Deterministic PDAs & Deterministic CFLs

Deterministic pda normally accept by final state. The reason is that for deterministic pdas, accepting by final state is "more powerful" than accepting by null stack.

Lemma A language L has the prefix property if no word in L is a proper prefix of another word in L. If $L = N(M)$ for some dpda M, then L has the prefix property

"Proof" Once the d.p.d.a accepts by emptying the stack (deterministically) it cannot accept any extension string.

Lemma $L = N(M)$ for dpda $M \Leftrightarrow L = L(M')$ for some dpda M' (accepting by final state) and L has the prefix property.

"Proof" $\Rightarrow$ By previous Lemma, L has the prefix property. Can always also find an equivalent d.p.da that accepts by final state. (How?)

$\Leftarrow$ Once you go to the final state, simply empty the stack.

Lemma There exist languages accepted by dpda that do not have the prefix property. Example: ab^*

Def: L is a deterministic c.f.l. $\Leftrightarrow \exists$ dpda M s.t. $L = L(M)$

Note: $L_1 = \{a^n b^{2n} : n \geq 0\}$ & $L_2 = \{a^n b^n : n \geq 0\}$ are dcfls. $L_1 \cup L_2$ is c.f.l. but not dcfl.

Properties of Context-free languages

The pumping lemma for context-free languages.
Let L be a c.f.l. Then there exists a constant m such that if $w \in L$, with $|w| \geq m$, then

$$w = uvxyz$$

with (a) $|vxy| \leq m$
(b) $|vy| \geq 1$
(c) for all $i \geq 0$ $uv^ixy^iz \in L$.

Example

$L = \{a^i b^i c^i \mid i \geq 1\}$ is not context free.

Proof

Suppose L is c.f.l. By lemma, choose m, Let $w = a^m b^m c^m$.
The lemma says w can be written as $uvxyz$ with $|vy| \geq 1$ and $|vxy| \leq m$ and $uv^ixy^iz \in L$ for $i \geq 0$.
Now, vxy cannot include a's, b's, and c's since $|vxy| \leq m$.
Suppose it includes only a's. $\therefore$ at least one a in vy since $|vy| \geq 1$. $\therefore$ uxz has m b's, m c's, but less than m a's. Contradiction since $uxz \in L$.
Similar arguments if vxy has only b's or c's.
Suppose vxy has both a's & b's. Then uxz has m c's but less than m a's or b's. Again a contradiction.
Similar argument if vxy has b's and c's.

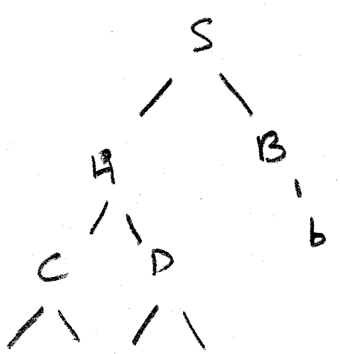
<u>Example</u>

$L = \{ a^i b^i c^j \mid j \geq i \}$ is not context-free.

Similar approach as previous example.
Consider m as in Lemma, and $w = a^m b^m c^m$
Can write this as $uvxyz$. Suppose only a's in vxy.
$\therefore$ uxz has m b's but less than m a's. Contradiction
Similarly for only b's. Suppose only c's. Then same argument
Etc ...

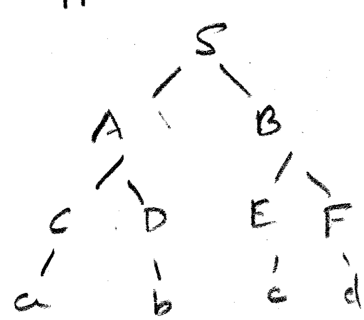
<u>Proof of Pumping Lemma</u>

Let G be a Chomsky normal form grammar for L. ($L \not\ni \lambda$).
Consider a derivation for w. Must be like:



Etc.

Suppose $k = 3$. k the depth of the tree

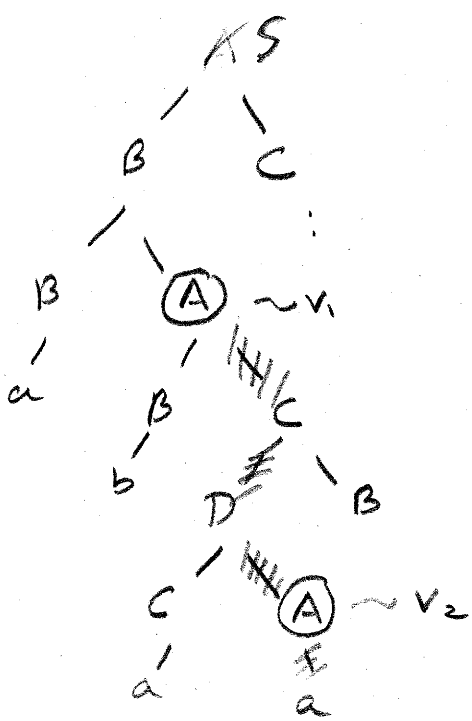


If $k = 3$
longest word is
2^{k-1}.

$\therefore$ if longest path is k, the longest derived word is 2^{k-1}.

Now, let G have k variables, and choose $m = 2^k$
and $|w| \geq m$.
Therefore, parse tree for w must have a path of length at least $k+1$. (If all paths of max length k, then all words of length max 2^{k-1}).

Since path is of length $\geq k+1$, vertices on path are $\geq k+2$
$\therefore$ some non leaf vertex must repeat. (variable must repeat)



Longest path p, with $|p| \geq k+1$.
$\therefore$ let v_2 be closest vertex to leaf that repeats and let v_1 be next closest.
Let T_1 be the subtree from vertex v_1 and
T_2 be the subtree from vertex v_2

Note that the longest path from v_1 is of length at most $k+1$. (By construction)

$\therefore$ If T_1 yields z_1 then $|z_1| \leq 2^k = m$
Let T_2 yield z_2.
$\therefore$ $z_1 = z_3 z_2 z_4$ with $|z_3 z_4| \geq 1$
(Chomsky NF & $A \Rightarrow A$ not allowed)

$\therefore$ $\left.\begin{array}{l} A \overset{*}{\Rightarrow} z_3 A z_4 \\ A \overset{*}{\Rightarrow} z_2 \end{array}\right\} \Rightarrow \quad A \overset{*}{\Rightarrow} z_3^i z_2 z_4^i$

$\therefore$ w can be written as $z_5 \underbrace{z_3 z_2 z_4}_{z_1} z_6$

and lemma follows.

Closure Properties for C.F.L.s

① C.F.L.s are closed under union, concatenation, Kleene star.

Proof

$\cup$ Let $G_1 = (V_1, T_1, S_1, P_1)$ and $G_2 = (V_2, T_2, S_2, P_2)$ that generate L_1 and L_2 respectively.
Let $G = (V_1 \cup V_2 \cup \{S\}, T_1 \cup T_2, S, P_1 \cup P_2 \cup P_3)$ with P_3: $S \rightarrow S_1 \mid S_2$
It can easily be established that $L(G) = L_1 \cup L_2$.

$\cdot$ Same as union but with P_3: $S \rightarrow S_1 \cdot S_2$

Kleene * Let $G_1 = (V_1, T_1, S_1, P_1)$ generating L_1.
Let $G = (V_1 \cup \{S\}, T_1, S, P_1 \cup P)$ where:
P: $S \rightarrow SS_1 \mid \lambda$.
It can easily be shown that $L(G) = L_1^*$.

② C.F.L.s are closed under substitution & inverse homomorphism.
(Proof not shown).

③ C.F.Ls are <u>not</u> closed under intersection or complement.
$L_1 = \{ a^i b^i c^j \mid i,j \geq 1\}$ and $L_2 = \{ a^i b^j c^j \mid i,j \geq 1\}$
can easily be shown to be C.F.L.s. (give P.D.A. or grammar).
But $L_1 \cap L_2 = \{a^i b^i c^i \mid i \geq 1\}$ is not a C.F.L. (why?)
$\therefore$ C.F.L.s not closed under intersection.

Note: ④ If L is a C.F.L. & R is a regular language, then $L \cap R$ is a c.f.l.

Note: D.C.F.L.s are <u>not</u> closed under union or intersection.
D.C.F.L.s are closed under complement.

Example

Let $L = \{ ww \mid w \in (a+b)^* \}$.

L is not a c.f.l.

Proof

Suppose L is a c.f.l. Consider $R = a^+b^+a^+b^+$

R is a regular language

$L \cap R = \{ a^i b^j a^i b^j \mid i,j \geq 1 \}$

By pumping lemma we can prove that $L \cap R$ is not c.f.l.

$\therefore$ L is not a c.f.l.

Example

$L = \{ a^n b^n : n \geq 0, n \neq 100 \}$ is a c.f.l

Let $L_1 = \{ a^n b^n : n \geq 0 \}$ L_1 is easily c.f.l.

Let $R_1 = a^* b^*$ R_1 is regular.

Let $R_2 = \{ a^{100} b^{100} \}$ R_2 is regular.

$R = R_1 - R_2$ is regular

$\therefore$ $L_1 \cap R$ is c.f.l.

$L = L_1 \cap R$. Proof complete

Decision Algorithms for C.F.L.s

① There exists an algorithm to decide if a C.F.L. is empty.
Proof: use the fact that we can determine if S is useless.

② There is an algorithm to determine if a C.F.L. is finite.
③ There is an algorithm to determine if a C.F.L. is infinite.
Proof: (2 & 3).
First eliminate λ - productions, useless productions, and unit productions. Then check remaining productions to determine if there is some nonterminal A s.t.

$A \overset{*}{\Rightarrow} xAy$ (a repeating variable).

(Note that this can be done using a dependency graph.)
If there exists a repeating variable, the language is infinite; else it is finite.

④ There exists an algorithm to determine if $w \in L(G)$ for a c.f.g. G. (membership)
Proof: Represent grammar in Chomsky Normal Form. If $|w| = n$, then there is an n^3 algorithm to determine if $w \in L(G)$.
(The algorithm is the CYK algorithm in Chapter 6.)

<u>Examples</u>

Show that $L=\{a^{n^2} \mid n \ge 1\}$ is not context free.

choose m so pumping lemma applies and consider a^{m^2}. This can be written as

$$a^{m^2} = uvxyz \qquad \text{with } |vxy| \le m, \quad |vy| \ge 1$$

and $uv^2xy^2z \in L$.

But $m^2+m \ge |uv^2xy^2z| > m^2$

$\therefore (m+1)^2 > |uv^2xy^2z| > m^2$. Contradiction!

Show that $L = \{ wcw \mid w \in \{a,b\}^* \}$ is not context-free.

Choose m as in lemma consider

$$a^m b^m c a^m b^m = uvxyz$$

if $vxy \in$ a's only or b's only then uxz gives a contradiction.

if vxy spans first a's & b's then again uxz gives a contradiction.

If vxy span some of first b's and some of second a's

$$\underbrace{b\cdots b}_{m}\ c\ \underbrace{a\cdots a}_{m}$$

$v\ x\ y$

assume $|v| \ge 1$. Then by pumping, there are more b's than in "second set".

similarly if $|y| \ge 1$. (More a's than in first set)

Context sensitive & general grammars

Find a grammar than generates $\{a^n b^n c^n \mid n \geq 1\}$ which has been proven not to be context-free.

$S \to aSBC \mid aBC$
$CB \to BC$
$aB \to ab$
$bB \to bb$
$bC \to bc$
$cC \to cc$

(This is a context-sensitive grammar)
(It generates the above language)

Find a grammar that generates $\{ww \mid w \in \{0,1\}^*\}$
Again, this has been shown not to be a C.F.L.
Consider the following grammar:

$S \to ABC$	"start"	
$AB \to 0AD \mid 1AE$	"add 0 or 1 Left & use D,E to mark second 0,1	
$DC \to B0C$	"add 0 right"	
$EC \to B1C$	"add 1 right"	**Example**
$D0 \to 0D$	move D right	ABC
$D1 \to 1D$	move D right	$0ADC$
$E0 \to 0E$	move E right	$0AB0C$
$E1 \to 1E$	move E right	$01AE0C$
$0B \to B0$	move B left	$01A0EC$
$1B \to B1$	move B left	$01A0B1C$
$AB \to \lambda$	eliminate AB	$01AB01C$
$C \to \lambda$	eliminate C	use AB & continue or end:

$01\lambda 01\lambda$
0101