Unit 5 : Context free languages

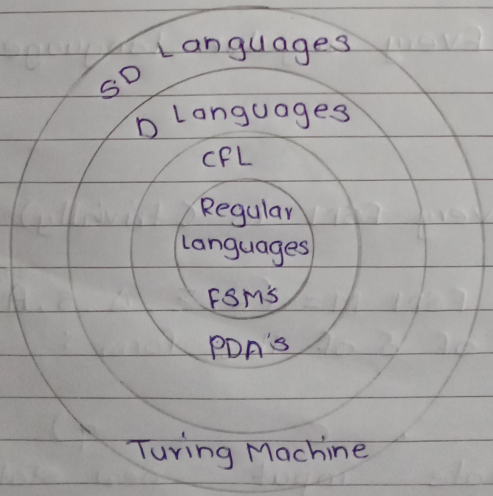
* Languages that are and are not context free :

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

$A^n B^n C^n = \{a^n b^n c^n : n \geq 0\}$ is not context free.

$a^* b^*$ is regular.

* Languages and Machines :



* The regular and the cf Languages :

Theorem :

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

Proof :

In two parts.

① Every regular language is CF

② There exists at least one language that is CF but not regular.

① The Regular and CF Languages :

Lemma :

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 $M' = (K, \Sigma, \{\emptyset\}, \Delta, s, A)$
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 stack.

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

Lemma :
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 language are proper subset of the context-free languages.

Lucky | Page No.:
Date : / /

* Showing that L is context-free :

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

① Exhibit a context-free grammar for L

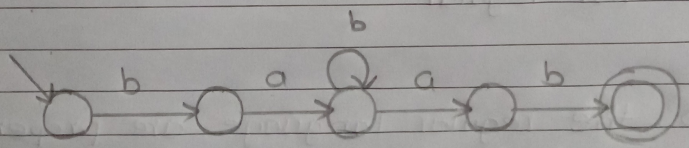
② Exhibit a PDA for L

③ Use the closure properties of context-free languages.

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

* Showing that L is not context free

Remember the pumping argument for regular languages.



Lucky | Page No.:
Date : / /

* A Review of Parse Trees :

A parse tree, derived by a grammar $G = (V, \Sigma, R, S)$ is a rooted, ordered tree in which :

① Every leaf node is labeled with element of $\Sigma \cup \{\varepsilon\}$,

② The root node is labeled S,

③ Every other node is labeled with some element of $V - \Sigma$,

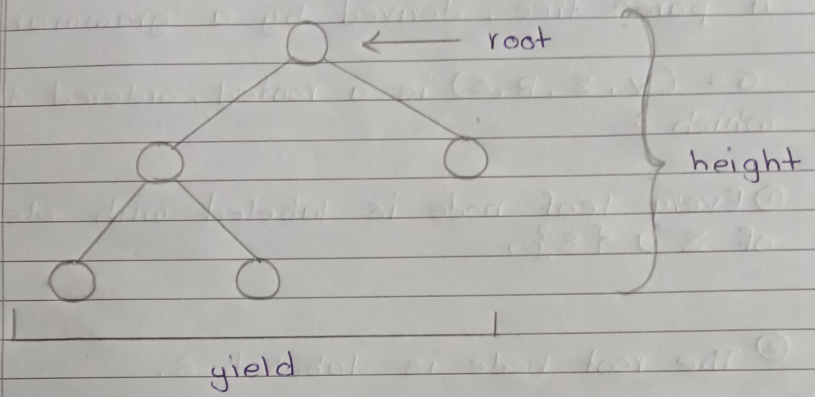
④ If m is nonleaf labeled X and the children of m are labeled $X_1, X_2, \ldots\ldots X_n$ then the rule $X \rightarrow X_1, X_2 \ldots\ldots X_n$ is in R.

* Some tree Basic :

The height of a tree is the length of 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 $\leq b^h$.

* From grammars to Trees :

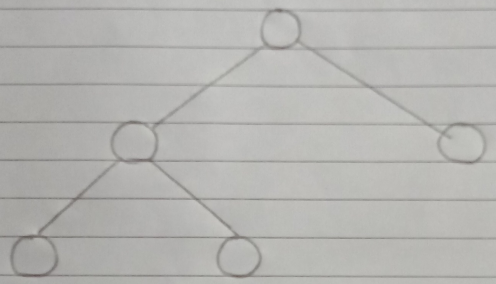
Given a context-free grammar G:

① Let n be the number of nonterminal symbols in G.

② Let b the branching factor of G

Lucky | Page No.:
Date : / /

Suppose that T is generated by G and go nonterminal appears more than once on any path :



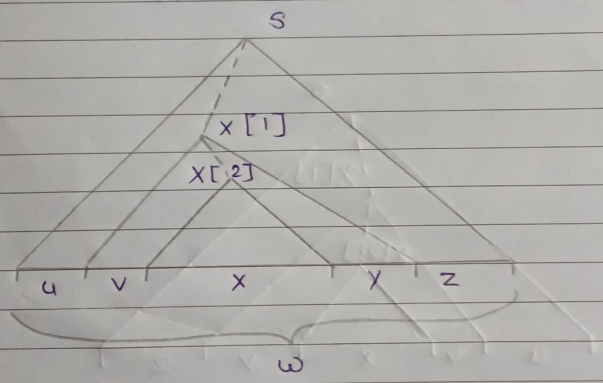
The maximum height of T and maximum length of T is yield depend on tree.

Lucky | Page No.:
Date : / /

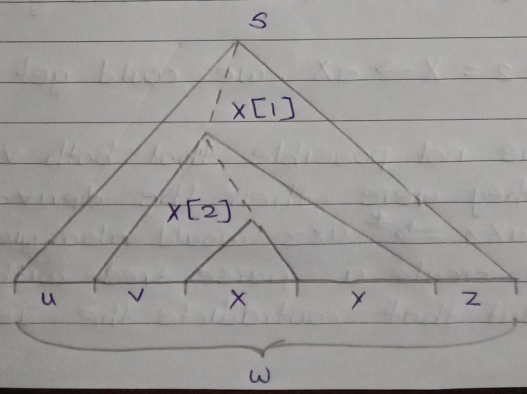
* The context-free Pumping Theorem :

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

If w is "long", then its parse trees must look like :



choose one such tree such that there's no other with fewer nodes.

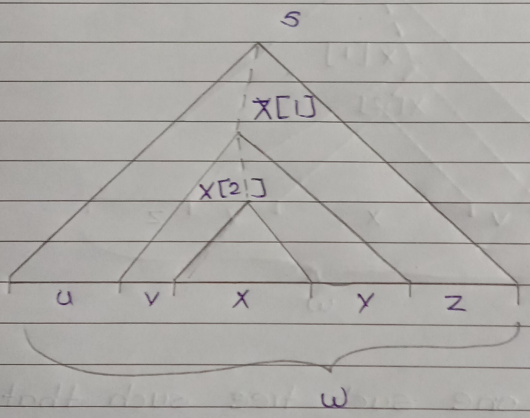


There is another derivation in G.

$S \Rightarrow^* uXz \Rightarrow^* uxz$.

in which at the point labeled [1], the nonrecursive rule 2 is used.

So uxz is also in L(G)

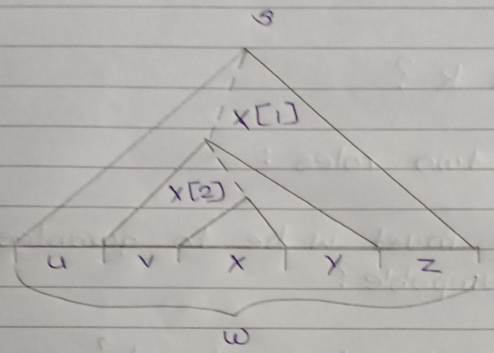


If $rule_1 = X \rightarrow Xa$, we could get $v = \varepsilon$.

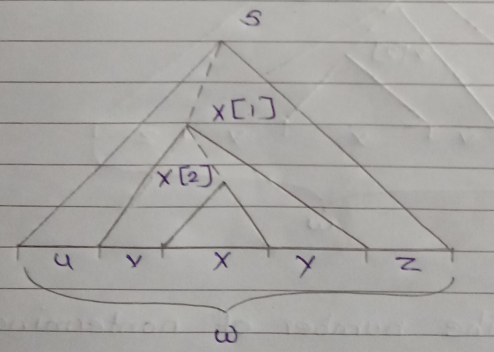
If $rule_2 = X \rightarrow aX$, we could get $y = \varepsilon$

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 contrddicts the assumption

that we started with a tree with the smallest possible number of nodes.



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



If L is a context-free language, then

$\exists k \geq 1$ ($\forall$ strings $w \in L$, where $|w| \geq k$
($\exists u, v, x, y, z$ ($w = uvxyz$,

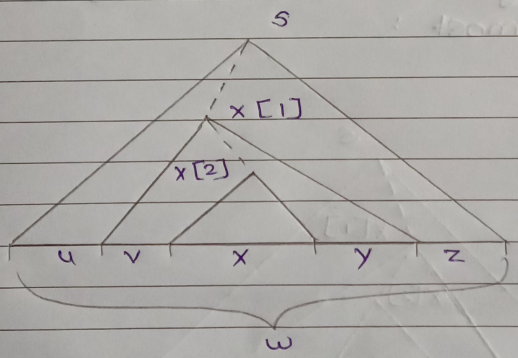
$|xy| \neq \varepsilon$,
$|vxy| \leq K$ and
$\forall q \geq 0$ ($uv^q xy^q z$ in is L))))).

* What is K?

K serves two roles:

① How long must w be to gurantee it is possible: pumpable?

② What's the bound on $|vxy|$?

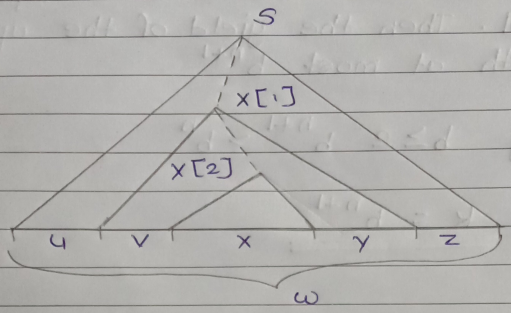


Let n be the number of nonterminals in G

Let n be the branching factor of G

Lucky | Page No.:
Date : / /

* How Long Must W be ?

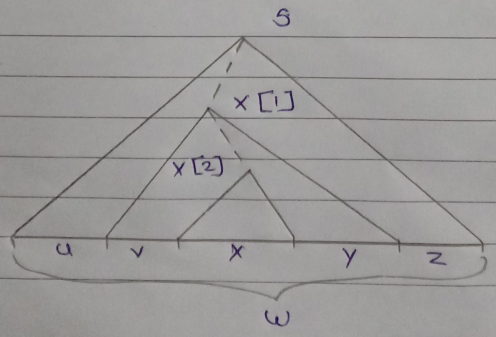


If height (T) > n, then some nonterminal occurs more than one on same path. so T is pumpable

If height (T) $\leq n$, then $|uvxyz| \leq b^n$

so if $|uvxyz| > b^n$, $w = uvxyz$ must be pumpable.

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



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

assuming $b \geq 2$ $b^{n+1} > b^n$

so let $k = b^{n+1}$

Lucky | Page No.:
Date : / /

* 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.

Proof :

L is generated by some CFG, $G = (V, \Sigma, R, S)$ with n nonterminals 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 case that $b^{n+1} > b^n$. So let w be any string in L(G) where $|w| \geq k$. Let T be the 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 smaller parse tree for w and we choose

If so that wasn't so. uxz must be in L because rule 2 could have been used on immediately at $[1]$. for any $q \geq 1$. $uy^q xy^q z$ must be in L because rule 1 could have been used q times before finally using rule 2.

Lucky | Page No.:
Date : / /

* Regular vs CF Pumping theorems :

similarities :

① We choose w, the string to be pumped

② We choose a value for q that shows that w isn't pumpable.

③ We may apply closure theorems before we start.

Differences :

① Two regions, v and y must be pumped in tandem.

② 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. $|vxy| \leq k$

③ Either v or y could be empty, although not both.

Rajdhani
DATE / /

* Example of Pumping :

① $A^n B^n C^n = \{ a^n b^n c^n , n \geq 0 \}$

choose $|w| = a^k b^k c^k$
1 | 2 | 3

- If either v or y spans regions, than let $q = 2$ (i.e., pump in once)

The resulting string will be have letters out of order and thus not be in $A^n B^n C^n$.

- 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 $A^n B^n C^n$

② $L = \{ a^n b^m a^n , n, m \geq 0 \text{ and } n \geq m \}$

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

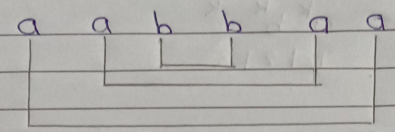
aaa ---- aaabbb ----- bbbaaa ---- aaa
1 1 1 2 1 3 1

Rajdhani

DATE / /

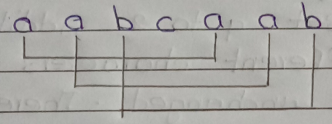
* Nested and cross-serial Dependencies :

$PalEven = \{ WW^R : W \in \{a, b\}^* \}$



The dependencies are nested

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



cross-serial dependencies

①

Rajdhani
DATE / /

① $WcW = \{WcW : W \in \{a,b\}^*\}$

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

aaa... aaabbb... bbbcaaa... aaabbb... bbb
1 1 1 2 |3| 4 1 5 1

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

① If v or y overlaps region 3, set q to 0. The resulting string will no longer contain a c.

② 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 other.

③ 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 $|vxy| \leq k$

④ 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 $|vxy| \leq k$.

Rajdhani
DATE / /

* closure Theorems for context-free languages.

The context free languages are closed under:

① union

② concatenation

③ kleene star

④ Reverse

⑤ Letter substitution

① 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.

Rajdhani
DATE / /

②

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 by exhibiting a CFG for it:

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

② 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 $L = L(G_1) \cdot L(G_2)$

Rajdhani
DATE / /

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

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

③ closure under kleene star

Let $G = (V, \Sigma, R, S)$

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.

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

④ 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.

Rajdhani
DATE / /

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

① $X \to a : L(X) = \{a\}$. $\{a\}^R = \{a\}$

② $X \to BC : L(X) = L(B)L(C)$ $(L(B)L(C)^R) = L(C)^R L(B)^R$

construct, from G, a new grammar G_1 such that $L(G) = L^R$:

$G = (V_G, \Sigma_G, R, S_G)$, where R is constructed as follows :

① for every rule in G of the form $X \to BC$, add to R the rule $X \to CB$

② for every rule in G of the form $X \to a$ add to R the rule $X \to a$.

Rajdhani
DATE / /

* 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 proved closure for regular ~~under either complement~~ 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 automata for L_1 and L_2 construct an automata for $L_1 \cap L_2$ by simulating the parallel operation of the two original machine, using states that are the cartesian product of the sets of states of the two original machines.

⑤ closure under Intersection

The context-free languages are not closed under intersection:

Rajdhani
DATE / /

The proof is by counterexample. Let:

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

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

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

But now consider:

$L = L_1 \cap L_2$

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

(6) closure under complement

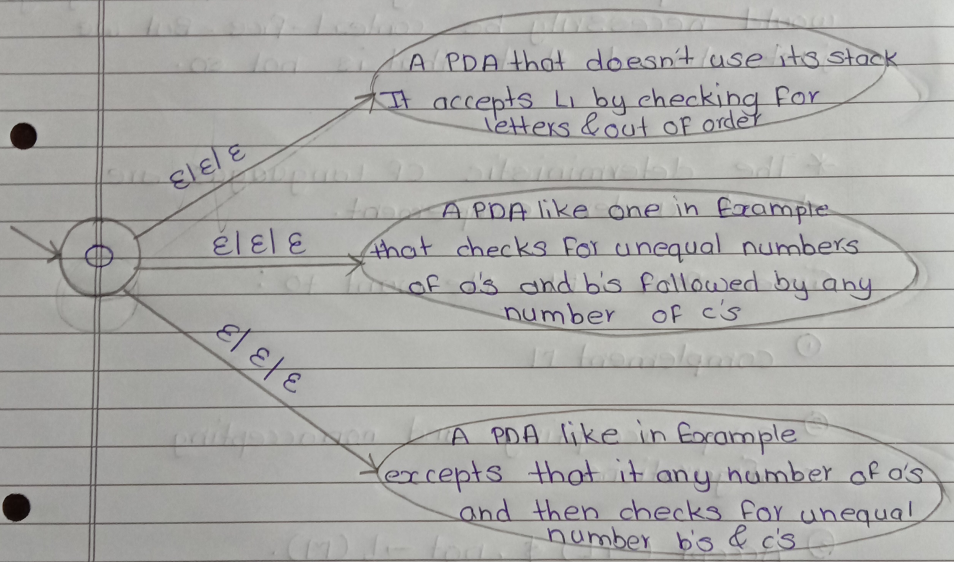
$L_1 \cap L_2 = \neg(\neg L_1 \cup \neg L_2)$

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).

Rajdhani
DATE / /

* closure under complement an Example

$\neg A^n B^n C^n$ is context-free



But $\neg(\neg A^n B^n C^n) = A^n B^n C^n$ is not context free

⑦ closure under difference

Are the context-free languages closed under difference ?

Rajdhani
DATE / /

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

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

* The deterministic CF languages are closed under complement.

Given PDA, M. We want to:

① complement M

② swap accepting and nonaccepting configurations.

③ Accept $(\neg L)\$$, not $\neg L(M)$.

A deterministic PDA may fail to accept an input string w because:

① Its computation ends before it finishes reading w.

② Its computation ends in an accepting state but the stack is not empty.

Rajdhani
DATE / /

③ Its computation loops forever, following ε-transition, without ever halting in either an accepting or a nonaccepting state.

④ Its computation ends in a nonaccepting state

If we simply swap accepting and nonaccepting states we will correctly fail to accept every string that M would have accepted (i.e., every string in L$). But we will not necessarily accept every string in (¬L)$.

* DCFLs under Intersection and Union.

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

The DCFLs ~~(regular)~~ are closed under complement.

* DCFLs are Not closed under Union

$L_1 = \{ a^i b^j c^k , i,j,k \geq 0 \text{ and } i \neq j \}$ (a DCFL)

$L_2 = \{ a^i b^j c^k , i,j,k \geq 0 \text{ and } j \neq k \}$ (a DCFL)

Rajdhani
DATE / /

$L' = L_1 \cap L_2$

$= \{a^i b^j c^k, i,j,k \geq 0 \text{ and } (i \neq j) \text{ or } (j \neq k)\}$

$L'' = \neg L'$.

$= \{a^i b^j c^k, i,j,k \geq 0 \text{ and } i = j = k\} \cup$

$\{w \in \{a,b,c\}^* : \text{the letters are out of order}\}$

$L''' = L'' \cap a^* b^* c^*$.

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

L''' is not even cf, much less dcf

Rajdhani
DATE / /

* DCFLs are Not closed under Intersection

$L_1 = \{ a^i b^j c^k , i,j,k \geq 0 \text{ and } i=j \}$

$L_2 = \{ a^i b^j c^k , i,j,k \geq 0 \text{ and } j=k \}$

$L' = L_1 \cap L_2$

=

L_1 and L_2 are deterministic context-free