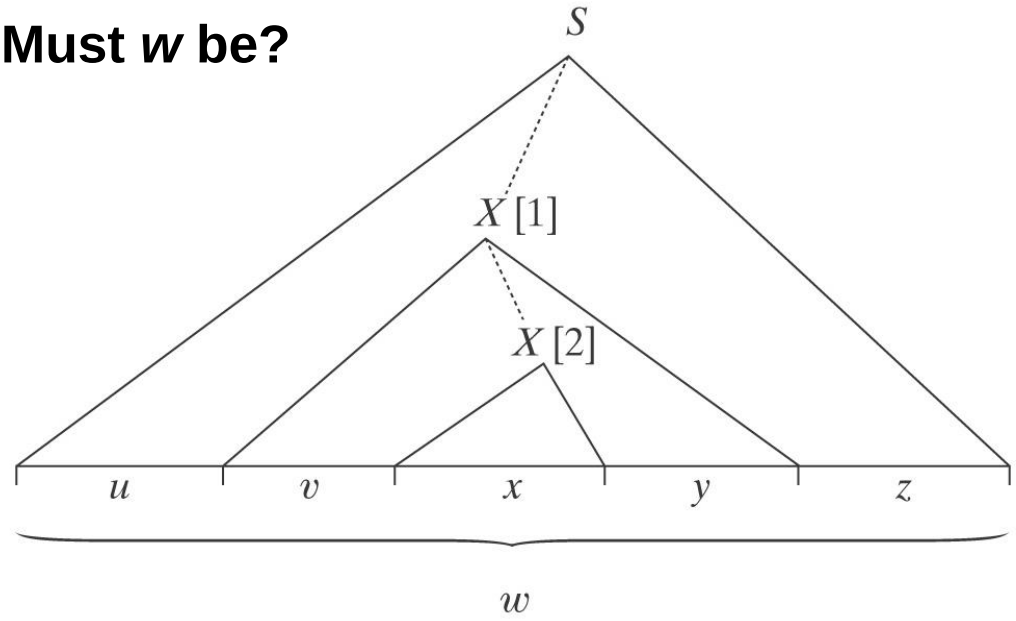


Let *n* be the number of nonterminals in *G*.

Let *b* be the branching factor of *G*.

THE CONTEXT-FREE PUMPING THEOREM

How Long Must *w* be?



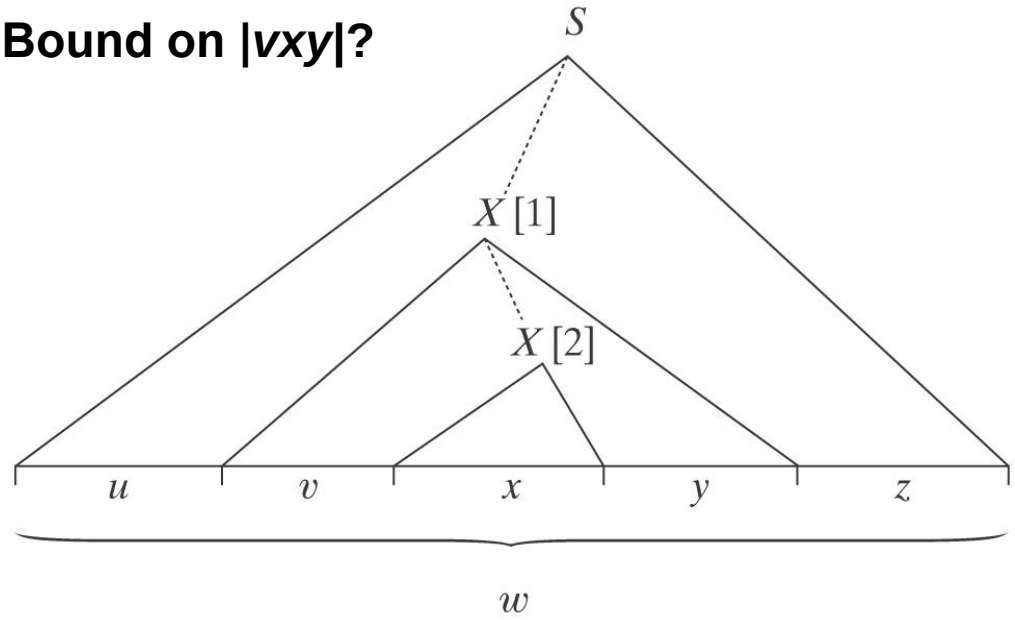
If *height*(*T*) > *n*, then some nonterminal occurs more than once on some path.
So *T* is pumpable.

If *height*(*T*) ≤ *n*, then |*uvxyz*| ≤ b^n.

So if |*uvxyz*| > b^n, *w* = *uvxyz* must be pumpable.

THE CONTEXT-FREE PUMPING THEOREM

What's the Bound on |*vxy*|?



Assume that we are considering the bottom-most two instances of a repeated nonterminal (X). Then the yield of the upper one (X[1]) has length at most b^{n+1}.

Assuming $b \geq 2$, $b^{n+1} > b^n$.

So let $k = b^{n+1}$.

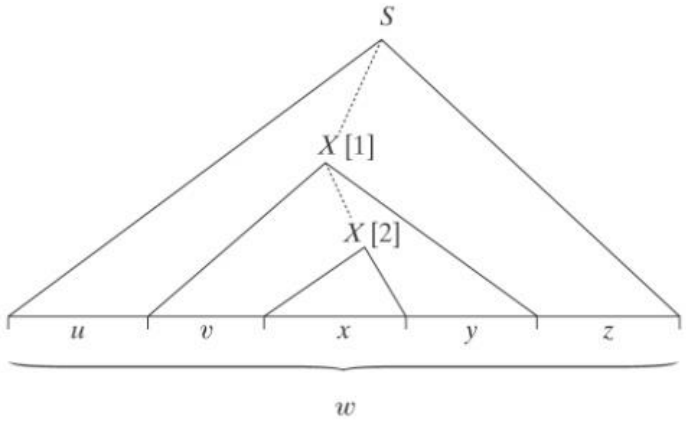
THE CONTEXT-FREE PUMPING THEOREM

If L is a context-free language, then $\exists k \geq 1$, such that

$\forall$ strings $w \in L$, where $|w| \geq k$, $\exists\ u, v, x, y, z$, such that:

$w = uvxyz$, and $vy \neq \varepsilon$, and $|vxy| \leq k$, and

$\forall q \geq 0$, $uv^q xy^q z$ is in L.



THE CONTEXT-FREE PUMPING THEOREM

Proof:

L is generated by some CFG $G = (V, \Sigma, R, S)$ with *n* nonterminal symbols and branching factor *b*. Let *k* be b^{n+1}.

The longest string that can be generated by *G* with no repeated nonterminals in the resulting parse tree has length b^n.

Assuming that $b \geq 2$, it must be the case that $b^{n+1} > b^n$. So let *w* be any string in $L(G)$ where $|w| \geq k$.

Let *T* be any smallest parse tree for *w*. *T* must have height at least $n + 1$.

Choose some path in *T* of length at least $n + 1$. Let *X* be the bottom-most repeated nonterminal along that path. Then *w* can be rewritten as *uvxyz*. The tree rooted at [1] has height at most $n + 1$.

Thus its yield, *vxy*, has length less than or equal to b^{n+1}, which is *k*. $vy \neq \varepsilon$ since if *vy* were ε then there would be a smaller parse tree for *w* and we chose *T* so that that wasn't so. *uxz* must be in *L* because $rule_2$ could have been used immediately at [1].

For any $q \geq 1$, uv^qxy^qz must be in *L* because $rule_1$ could have been used *q* times before finally using $rule_2$.

REGULAR VS CF PUMPING THEOREMS

Similarities:

- We choose a long string *w*, the string to be pumped.
- We choose a value for *q* that shows that *w* isn't pumpable.
- We may apply closure theorems before we start.

Differences:

- Two regions, *v* and *y*, must be pumped in tandem.
- We don't know anything about where in the strings *v* and *y* will fall. All we know is that they are reasonably "close together", i.e., $|vxy| \leq k$.
- Either *v* or *y* could be empty, although not both.

AN EXAMPLE OF PUMPING: $A^nB^nC^n$

$A^nB^nC^n = \{a^n b^n c^n, n \geq 0\}$

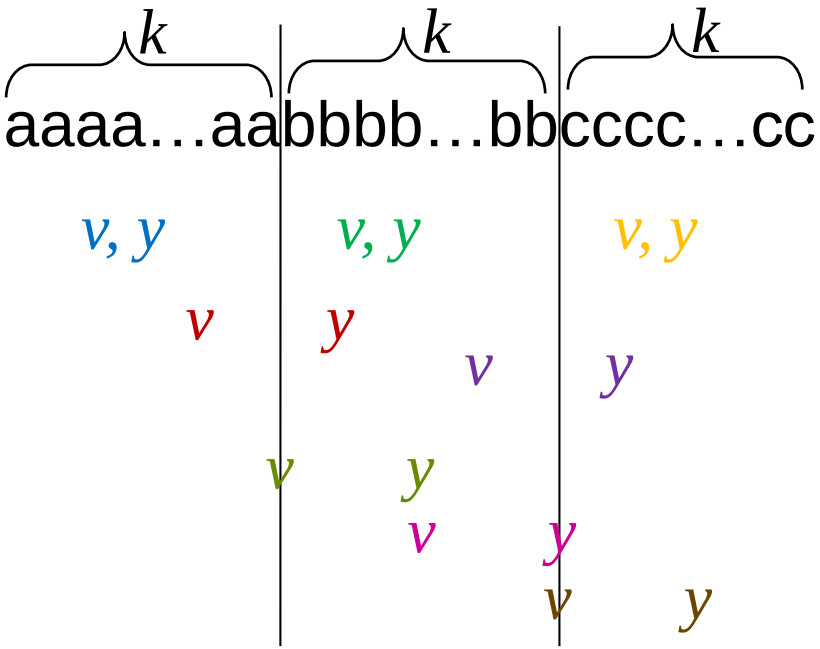
AN EXAMPLE OF PUMPING: $A^nB^nC^n$

$A^nB^nC^n = \{a^n b^n c^n, n \geq 0\}$

Choose $w = a^k\ b^k\ c^k$

1 | 2 | 3

$w = uvxyz$, and
$vy \neq \varepsilon$, and $|vxy| \leq k$,



AN EXAMPLE OF PUMPING: $A^nB^nC^n$

$A^nB^nC^n = \{a^n b^n c^n, n \geq 0\}$

Choose $w = a^k b^k c^k$

1 | 2 | 3

If either *v* or *y* spans regions, then let $q = 2$ (i.e., pump in once). The resulting string will have letters out of order and thus not be in $A^nB^nC^n$.

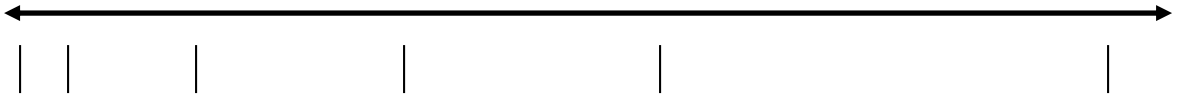
If both *v* and *y* each contain only one distinct character then set *q* to 2. Additional copies of at most two different characters are added, leaving the third unchanged. There are no longer equal numbers of the three letters, so the resulting string is not in $A^nB^nC^n$.

AN EXAMPLE OF PUMPING: $\{a^{n^2}, n \geq 0\}$

$L = \{a^{n^2}; n \geq 0\}$

The elements of L:

n	w
0	ε
1	a^1
2	a^4
3	a^9
4	a^{16}
5	a^{25}
6	a^{36}



AN EXAMPLE OF PUMPING: $\{a^{n^2}, n \geq 0\}$

$L = \{a^{n^2}; n \geq 0\}$

If $n = k^2$, then $n^2 = k^4$. Let $w = a^{k^4}$.

AN EXAMPLE OF PUMPING: $\{a^{n^2}, n \geq 0\}$

$L = \{a^{n^2}, n \geq 0\}$. If $n = k^2$, then $n^2 = k^4$. Let $w = a^{k^4}$.

$vy = a^p$, for some nonzero p.

Set q to 2. The resulting string, s, is a^{k^4+p}. It must be in L. But it isn't because it is too short:

w:	next longer string in L:
$(k^2)^2$ a's	$(k^2 + 1)^2$ a's
k^4 a's	$k^4 + 2k^2 + 1$ a's

For s to be in L, $p = |vy|$ would have to be at least $2k^2 + 1$.

But $|vxy| \leq k$, so p can't be that large. Thus s is not in L and L is not context-free.

ANOTHER EXAMPLE OF PUMPING

$L = \{a^n b^m a^n, n, m \geq 0 \text{ and } n \geq m\}$.

Let w =

ANOTHER EXAMPLE OF PUMPING

$L = \{a^n b^m a^n, n, m \geq 0 \text{ and } n \geq m\}$.

Let $w = a^k b^k a^k$

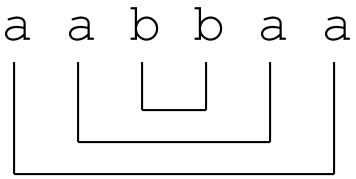
aaa … aaabbb … bbbaaa … aaa

| 1 | 2 | 3 |

Case 1: v or y crosses the region
Case 2: (1,1) → both v and y in region 1
Case 3: (2,2) → both v and y in region 2
Case 4: (3,3) → both v and y in region 3
Case 5: (1,2) → v falls in region 1 and y falls in region 2
Case 6: (2,3) → v falls in region 2 and y falls in region 3
Case 7: (1,3) → v falls in region 1 and y falls in region 3 ??

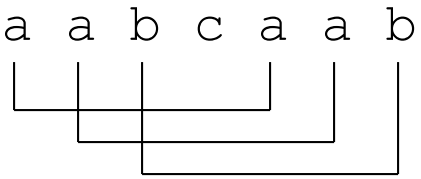
CROSS-SERIAL DEPENDENCIES

PalEven = $\{ww^R : w \in \{a, b\}^*\}$



The dependencies are nested.

WcW = $\{wcw : w \in \{a, b\}^*\}$



Cross-serial dependencies.

WcW = {*wcw* : *w* ∈ {a, b}*}

WcW = {*w*c*w* : *w* ∈ {a, b}*}

Let $w = a^k b^k c a^k b^k$.

```
aaa … aaabbb …  bbbcaaa … aaabbb … bbb
|     1    |      2     |3|    4    |     5    |
```

Call the part before c the left side and the part after c the right side.

- If *v* or *y* overlaps region 3, set *q* to 0. The resulting string will no longer contain a c.
- If both *v* and *y* occur before region 3 or they both occur after region 3, then set *q* to 2. One side will be longer than the other.
- If either *v* or *y* overlaps region 1, then set *q* to 2. In order to make the right side match, something would have to be pumped into region 4. Violates $|vxy| \leq k$.
- If either *v* or *y* overlaps region 2, then set *q* to 2. In order to make the right side match, something would have to be pumped into region 5. Violates $|vxy| \leq k$.

CLOSURE THEOREMS FOR CONTEXT-FREE LANGUAGES

The context-free languages are closed under:

- Union
- Concatenation
- Kleene star
- Reverse
- Letter substitution

CLOSURE UNDER UNION

Let $G_1 = (V_1, \Sigma_1, R_1, S_1)$, and
$G_2 = (V_2, \Sigma_2, R_2, S_2)$.

Assume that G_1 and G_2 have disjoint sets of nonterminals, not including S.

Let $L = L(G_1) \cup L(G_2)$.

We can show that L is CF by exhibiting a CFG for it:

CLOSURE UNDER UNION

Let $G_1 = (V_1, \Sigma_1, R_1, S_1)$, and
$G_2 = (V_2, \Sigma_2, R_2, S_2)$.

Assume that G_1 and G_2 have disjoint sets of nonterminals, not including S.

Let $L = L(G_1) \cup L(G_2)$.

We can show that L is CF by exhibiting a CFG for it:

$$G = (V_1 \cup V_2 \cup \{S\}, \Sigma_1 \cup \Sigma_2, R_1 \cup R_2 \cup \{S \to S_1, S \to S_2\}, S)$$

CLOSURE UNDER CONCATENATION

Let $G_1 = (V_1, \Sigma_1, R_1, S_1)$, and
$G_2 = (V_2, \Sigma_2, R_2, S_2)$.

Assume that G_1 and G_2 have disjoint sets of nonterminals, not including S.

Let $L = L(G_1)L(G_2)$.

We can show that L is CF by exhibiting a CFG for it:

CLOSURE UNDER CONCATENATION

Let $G_1 = (V_1, \Sigma_1, R_1, S_1)$, and
$G_2 = (V_2, \Sigma_2, R_2, S_2)$.

Assume that G_1 and G_2 have disjoint sets of nonterminals, not including S.

Let $L = L(G_1)L(G_2)$.

We can show that L is CF by exhibiting a CFG for it:

$$G = (V_1 \cup V_2 \cup \{S\}, \Sigma_1 \cup \Sigma_2, R_1 \cup R_2 \cup \{S \rightarrow S_1\ S_2\}, S)$$

CLOSURE UNDER KLEENE STAR

Let $G = (V, \Sigma, R, S_1)$.

Assume that G does not have the nonterminal S.

Let $L = L(G)^*$.

We can show that L is CF by exhibiting a CFG for it:

CLOSURE UNDER KLEENE STAR

Let $G = (V, \Sigma, R, S_1)$.

Assume that G does not have the nonterminal S.

Let $L = L(G)^*$.

We can show that L is CF by exhibiting a CFG for it:

$$G = (V_1 \cup \{S\}, \Sigma_1, \\ R_1 \cup \{S \to \varepsilon, S \to S\, S_1\}, \\ S)$$

CLOSURE UNDER REVERSE

$L^R = \{w \in \Sigma^* : w = x^R \text{ for some } x \in L\}$.

Let $G = (V, \Sigma, R, S)$ be in Chomsky normal form.

Every rule in G is of the form $X \rightarrow BC$ or $X \rightarrow a$, where X, B, and C are elements of $V - \Sigma$ and $a \in \Sigma$.

- $X \rightarrow a$: $L(X) = \{a\}$. $\{a\}^R = \{a\}$.
- $X \rightarrow BC$: $L(X) = L(B)L(C)$. $(L(B)L(C))^R = L(C)^R L(B)^R$.

Construct, from G, a new grammar G', such that $L(G') = L^R$:
$G' = (V_G, \Sigma_G, R', S_G)$, where R' is constructed as follows:

- For every rule in G of the form $X \rightarrow BC$, add to R' the rule $X \rightarrow CB$.
- For every rule in G of the form $X \rightarrow a$, add to R' the rule $X \rightarrow a$.

WHAT ABOUT INTERSECTION AND COMPLEMENT?

Closure under complement implies closure under intersection, since:

$$L_1 \cap L_2 = \neg(\neg L_1 \cup \neg L_2)$$

But are the CFLs closed under either complement or intersection?

We can prove closure for regular languages two different ways:

1. Given a DFSM for L, construct a DFSM for $\neg L$ by swapping accepting and rejecting states. If closed under complement and union, must be closed under intersection.

2. Given an FSM for L_1 and L_2, construct another FSM for $L_1 \cap L_2$ by simulating the parallel operation of the two original machines, using states that are the Cartesian product of the sets of states of the two original machines.

Does either work here?

CLOSURE UNDER INTERSECTION

The context-free languages are not closed under intersection:

The proof is by counterexample. Let:

$L_1 = \{a^n b^n c^m : n, m \geq 0\}$ /* equal a's and b's.
$L_2 = \{a^m b^n c^n : n, m \geq 0\}$ /* equal b's and c's.

Both L_1 and L_2 are context-free, since there exist straightforward context-free grammars for them.

But now consider:

$L = L_1 \cap L_2 =$

CLOSURE UNDER INTERSECTION

The context-free languages are not closed under intersection:

The proof is by counterexample. Let:

$L_1 = \{a^n b^n c^m : n, m \geq 0\}$ /* equal a's and b's.
$L_2 = \{a^m b^n c^n : n, m \geq 0\}$ /* equal b's and c's.

Both L_1 and L_2 are context-free, since there exist straightforward context-free grammars for them.

But now consider:

$L = L_1 \cap L_2 = \{a^n b^n c^n : n \geq 0\}$

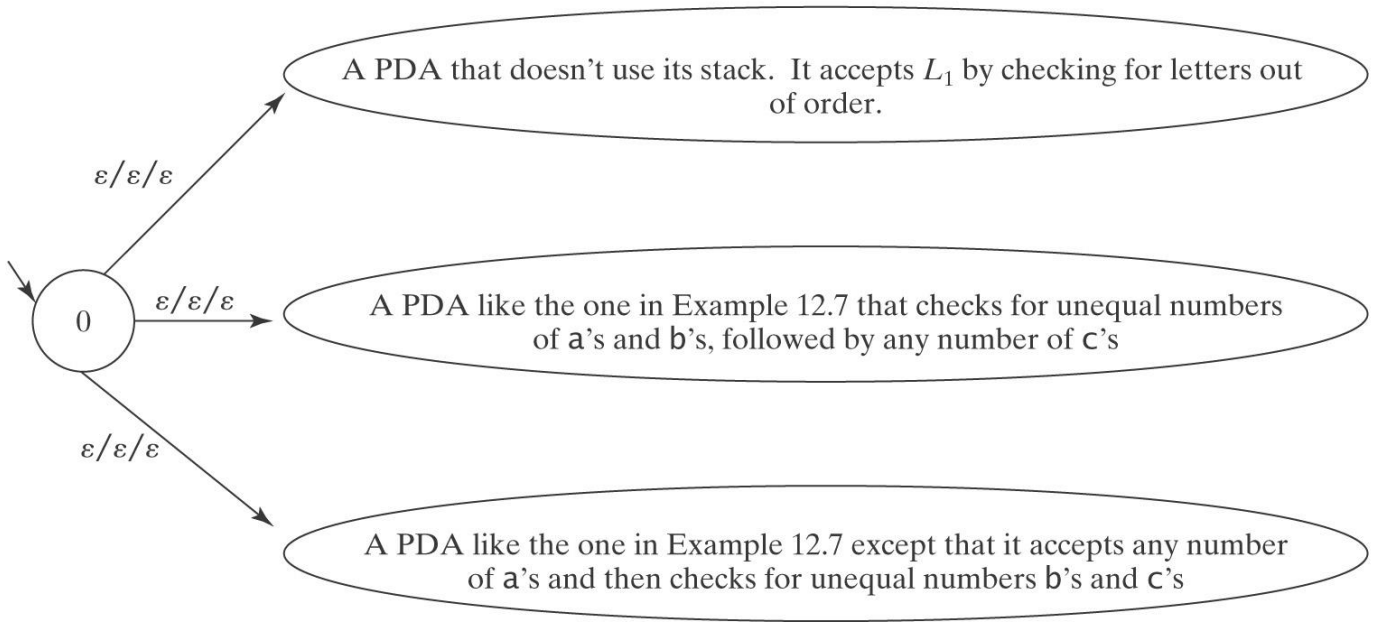
CLOSURE UNDER COMPLEMENT

$L_1 \cap L_2 = \neg(\neg L_1 \cup \neg L_2)$

The context-free languages are closed under union, so if they were closed under complement, they would be closed under intersection (which they are not).

CLOSURE UNDER COMPLEMENT

$\neg A^nB^nC^n$ is context-free:



But $\neg(\neg A^nB^nC^n) = A^nB^nC^n$ is not context-free.

CLOSURE UNDER DIFFERENCE

Are the context-free languages closed under difference?

CLOSURE UNDER DIFFERENCE

Are the context-free languages closed under difference?

$\neg L = \Sigma^* - L$.

Σ^* is context-free. So, if the context-free languages were closed under difference, the complement of any context-free language would necessarily be context-free. But we just showed that that is not so.

INTERSECTION OF A CONTEXT-FREE LANGUAGE AND A REGULAR LANGUAGE IS CONTEXT-FREE

$L = L(M_1)$, a PDA = $(K_1, \Sigma, \Gamma_1, \Delta_1, s_1, A_1)$.
$R = L(M_2)$, a deterministic FSM = $(K_2, \Sigma, \delta, s_2, A_2)$.

We construct a new PDA, M_3, that accepts $L \cap R$ by simulating the parallel execution of M_1 and M_2.

$M = (K_1 \times K_2, \Sigma, \Gamma_1, \Delta, (s_1, s_2), A_1 \times A_2)$.

Insert into Δ:

For each rule $((q_1, a, \beta), (p_1, \gamma))$ in Δ_1,
and each rule (q_2, a, p_2) in δ,
add $(((q_1, q_2), a, \beta), ((p_1, p_2), \gamma))$.

For each rule $((q_1, \varepsilon, \beta), (p_1, \gamma)$ in Δ_1,
and each state q_2 in K_2,
add $(((q_1, q_2), \varepsilon, \beta), ((p_1, q_2), \gamma))$.

This works because: we can get away with only one stack.

DIFFERENCE BETWEEN A CONTEXT-FREE LANGUAGE AND A REGULAR LANGUAGE IS CONTEXT-FREE

Theorem: The difference ($L_1 - L_2$) between a context-free language L_1 and a regular language L_2 is context-free.

Proof: $L_1 - L_2 = L_1 \cap \neg L_2$.

If L_2 is regular then so is $\neg L_2$.

If L_1 is context-free, so is $L_1 \cap \neg L_2$.

AN EXAMPLE: A FINITE NUMBER OF EXCEPTIONS

Let:

$L = \{a^n b^n : n \geq 0 \text{ and } n \neq 1776\}$.

Alternatively:

$L = \{a^n b^n : n \geq 0\} - \{a^{1776} b^{1776}\}$.

$\{a^n b^n : n \geq 0\}$ is context-free.
$\{a^{1776} b^{1776}\}$ is regular.

DON'T TRY TO USE CLOSURE BACKWARDS

One Closure Theorem:

If L_1 and L_2 are context free, then so is

$$L_3 = L_1 \cup L_2.$$

But what if L_3 and L_1 are context free? What can we say about L_2?

$$L_3 = L_1 \cup L_2.$$

DON'T TRY TO USE CLOSURE BACKWARDS

One Closure Theorem:

If L_1 and L_2 are context free, then so is

$$L_3 = L_1 \cup L_2.$$

But what if L_3 and L_1 are context free? What can we say about L_2?

$$L_3 = L_1 \cup L_2.$$

Example: $a^n b^n c^* = a^n b^n c^* \cup a^n b^n c^n$.

USING THE CLOSURE THEOREMS WITH THE PUMPING THEOREM

Let WW = {*ww* : *w* ∈ {a, b}* }.

Let's try pumping: Choose *w* = $(\mathtt{ab})^{2k}$

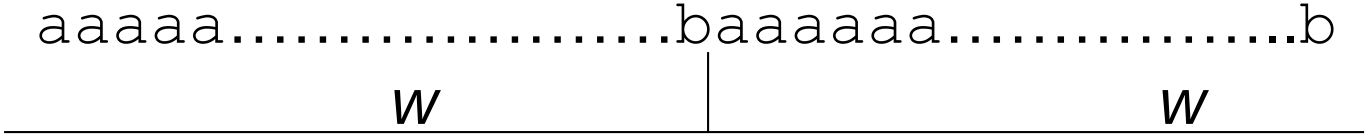
(Don't get confused about the two uses of *w*.)

But this pumps fine with *v* = ab and *y* = ab

EXPLOITING REGIONS

WW = {*ww* : *w* ∈ {a, b}* }.

Choose the string $a^k b a^k b$.

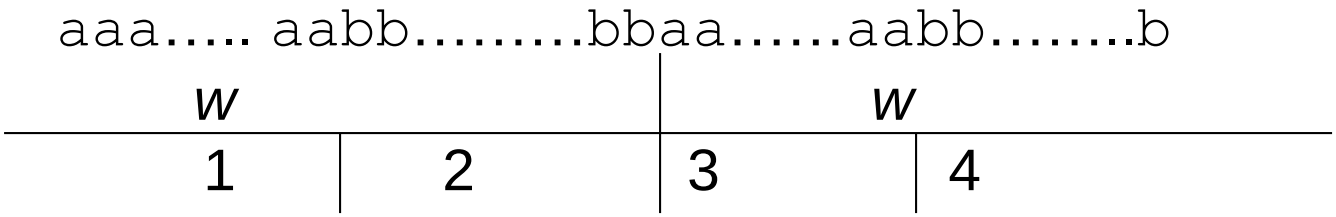


But this also pumps fine [with v=a in the first group of a's and y=a in the second group of a's].

MAKE ALL REGIONS "LONG"

WW = {*ww* : *w* ∈ {a, b}* }.

Choose the string $a^k b^k a^k b^k$.



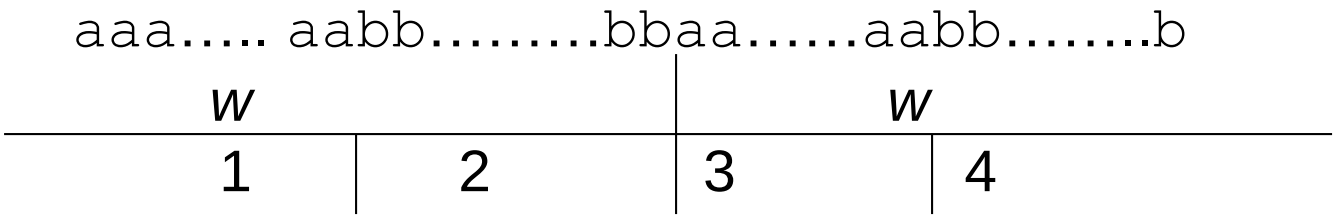
Now we list the possibilities:
(1, 1), (2, 2), (3, 3), (4, 4), (1, 2), (2, 3), (3, 4), (1/2, 2), (1, 1/2), (2/3, 3),…

Whenever *v* or *y* spans regions, we'll no longer have a string of the same form, but that's okay given the definition of *L*.

USING INTERSECTION WITH A REGULAR LANGUAGE

WW = {*ww* : *w* ∈ {a, b}* }.

Recall our last choice of *w*: $a^k b^k a^k b^k$.



But let's consider *L'* = *L* ∩

USING INTERSECTION WITH A REGULAR LANGUAGE

WW = {*ww* : *w* ∈ {a, b}* }.

But let's consider *L'* = *L* ∩ a*b*a*b*.

L' is not context-free. Let *w* = $a^k b^k a^k b^k$.

aaa….. aabb……..bbaa……aabb……..b

w *w*

1 2 3 4

Why are the Context-Free Languages Not Closed under Complement, Intersection and Subtraction But the Regular Languages Are?

Given an NDFSM M_1, build an FSM M_2 such that $L(M_2) = \neg L(M_1)$:

1. From M_1, construct an equivalent deterministic FSM M', using *ndfsmtodfsm*.
2. If M' is described with an implied dead state, add the dead state and all required transitions to it.
3. Begin building M_2 by setting it equal to M'. Then swap the accepting and the nonaccepting states. So:

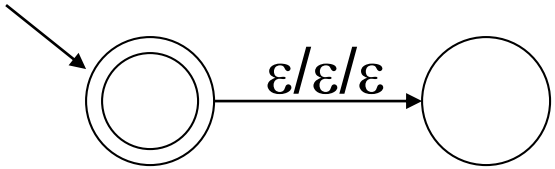
$$M_2 = (K_{M'}, \Sigma, \delta_{M'}, s_{M'}, K_{M'} - A_{M'}).$$

But we cannot do step 1 for PDA and CFLs are not closed under complement.

Deterministic PDAs

A PDA *M* is ***deterministic*** iff:

- Δ_M contains no pairs of transitions that compete with each other, and
- Whenever *M* is in an accepting configuration it has no available moves.



M can choose between accepting and taking the ε-transition, so it is not deterministic.

Deterministic CFLs

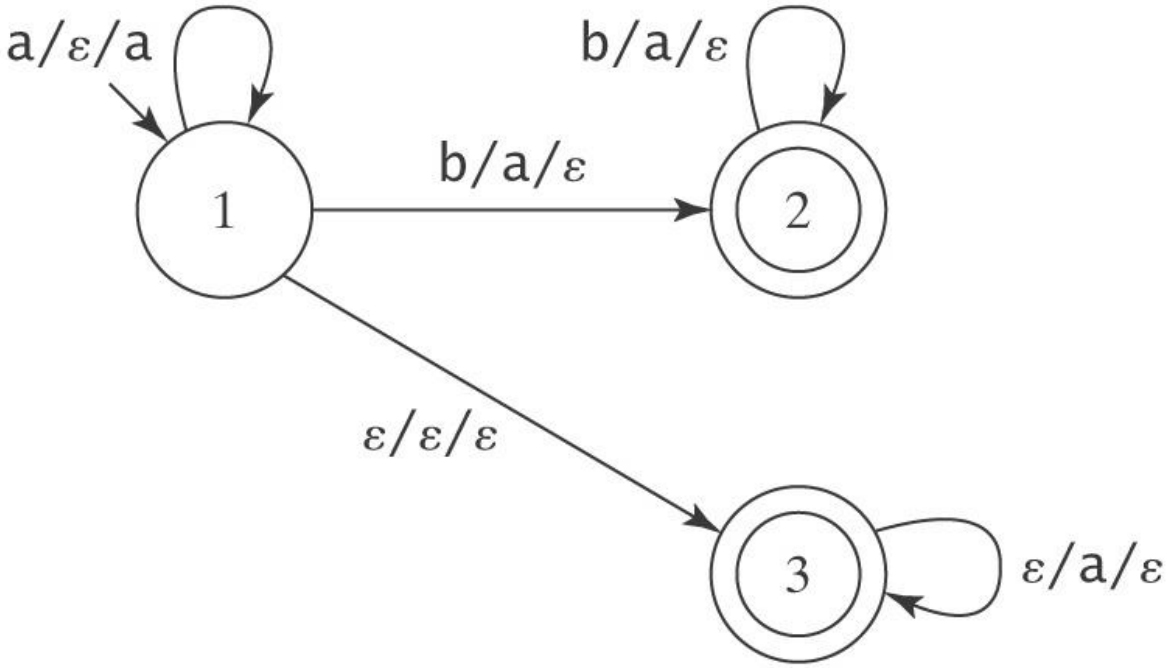
A language L is ***deterministic context-free*** iff $L\$$ can be accepted by some deterministic PDA.

Why \$?

Let $L = a^* \cup \{a^n b^n : n > 0\}$.

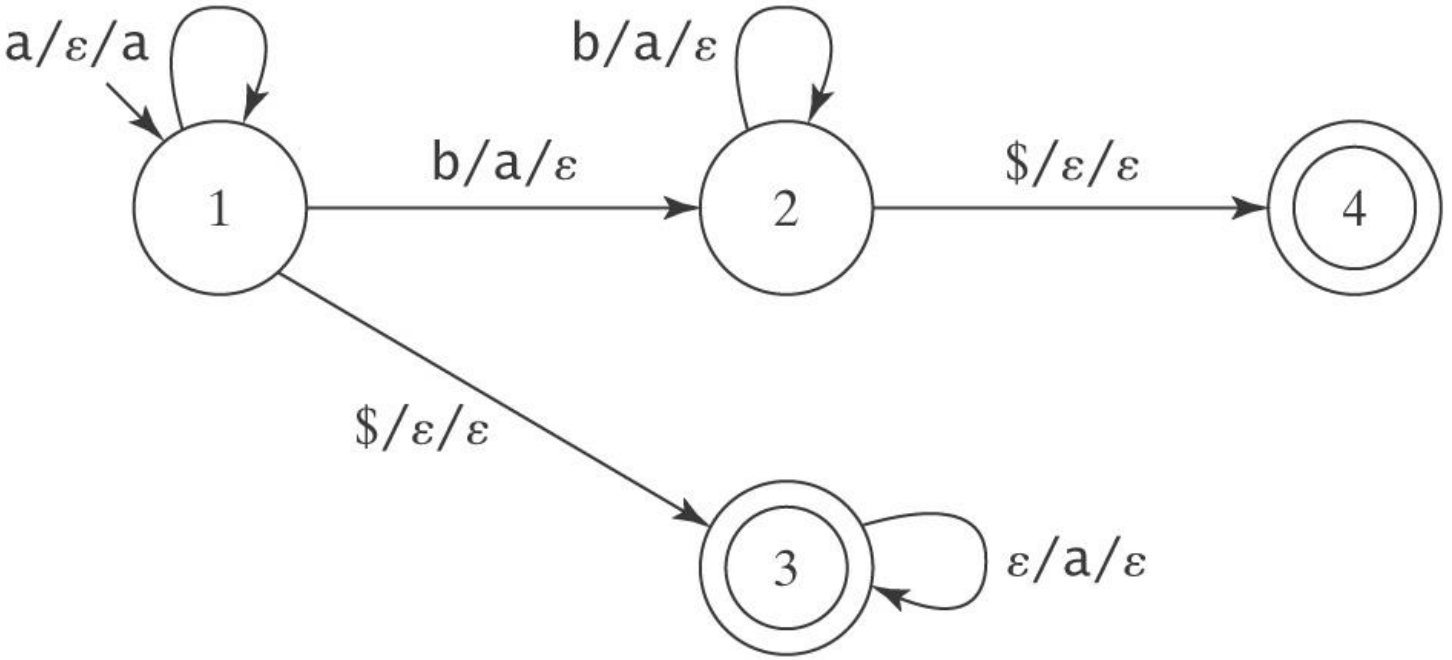
An NDPDA for *L*

$L = a^* \cup \{a^n b^n : n > 0\}$.



A DPDA for *L*$

$L = a^* \cup \{a^n b^n : n > 0\}$.



Adding $ Doesn't Add Power

- Adding '$' makes it easier to build deterministic PDA
- It does not make it possible to build a PDA for a language L that was not already context free.

Closure properties of Deterministic CF Languages

- Closed under complement
- Not closed under union
- Not closed under intersection

Nondeterministic CFLs

Theorem: There exist CFLs that are not deterministic.

Proof: By example. Let $L = \{a^i b^j c^k, i \neq j \text{ or } j \neq k\}$. L is CF. If L is DCF then so is:

$L' = \neg L$.
$= \{a^i b^j c^k, i, j, k \geq 0 \text{ and } i = j = k\} \cup$
$\{w \in \{a, b, c\}^* : \text{the letters are out of order}\}$.

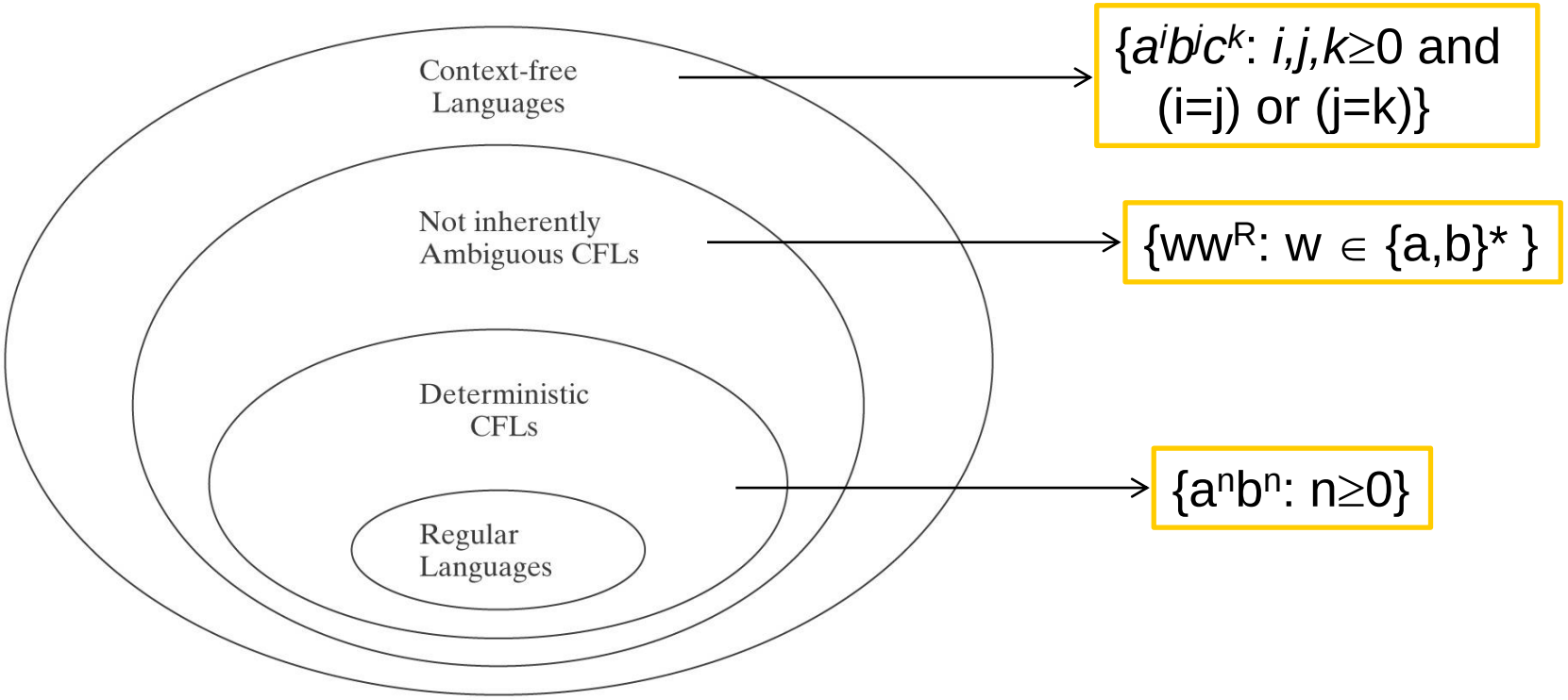
But then so is:

$L'' = L' \cap a^*b^*c^*$.
$= \{a^n b^n c^n, n \geq 0\}$.

But it isn't. So L is context-free but not deterministic context-free.

This simple fact poses a real problem for the designers of efficient context-free parsers.

The CFL Hierarchy



References

- **Automata, Computability and Complexity. Theory and Applications**
  - By Elaine Rich
- Chapter 13:
  - Page : 279-301.