Normal Forms (Chomsky, Greibach) for CFGs

(A) Eliminate useless symbols

Definition

A symbol $X \in V \cup T$ *is called :*

- ***generating*** *if* $X \Rightarrow^* z$ *for some* $z \in T^*$
- ***reachable*** *if* $S \Rightarrow^* \alpha X \beta$ *for some* $\alpha, \beta \in (V \cup T)^*$
- ***useful*** *if it is both generating and reachable*

Example

$S \to A\,B \mid a \;;\; B \to b \;;\; C \to cD \mid b$

A *is non-generating ;* C *is non-reachable ;*

D *is both non-reachable and non-generating*

Algorithm for eliminating useless symbols

Given a CFG G = (V,T, R, S)

(1) Eliminate all ***non-generating*** *symbols to end up in the CFG* **:** $\boldsymbol{G1 = (V_1, T_1, R_1, S)}$.

Do this by the following inductive method :

Basis **:** *Elements of* ***T*** *are* ***generating*** *by definition of zero step derivation.*

Induction **:** *If for a production* $A \to \alpha$ *all the elements of* α *are* ***generating*** *or* $\alpha = e$ *then* ***A*** *is generating.*

If ***X*** *is non-generating then remove all productions of the form* $X \to \alpha$ *and* $C \to \alpha X \beta$

(2) Eliminate all ***non-reachable*** *symbols to end up in the CFG* **:** $\boldsymbol{G2 = (V_2, T_2, R_2, S)}$.

Basis **:** ***S*** *is* ***reachable*** *by definition.*

Induction **:** *If within a production* $A \to \alpha$, ***A*** *is* ***reachable*** *then all the elements of* α *are* ***reachable***

If ***X*** *is non-reachable then remove all productions of the form* $X \to \alpha$

Fact *: After* ***first*** *removing all productions involving* ***nongenerating*** *variables on its LHS or RHS and* ***then*** *removing productions involving* ***unreachable*** *symbols (terminals and nonterminals) all remaining symbols are useful ; i.e. both* ***reachable*** *and* ***generating*** *!*

Consider the productions

$\boldsymbol{S \rightarrow A\,B \mid a}$

$\boldsymbol{B \rightarrow b}$

*then all **A,B , a** and **b** are **reachable.***

*But at the next step of generability **A** is **non-generating ,** hence the new grammar has the productions :*

$\boldsymbol{S \rightarrow a}$

$\boldsymbol{B \rightarrow b}$

*But then **B** is **non-reachable** which is missed out in the first step.*

*Hence the **correct** algorithmic method is : (1) Eliminate **non-generating** symbols and productions first and (2) Eliminate the **non-reachable** symbols out of the remaining symbols and productions.*

*Applied to the example above first eliminate the **non-generating** variable **A** and the associated production* $\boldsymbol{S \rightarrow A\,B}$ *and then eliminate the **non-reachable** symbol **B** and the associated production* $\boldsymbol{B \rightarrow b}$

Theorem

*The CFG **G2** generated by the algorithm above has the property :*

(1)** Every non- terminal and terminal variable of **G2** is **useful** in **G ,

*i.e. it is both **generating** and **reachable** in **G***

(2) $L_G = L_{G2}$

Proof** Exercise **: Prove (1)

*We prove **(2)** in two steps : **(i)** $L_G \subseteq L_{G2}$ and **(ii)** $L_{G2} \subseteq L_G$*

(i) $w \in L_G$ *and* $S \Rightarrow_G \ldots \Rightarrow_G \alpha_j \Rightarrow_G \ldots w$, *be a derivation of **w** in **G** then*

$S \Rightarrow_{G2} \ldots \Rightarrow_{G2} \alpha_j \Rightarrow_{G2} \ldots w$ *,since each* α_j *consists only of useful terms*

by definition

*(ii) is trivially true since **G2** is a sub-grammar of **G***

(B) Eliminate e (epsilon) productions : $A \to e$

Definition

A is called ***nullable*** *if* $A \Rightarrow^* e$

Compute all nullable variables inductively

Basis : *A is nullable if* $A \to e$;

Induction : *If* $B \to C_1C_2 \ldots C_n$ *and each* C_i *is nullable*

*then **B** is nullable*

Algorithm to eliminate e-productions

Construct a new grammar **G'= (V,T,R',S)** *from* **G = (V,T,R,S)**

*Productions in **R** are of the form:* $A \to X_1X_2\ldots X_m$ *where* $k \le m$ *of the* X_j

*variables (which are necessarily non-terminal) are **nullable***

Include in **R’**, **2**^{k} *productions where each nullable* $\boldsymbol{X_j}$ *is present or absent; (except when* **m=k** *avoid the* $\boldsymbol{A \to e}$ *case that corresponds to absence of all terms) also* ***remove*** *all productions of the form* $\boldsymbol{A \to e}$

Theorem $L_{G'} = L_G - \{e\}$

Proof : For any production used in a derivation use the version where the eventually nullified variables are absent ! Hence

$$\ldots \Rightarrow_G \mu X \nu \Rightarrow_G \ldots \Rightarrow_G \alpha X \beta \underset{X \to e}{\Rightarrow_G} \alpha \beta$$

to be replaced by

$$\ldots \Rightarrow_{G'} \mu' \cancel{X} \nu' \Rightarrow_{G'} \ldots \Rightarrow_{G'} \alpha' \cancel{X} \beta'$$

production where ***X*** *is absent is used*

Example for epsilon productions

R :

$S \rightarrow Sa \mid AB \mid e$; $A \rightarrow BCbDa \mid cd$; $B \rightarrow Db \mid e$; $D \rightarrow BC \mid d$; $C \rightarrow aC \mid e$

S, B *and* ***C*** *are* ***nullable*** *because of their* ***e*** *productions !*

D *is* ***nullable*** *because of* $D \rightarrow BC$ *where* ***B*** *and* ***C*** *are* ***nullable***.

$S \rightarrow Sa \mid a \mid AB \mid A$;

111 110 101 011 100 010 001 000

$A \rightarrow BCbDa \mid BCba \mid BbDa \mid CbDa \mid Bba \mid Cba \mid bDa \mid ba \mid cd$;

$B \rightarrow Db \mid b$; $D \rightarrow BC \mid B \mid C \mid d$; $C \rightarrow aC \mid a$

(C) Eliminating unit productions : $A \to B$, $B \in V$

Definition

*A production of the form **$A \to B$** is called a **unit** production*

*Call **(A,B)** with **A,B** $\in$ **V** a **unit pair** if **$A \Rightarrow^* B$** where only **unit productions** are used in the derivation*

- Algorithm** to determine **unit pairs

*Construct a **digraph D** where variables are the nodes and there is a directed edge from **A** to **B** iff there is a unit production **$A \to B$**.*

*Then **(A,B)** is a **unit pair** iff there is a path from **A** to **B** in **D**.*

- ***Algorithm*** *for computing unit production-free* $\mathbf{G' = (V,T,R',S)}$ *from* **G**

(1) Compute all ***unit pairs*** *of* **G**

(2) Include all ***non-unit productions*** *of* **R** *in* **R'** *and in addition*

for each unit pair **(A,B)** *add to* **R'** *the production* $A \to \alpha$ *if* $B \to \alpha$ *is a non-unit production in* **R**

Theorem $L_{G'} = L_G$

Chomsky Normal Form (CNF)

2 kinds of productions are allowed and there are no useless symbols :

(1) $A \rightarrow BC$, $B,C \in V$

(2) $A \rightarrow a$, $a \in T$

Algorithm for computing the CNF

*(i) eliminate **(a)** epsilon productions ;**(b)** unit productions ;**(c)** useless symbols (first nongenerating then nonreachable)*

(ii)For every production of the form $W \rightarrow X_1 X_2 \ldots X_n$, *if* $X_i \in T$ *then replace* X_i *with a new variable* Λ_i *in this production and add the new production* $\Lambda_i \rightarrow X_i$

(iii) Replace every production of the type $A \rightarrow B_1 B_2 \ldots B_n$ *for* $n \geq 3$ *with the productions :* $A \rightarrow B_1 C_1$, $C_1 \rightarrow B_2 C_2$, … , $C_{n-2} \rightarrow B_{n-1} B_n$ *where* C_i , $i = 1, \ldots, n-2$ *are new variables.*

Example (Chomsky Normal Form) (Start symbol is E)

$E \rightarrow T \mid E+T$

$T \rightarrow F \mid T*F$

$F \rightarrow I \mid (E)$

$I \rightarrow 0 \mid 1J \mid x0 \mid x1J \mid$

$J \rightarrow 0J \mid 1J \mid e$

Eliminate null production $J \rightarrow e$

$E \rightarrow T \mid E+T$

$T \rightarrow F \mid T*F$

$F \rightarrow I \mid (E)$

$I \rightarrow 0 \mid 1 \mid 1J \mid x0 \mid x1 \mid x1J$

$J \rightarrow 0J \mid 1J \mid 0 \mid 1$

$zero \rightarrow 0 \quad X \rightarrow x \quad [\ \rightarrow (\quad mult \rightarrow *$

$one \rightarrow 1 \quad]\ \rightarrow)\quad add \rightarrow +$

Eliminate unit pairs

$E \rightarrow T \rightarrow F \rightarrow I$

unit pairs (E,T),(E,F),(E,I),(T,F),(T,I),(F,I)

(E,I) (E,F) (E,T)

$E \rightarrow 0 \mid 1 \mid 1J \mid x0 \mid x1 \mid x1J \mid (E) \mid T*F \mid E+T$

$T \rightarrow 0 \mid 1 \mid 1J \mid x0 \mid x1 \mid x1J \mid (E) \mid T*F$

$F \rightarrow 0 \mid 1 \mid 1J \mid x0 \mid x1 \mid x1J \mid (E)$

$I \rightarrow 0 \mid 1 \mid 1J \mid x0 \mid x1 \mid x1J$ → (I is nonreachable)

$J \rightarrow 0J \mid 1J \mid 0 \mid 1$

$E \rightarrow 0 \mid 1 \mid one\ J \mid X\ zero \mid X\ one \mid X\ one\ J \mid [E] \mid T\ mult\ F \mid E\ add\ T$

$T \rightarrow 0 \mid 1 \mid one\ J \mid X\ zero \mid X\ one \mid X\ one\ J \mid [E] \mid T\ mult\ F$

$F \rightarrow 0 \mid 1 \mid one\ J \mid X\ zero \mid X\ one \mid X\ one\ J \mid [E]$

$J \rightarrow zero\ J \mid one\ J \mid 0 \mid 1$

Example (Chomsky Normal Form , continued)

$E \rightarrow 0 \mid 1 \mid one\ J \mid X\ zero \mid X\ one \mid A\ J \mid B\ ] \mid CF \mid DT$

$T \rightarrow 0 \mid 1 \mid one\ J \mid X\ zero \mid X\ one \mid A\ J \mid B\ ] \mid CF$

$F \rightarrow 0 \mid 1 \mid one\ J \mid X\ zero \mid X\ one \mid A\ J \mid B\ ]$

$J \rightarrow zero\ J \mid one\ J \mid 0 \mid 1$

$zero \rightarrow 0$

$one \rightarrow 1$

$X \rightarrow x$

$[\rightarrow ($

$] \rightarrow)$

$mult \rightarrow *$

$add \rightarrow +$

$A \rightarrow X\ one$

$B \rightarrow [\ E$

$C \rightarrow T\ mult$

$D \rightarrow E\ add$

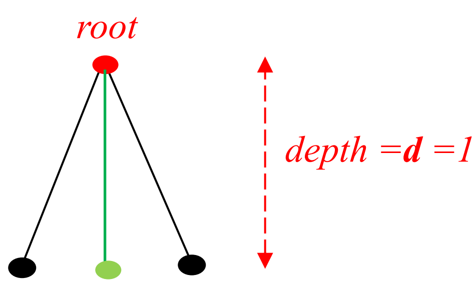
$A \rightarrow X\ one$ $B \rightarrow [E$ $C \rightarrow T\ mult$ $D \rightarrow E\ add$

$E \rightarrow 0 \mid 1 \mid one\ J \mid X\ zero \mid X\ one \mid X\ one\ J \mid [E] \mid T\ mult\ F \mid E\ add\ T$

$T \rightarrow 0 \mid 1 \mid one\ J \mid X\ zero \mid X\ one \mid X\ one\ J \mid [E] \mid T\ mult\ F$

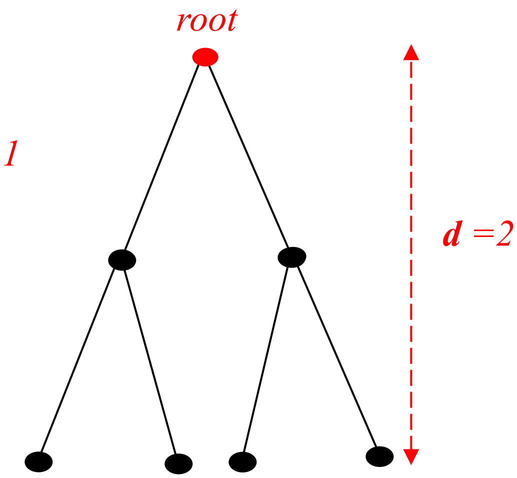
$F \rightarrow 0 \mid 1 \mid one\ J \mid X\ zero \mid X\ one \mid X\ one\ J \mid [E]$

A word on Binary Trees

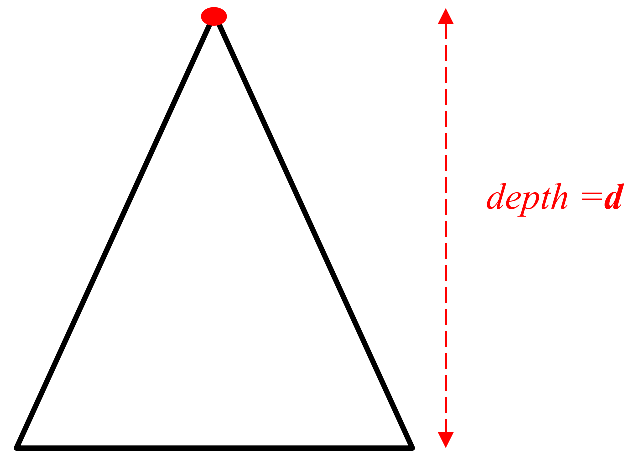


at most ***2*** *chidren*
for a ***full*** *binary tree*

could be 1 if not full



at most ***4*** *leaves for a* ***full*** *binary tree*



at most $\mathbf{2^d}$ *leaves for a* ***full*** *binary tree*

depth of a binary tree :=

the longest distance - *measured as the number of edges -* ***from the root to any of the leaves .***

Note that the ***parse tree*** *of any word generated by CFG in* ***Chomsky Normal Form*** *is a binary tree !*

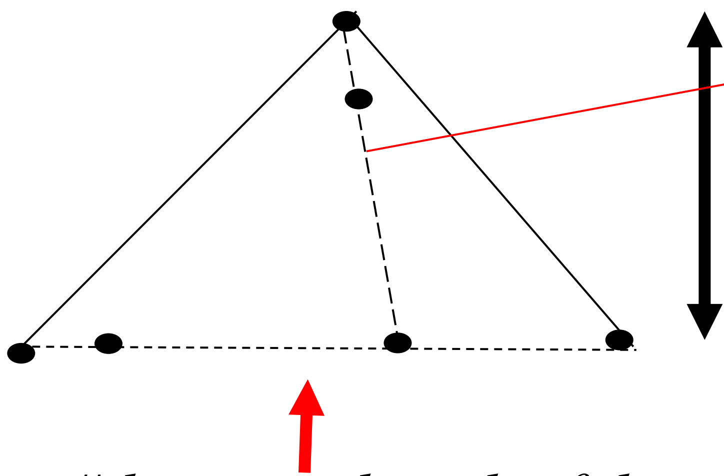
The Pumping Lemma for CFGs

The structure of a ***Parse Tree*** of a *CFG* (= *binary* tree if in *CNF*)

Let $\boldsymbol{m := |V|}$ *and choose a word* z *of length* $|z| = k \geq 2^{m+1}$

then if $\boldsymbol{d}$ *is the depth of the parse tree for* z *then* $2^{m+1} \leq |z| = k \leq 2^d$

hence $\boldsymbol{m+1 \leq d}$ *; and thus at least one* ***variable*** *in* V *occurs repeated on some longest path !*



longest path length = d $\Rightarrow$
d variable nodes on the path

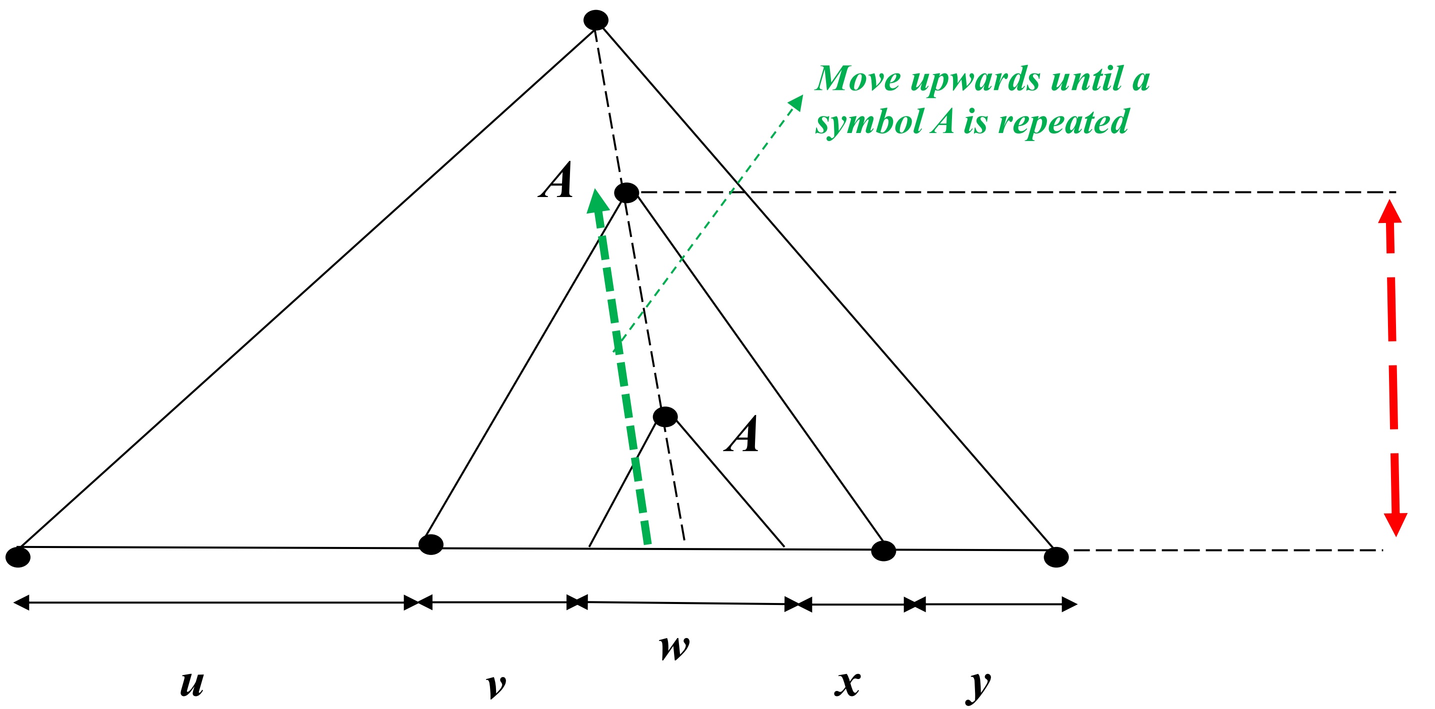
$\boldsymbol{d}$ *=depth of the tree = longest path from the root*

$$\boldsymbol{n := 2^{m+1} \leq k \leq 2^d \Rightarrow m+1 \leq d}$$

$\boldsymbol{k}$ *= # leaves = length of the yield string* z

$z = u\,v\,w\,x\,y$

$|v\,w\,x| \leq 2^{m+1} = n$



$A \Rightarrow^* v\,A\,x$ *and* $A \Rightarrow^* w$ *hence* $A \Rightarrow^* v^i w\,x^i$, $i = 0,1\ldots$

hence $S \Rightarrow^* uAy \Rightarrow^* u\,v^i w\,x^i y$,

$i = 0, 1, \ldots$ *where* $|vwx| \leq n$ *and* $|v\,x| > 0$

Pumping Lemma for CFGs

*Let **L** be a CFL . Then there exists a constant **n** such that for any string*

$z \in L$ with $|z| \geq n$, z can be written as $z = u\,v\,w\,x\,y$ where :

(1) $|v\,w\,x| \leq n$

(2) $|v\,x| > 0$

(3) $u\,v^i\,w\,x^i\,y \in L$ *for all* $i \geq 0$

explanatory remark

$n=2^{|V|+1}$ *where*

CFG is $G=(V,T,R,S)$

Applications of the Pumping Lemma

The following are examples of non-CF languages

1 - $L = \{ a^k b^k c^k \mid k \geq 1 \} \subseteq \{a, b, c\}^*$

2 - $L = \{ a^k b^m c^k d^m \mid k, m \geq 1 \} \subseteq \{a, b, c, d\}^*$

3 - $L = \{ a^p b^r c^s \mid p > r > s \geq 0 \} \subseteq \{a, b, c\}^*$

4 - $L = \{ t\,t \mid t \in \{a, b\}^* \}$

1 – Let $\boldsymbol{n}$ *be as in Pumping Lemma and choose* $z = a^n b^n c^n \in L$. *Then by PL*

$a^n b^n c^n = uvwxy$ *and we show that* $uwy \notin L$, *a contradiction to PL.*

Since by PL $|vwx| \leq n$ *either : (i)* $vwx = a^k$ *or* $= b^k$ *or* $= c^k$ *where* $0 < k \leq n$

or : (ii) $vwx = a^i b^j$ *or* $= b^i c^j$ *where* $0 < i+j \leq n$

moreover again by PL , $p := |vx| > 0$ *, hence :*

If (i) holds then $uwy = a^{n-p} b^n c^n$ *or* $= a^n b^{n-p} c^n$ *or* $= a^n b^n c^{n-p}$

If (ii) holds then $uwy = a^m b^k c^n$ *or* $= a^n b^m c^k$ *where* $m+k = 2n - p < 2n$

for all cases $uwy \notin L$ *and the result follows.*

2 – Let n *be as in Pumping Lemma (PL) and choose* $z = a^n b^n c^n d^n \in L$. *Then by PL* $a^n b^n c^n d^n = uvwxy$ *and we show that* $uwy \notin L$, *a contradiction to PL.*

Since $|vwx| \leq n$, *either* vwx *covers (i) one symbol among* **a,b,c** *and* **d** *or (ii) contains two adjacent symbols*

If (i) holds then $vwx = a^k$ *or* $= b^k$ *or* $= c^k$ *or* $= d^k$ *where* $0 < k \leq n$

If (ii) holds then $vwx = a^i b^j$ *or* $= b^i c^j$ *or* $= c^i d^j$ *where* $0 < i+j \leq n$

moreover by PL , $p := |vx| > 0$ *, hence :*

If (i) holds then $uwy = a^{n-p} b^n c^n d^n$ *or* $= a^n b^{n-p} c^n d^n$ *or* $= a^n b^n c^{n-p} d^n$ *or* $= a^n b^n c^n d^{n-p}$

If (ii) holds then $uwy = a^m b^k c^n d^n$ *or* $= a^n b^m c^k d^n$ *or* $= a^n b^n c^m d^k$ *where* $m,k \leq n$, $m+k = 2n - p < 2n$.

In all cases $uwy \notin L$ *and the result follows.*

3 - *Let* n *be as in Pumping Lemma (PL) and choose* $z = a^{n+2}b^{n+1}c^{n} \in L$. *Then* $|z| = 3n+3 > n$ *and by PL* $a^{n+2}b^{n+1}c^{n} = uvwxy$. *We show depending on cases either* $uwy \notin L$; *or* $uv^2wx^2y \notin L$ *both contradicting PL.*

By PL $|vwx| \leq n$, *hence ; (i)* $vwx = a^k$; *or (ii)* $vwx = b^k$; *or (iii)* $vwx = c^k$

If (i) or (ii) holds then using $|vx| = q > 0$ *dictated by PL* $uwy = a^{n+2-q}b^{n+1}c^{n}$ *or* $uwy = a^{n+2}b^{n+1-q}c^{n}$ *which imply* $uwy \notin L$ *since* $n+2-q \leq n+1$ *or* $n+1-q \leq n$;

on the other hand if (iii) holds then $uv^2wx^2y \notin L$ *since* $uv^2wx^2y = a^{n+2}b^{n+1}c^{n+q}$ *and* $n+1 \leq n+q$. *Other two cases are (iv)* $vwx = a^i b^j$ *or (v)* $vwx = b^i c^j$ *with* $n \geq i+j \geq q > 0$;

if (iv) holds then $uwy = a^{n+2-q1}\, b^{n+1-q2}\, c^{n} \notin L$ *since* $q1+q2 = q > 0$; *if (v) holds then* $uv^2wx^2y = a^{n+2}b^{n+1+q1}\, c^{n+q2} \notin L$ *since* $q1+q2 = q > 0$ *and if* $q1 > 0$ *then* $n+2 \leq n+1+q1$ *and if* $q1=0$ *then* $q2=q$ *and* $n+1+q1 = n+1 \leq n+q2 = n+q$.

*4 - Let **n** be as in Pumping Lemma (PL) and choose* $z = a^n b^n a^n b^n \in L$*. Then by PL* $a^n b^n a^n b^n = uvwxy$ *.We show that* $uwy \notin L$*, a contradiction to PL.*

Since by PL $|vwx| \leq n$*, either ; (i)* $vwx = a^k$ *or* $vwx = b^k$ *;* $0 < k \leq n$ *, or :*

(ii) $vwx = a^r b^q$ *or ;* $vwx = b^r a^q$ *;* $0 < r+q \leq n$*, and by PL* $p := |vx| > 0$ *.*

If (i) holds then $uwy = a^{n-p} b^n a^n b^n$ *or* $= a^n b^n a^{n-p} b^n$ *; or*

$uwy = a^n b^{n-p} a^n b^n$ *or* $= a^n b^n a^n b^{n-p}$ *where **p** is as above hence clearly* $uwy \notin L$.

If (ii) holds then $uwy = a^i b^j a^n b^n$ *; or* $uwy = a^n b^j a^i b^n$ *;or* $uwy = a^n b^n a^i b^j$

with $i,j \leq n$ *and* $i+j = 2n-p < 2n$ *where again **p** is as above.*

in all cases above $uwy \notin L$*.*

Theorem 1

The (i) union, (ii) concatenation , (iii) Kleene ('') and positive ('+') closure and (iv) string reversal of CFLs are context-free languages.*

Proof : *Suppose* $\boldsymbol{S_1}$ *and* $\boldsymbol{S_2}$ *are the start symbols of* $\boldsymbol{L_1}$ *and* $\boldsymbol{L_2}$ *then*

(i) Set $\boldsymbol{S \rightarrow S_1 \mid S_2}$ *for* $\boldsymbol{L = L_1 \cup L_2}$

(ii) Set $\boldsymbol{S \rightarrow S_1 . S_2}$ *for* $\boldsymbol{L = L_1 . L_2}$

(iii) Set $\boldsymbol{S \rightarrow S_1 . S}$ *for* $\boldsymbol{L = L_1{}^*}$

(iv) Construct $\boldsymbol{G_R}$ *by reversing each production in* $\boldsymbol{G}$ **.**

Then each leftmost derivation of $\boldsymbol{w}$ *in* $\boldsymbol{G}$ *has a symmetric rightmost derivation in* $\boldsymbol{G_R}$ *that generates* $\boldsymbol{w^R}$

Theorem 2

If $\mathbf{L_P}$ is a CFL and $\mathbf{L_A}$ a regular language then $\mathbf{L_P \cap L_A}$ is a CFL

*Proof : Let the PDA **P** accept $\mathbf{L_P}$ and let the DFA **A** accept $\mathbf{L_A}$.*

Then the product automaton $\mathbf{P \times A}$ which is a PDA accepts $\mathbf{L_P \cap L_A}$

*What is a product automaton $\mathbf{P \times A}$? It is a **PDA** defined as below :*

*Let δ_P and δ_A be the transition functions of **P** and **A** then :*

$\mathbf{((q',r'),\alpha) \in \delta_{P\times A}((q,r),a,X)}$ ***iff***

$\mathbf{(q',\alpha) \in \delta_P(q,a,X)}$ ***and :*** *(i) if $\mathbf{a \neq e}$ then $\mathbf{\delta_A(r,a)=r'}$ **;** (ii) if $\mathbf{a=e}$ then $\mathbf{r'=r}$*

*where **q,q'** and **r,r'** are elements of the state sets **Q** of **P** and **R** of **A** respectively.*

Using

$((q,r),a,X) \rightarrow ((q',r'),\alpha)$ **iff**

$(q,a,X) \rightarrow (q',\alpha) \wedge (r' = \delta_A(r,a)$ *or* $r'=r$ *if* $a=e)$

Show that (using induction on the length of string u *)*

$((q,r), u ,\gamma) \vdash^*_{PxA} ((q',r'),e , \gamma')$ **iff**

$(q, u ,\gamma) \vdash^*_{P} ((q',e , \gamma') \wedge r' = \delta_A E(r,u)$

Applying above with $f_P \in F_P$ *and* $f_A \in F_A$

$((q_{0P} , q_{0A}),w,Z_0) \vdash^*_{PxA} ((f_P,f_A),e,\gamma')$ **iff** $w \in L_{PxA}$ **iff**

$(q_{0P},w , Z_0) \vdash^*_{P} ((f_P,e, \gamma') \wedge f_A = \delta_A E(q_{0A},w)$ **iff**

$w \in L_P \wedge w \in L_A$ **iff** $w \in L_P \cap L_A$

Theorem 3

The intersection and complementation of CFLs are not necessarily context-free

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

CFL's *not a CFL !*

We prove (by contradiction) that ***complementation*** *does not necessarily preserve the 'context free'ness property using De Morgan's formula :*

$$A \cap B = (A^c \cup B^c)^c$$

Measuring Complexities

Note that if $m < n$
$O(m) \Rightarrow O(n)$
hence $O(|V|) \Rightarrow O(n)$ etc

For $\mathbf{G = (V , \Sigma , R , S)}$ *measure of size is :*

$n := |V| + |\Sigma| + |R|.K$, *hence :* $O(|V| + |\Sigma| + |R|.K) = O(n)$

where $\mathbf{K}$ *is the maximum of length among all productions.*

For $\mathbf{P = (Q , \Sigma, \Gamma, \delta, q_0 , Z_0 , F)}$ *measure of size is :*

$n := |Q| + |\Sigma| + |\Gamma| + |\delta| K$, *hence :* $O(|Q| + |\Sigma| + |\Gamma| + |\delta| K) = O(n)$

where $\mathbf{K}$ *is the maximum of length among all transitions.*

Conversion from **G** *to* **P** *is* **:**

$$\textbf{Size of } P = O(\,|\Sigma| + (|\Sigma| + |V|) + |\Sigma| + |R|.K\,) = O(n)$$

inputs stack input trans. stack transitions

Note that : $O(\,|V| + |\Sigma| + |R|.K\,) = O(n)$ *implies* $|V|, |\Sigma|, |R|, K$

are all $\leq$ $n := |V| + |\Sigma| + |R|.K$ *hence* $= C.n$ *for some* $0 < C < 1$

hence each term is an **O(n)** *expression.*

similarly $O(\,|Q| + |\Sigma| + |\Gamma| + |\delta|K\,) = O(n)$ *implies*

$|Q|, |\Sigma|, |\Gamma|, |\delta|, K$ *are each an* **O(n)** *expression.*

Conversion from ***P*** *to* ***G*** *is as follows : for each transition*

$(p\,,\,Y_1\,Y_2 \dots Y_k\,) \in \delta(q,\,a,\,X)$ *there are productions*

$|Q|^k;\ |Q| < n\ ;\ k < K < n$
productions/transition

$[q\,X\,q_k] \rightarrow a[p\,Y_1\,q_1] \dots [q_{k-1}\,Y_k\ q_k]$ *for all* $q_1, \dots, q_k$ *in* Q

which sum up to $O(n^K) = O(n^n)$ *where* $|Q| = O(n)$ *and*

$K :=$ ***max*** *{length of all transitions}* $= O(n)$ *!*

This is exponential in ***n*** *! But there is a solution :*

Decompose each $(p,\,Y_1\,Y_2 \dots Y_k\,) \in \delta(\,q\,,a\,,\,X\,)$ *as* ***k-1*** *transitions :*

$\delta(q,\,a,\,X) = \{(p_{k-1}\,,\,Y_{k-1}\,Y_k\,)\} \rightarrow$ *push* $Y_{k-1}\,Y_k$,

O(k) $\Rightarrow$ ***O(n)***

$\delta(p_{k-1}\,,\,e,\,Y_{k-1}) = \{(p_{k-2}\,,\,Y_{k-2}\,Y_{k-1})\} \rightarrow$ *push* $Y_{k-2}\,Y_{k-1}$,

$\delta(p_2\,,\,e\,,\,Y_2) = \{(p\,,\,Y_1\,Y_2)\}$ *then new production length size is* ***2*** *and*

total ***productions/transition*** *:* $|k-1|.|Q|^2 = O(n).O(n)^2 = O(n^3)$

Complexity of conversion to Chomsky Normal Form

(1) *Elimination of* ***null*** *productions: limit production size to* ≤ 2.

Do this by replacing $\boldsymbol{B \rightarrow X_1 X_2 \ldots X_k}$ *by* $\boldsymbol{B \rightarrow X_1 Y_1 ; Y_1 \rightarrow X_2 Y_2}$ ***etc.***

hence a production of size ***k*** *is replaced by* ***k*** *productions of size* ***2*** *via* ***k-1***

new variables $\boldsymbol{Y_1}$ *to* $\boldsymbol{Y_{k-1}}$ *; then complexity is* ***not*** *exponential in* ***n***

(2) *Eliminating the unit productions*

Computation of unit pairs ***(i,j) =*** *Connectivity of a graph with* ***O(n)*** *nodes:*

Warshall's *algorithm : complexity* $\boldsymbol{O(n^3)}$

effective size of new grammar ***O(n) : Why ?*** *Productions repeated !*

Complexity of conversion to Chomsky Normal Form (Cont')

(3) *Elimination of non-generating and non-reachable symbols* $\boldsymbol{O(n^3)}$

no, of productions

(i) *Elimination of non-generating variables.*

size of productions

Basis *: every terminal symbol is generating* $\boldsymbol{a \Rightarrow^* a}$ *(zero-step derivation)*

Induction : *If in* $A \rightarrow \alpha$ *production every component of the* α *sequence is generating then* ***A*** *is generating. Check the right hand side of every production:complexity is* $\boldsymbol{O(|R|.(|K|.O(n)) = O(n^3)}$ **;**

Each element of the RHS of each production is compared with ***each*** *element of the list of generating variables .*

(ii) *Elimination of non-reachable variables.*

Every 'each' above is O(n)

Digraph where there is an edge from ***A*** *(variable) to* ***X*** *(a variable or a terminal symbol).Reachability with # nodes* $|V|+|T|$ *; initial node =* ***S .***

Complexity : $\boldsymbol{O((|V|+|T|)^2) = O(n^2)}$

Complexity of conversion to Chomsky Normal Form (Cont')

(4) *Replacement of terminals by variables :**O(n)** ;*

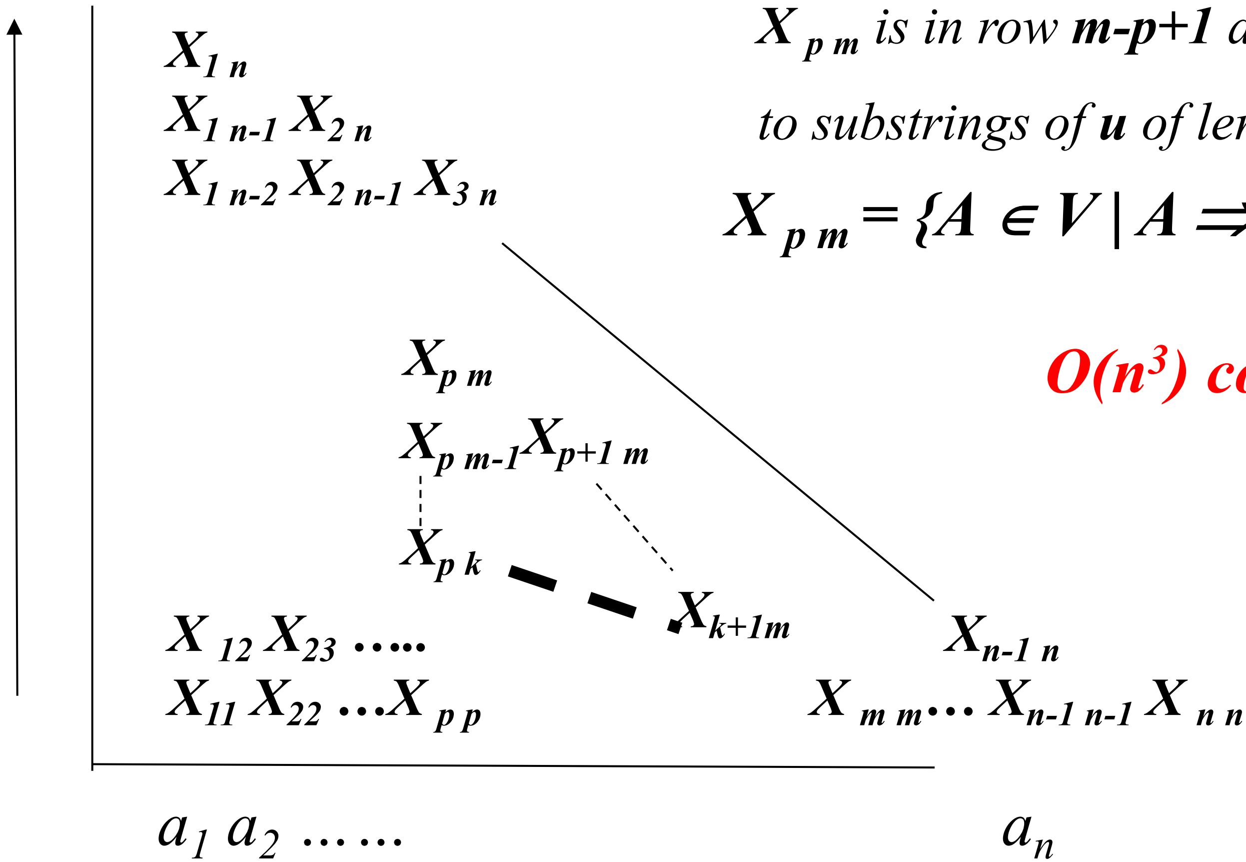
*size of new grammar **O(n)***

(5) *Breaking of bodies of size **> 2** into **2 :***

O(n) ; *size of new grammar **O(n)***

Result : Computational complexity of CNF reduction is $O(n^3)$

Is $\boldsymbol{u = a_1 a_2 \ldots a_n}$ a member of $\boldsymbol{L_G}$?



*$X_{p\,m}$ is in row **m-p+1** and corresponds to substrings of **u** of length **m-p+1***

$$X_{p\,m} = \{A \in V \mid A \Rightarrow_G^* a_p\, a_{p+1} \ldots a_m\}$$

$O(n^3)$ computations

***$O(n^2)$** entries of table, each has **$O(n)$** pairs to compute, for each pair **$O(1)$** effort !*

Divide $X_{p\,m}$ as $X_{p\,k}$ and $X_{k+1\,m}$ and check for $A \rightarrow BC$ where $B \in X_{p\,k}$ and $C \in X_{k+1\,m}$

Example

$S \rightarrow aSb \mid e$ … $S \rightarrow ASB \mid AB$ … $S \rightarrow AC \mid AB \,;\, C \rightarrow SB \,;$ …

CNF :

$S \rightarrow AC \mid AB \,,\, C \rightarrow SB \,,\, A \rightarrow a \,,\, B \rightarrow b$

$X_{11} = X_{22} = \{X \in V \mid X \Rightarrow_G^* a\} = A$
$X_{33} = X_{44} = \{X \in V \mid X \Rightarrow_G^* b\} = B$

Is **aabb** *in* L_G ?

$X_{11} = X_{22} = \{A\} \,;\, X_{33} = X_{44} = \{B\}$ *using* $A \rightarrow a \,,\, B \rightarrow b$

$X_{12} = X_{34} = \varnothing \;;\; (X_{22}, X_{33})$ *generated by* $S \rightarrow AB$ *hence* $X_{23} = \{ S \}$

$X_{13} = \varnothing;\; (X_{23}, X_{44})$ *generated by* $C \rightarrow SB$ *hence* $X_{24} = \{ C \}$

(X_{11}, X_{24}) *generated by* $S \rightarrow AC$ *hence* $X_{14} = \{ S \}$

Hence $S \Rightarrow^* aabb$

Determinism in PDA and Parsing

Simple Example for top down look – ahead parser

$S \rightarrow a\,S\,b \mid e$ *leads to PDA below:*

$\delta(q_0, e, Z_0) = \{(q, SZ_0)\}$

$\delta(q, x, x) = \{(q, e)\}$ for $x = a$ and $x = b$

$\delta(q, e, S) = \{(q, aSb), (q, e)\}$ Non-determinism !!

$\delta(q, e, Z_0) = \{(f, Z_0)$ **OR** $(q, e)\}$

$\delta(q_0, e, Z_0) = \{(q, SZ_0)\}$

$\delta(q, a, S) = \{(q_a, S)\}$

$\delta(q_a, e, S) = \{(q, Sb)\}$

Look-ahead for input a

$\delta(q, b, S) = \{(q_b, S)\}$

$\delta(q_b, e, S) = \{(q_b, e)\}$

$\delta(q_b, e, b) = \{(q, e)\}$

Look-ahead for input b

DPDA

$\delta(q, b, b) = \{(q, e)\}$

$\delta(q, a, a) = \{(q, e)\}$ *(transition not used)*

$\delta(q, e, Z_0) = \{(f, Z_0)\}$ **OR** $\{(q, e)\}$

Top down parsing

Given a grammar $\boldsymbol{G}$ *we elaborate on the productions before we apply a modified version of the PDA given in the proof of the theorem : from* $\boldsymbol{G}$ *to* ***PDA***

(i) *If* $A \rightarrow c\ \alpha_1 \mid \ldots \mid c\ \alpha_n$ *is a collection of productions of* $\boldsymbol{G}$ *where* $\boldsymbol{c}$ *is a terminal or a nonterminal then replace these productions by :*

$A \rightarrow cA'$ *and* $A' \rightarrow \alpha_1 \mid \ldots \mid \alpha_n$ *where* A' *is a new variable.*

The resulting grammar $\boldsymbol{G_1}$ *yields the same language as* $\boldsymbol{G}$

(ii) *(Left recursion) If* $A \rightarrow A\alpha_1 \mid \ldots \mid A\alpha_n$ *and* $A \rightarrow \beta_1 \mid \ldots \mid \beta_m$ *are productions where* $n,m > 0$ *and first element of each* β_i *is different from* $\boldsymbol{A}$ *then replace these productions by :*

$A \rightarrow \beta_1 B \mid \ldots \mid \beta_m B$ *and* $B \rightarrow \alpha_1 B \mid \ldots \mid \alpha_n B \mid e$.

The resulting grammar $\boldsymbol{G_2}$ *yields the same language as* $\boldsymbol{G}$

These arguments lead us to the Greibach Normal Form (GNF):

Each production is of the type $\boldsymbol{A \rightarrow a\ \alpha}$ *where* ***a*** *is a terminal.*

Apply case ***(i)*** *above and if necessary* ***GNF*** *repeatedly until*

for each production group $\boldsymbol{A \rightarrow a_1\ \alpha_1 \mid \ldots \mid a_m\ \alpha_m}$ *the terminals* $\boldsymbol{a_j}$

are all ***distinct ;***

and so the ***lookahead*** *technique of* ***top down parsing*** *can be*

applied via a ***DPDA*** *in the manner demonstrated by the example*

$\boldsymbol{L_G = \{a^n b^n\}}$ *done in class*

Example

CFG is $G = (V, \Sigma, R, E)$ *where*

$V = \{E, T, F, I\}$; $\Sigma = \{+, *, (,), x, y, z\}$ (x, y, z *are either variables or function letters)*

R :

$E \to E + T \mid T$; $T \to T * F \mid F$; $F \to I \mid (E) \mid I(E)$; $I \to x \mid y \mid z$

PDA : $P = (\{q_0, s, f\}, \Sigma, V \cup \Sigma \cup \{Z_0\}, \delta, q_0, Z_0, \{f\})$

$\delta(q_0, e, Z_0) = \{(s, E Z_0)\}$

$\delta(s, t, t) = \{(s, e)\}$ *for all* $t \in \Sigma$

$\delta(s, e, E) = \{(s, E+T), (s, T)\}$

$\delta(s, e, T) = \{(s, T*F), (s, F)\}$

$\delta(s, e, F) = \{(s, I), (s, (E)), (s, I(E))\}$

$\delta(s, e, I) = \{(s, x), (s, y), (s, z)\}$

$\delta(s, e, Z_0) = \{(f, Z_0)\}$

(1) Fix left recursion :

replace $E \rightarrow E + T \mid T$ *by* $E \rightarrow T\,B$; $B \rightarrow +T\,B \mid e$

replace $T \rightarrow T * F \mid F$ *by* $T \rightarrow F\,C$; $C \rightarrow * F\,C \mid e$

(2) Fix common production start symbol :

replace $F \rightarrow I \mid (E) \mid I\,(E)$ *by* $F \rightarrow I\,A \mid (E)$ *and* $A \rightarrow (E) \mid e$

(3) Substitute until GNF-like structure prevails !! No need for ***I*** *at the end*

$E \rightarrow$ x-y-z $ACB \mid (E)\,CB$

$E \rightarrow T\,B \rightarrow F\,CB \rightarrow I\,ACB \mid (E)\,CB$

$B \rightarrow + TB \mid e$

$T \rightarrow$ x-y-z $AC \mid (E)\,C$

$T \rightarrow FC \rightarrow I\,A\,C \mid (E)\,C$

$C \rightarrow * F\,C \mid e$

$F \rightarrow$ x-y-z $A \mid (E)$

$A \rightarrow (E) \mid e$

look-ahead works for this *GNF !*

extra 7 states required

The (DPDA ?) for the grammar G defined above !

$(q_0, e, Z_0) \rightarrow (s, EZ_0)$

$(s, + - * - (-) - x\text{-}y\text{-}z, V) \rightarrow (q_+ - q_* - q_(- q_) - q_x\text{-}q_y\text{-}q_z, V)$

$(q_+, e, C\text{-}A) \rightarrow (q_+, e)$

$(q_+, e, B) \rightarrow (s, TB)$

$(q_*, e, A\text{-}B) \rightarrow (q_*, e)$

$(q_*, e, C) \rightarrow (s, FC)$

$(q_(, e, C\text{-}B) \rightarrow (q_(, e)$

$(q_(, e, A\text{-}F\text{-}T\text{-}E) \rightarrow (s, E) - E) - E)C - E)CB)$

$(q_), e, C\text{-}A\text{-}B) \rightarrow (q_), e)$

$(q_x, e, E\text{-}T\text{-}F) \rightarrow (s, ACB - AC - A)$

$(q_y, e, E\text{-}T\text{-}F) \rightarrow (s, ACB - AC - A)$

$(q_z, e, E\text{-}T\text{-}F) \rightarrow (s, ACB - AC - A)$

$(s, input, input) \rightarrow (s, e)$

$(q_{input}, e, input) \rightarrow (s, e)$

$(s, e, A\text{-}C\text{-}B) \rightarrow (s, e)$

$(s, e, Z_0) \rightarrow (f, Z_0)$

$(q_x, e, A\text{-}B\text{-}C) \rightarrow (q_x, e)$

$(q_y, e, A\text{-}B\text{-}C) \rightarrow (q_y, e)$

$(q_z, e, A\text{-}B\text{-}C) \rightarrow (q_z, e)$

V = any non-terminal variable

non-deterministic transitions

This transition is not used in the following example

Resolution : prioritize $(q,a,V) \rightarrow ..$ to $(q,e,V) \rightarrow ..$

$E \rightarrow x\text{-}y\text{-}z\ ACB \mid (E)\ CB$

$B \rightarrow +\ TB \mid e$

$T \rightarrow x\text{-}y\text{-}z\ AC \mid (E)\ C$

$C \rightarrow *\ F\ C \mid e$

$F \rightarrow x\text{-}y\text{-}z\ A \mid (E)$

$A \rightarrow (E) \mid e$

Parse $: x+(y*z(x)+x)$

$E \to$ x-y-z $ACB \mid (E)\ CB$

$B \to + TB \mid e$

$T \to$ x-y-z $AC \mid (E)\ C$

$C \to *FC \mid e$

$F \to$ x-y-z $A \mid (E)$

$A \to (E) \mid e$

$(s, x+(y*z(x)+x), E Z_0)$ |-- $(q_x, +(y*z(x)+x), E Z_0)$

|-- $(s, +(y*z(x)+x), ACB Z_0)$ |-- $(q_+, (y*z(x)+x), ACB Z_0)$

|-- $(q_+, (y*z(x)+x), CB Z_0)$ |-- $(q_+, (y*z(x)+x), B Z_0)$

|-- $(s, (y*z(x)+x), TB Z_0)$ |-- $(q_(, y*z(x)+x), TB Z_0)$

|-- $(s, y*z(x)+x), E)CB Z_0)$ |-- $(q_y, *z(x)+x), E)CB Z_0)$

|-- $(s, *z(x)+x), ACB)\ C B Z_0)$ |-- $(q_*, z(x)+x), ACB)\ C B Z_0)$

|-- $(q_*, z(x)+x), CB)\ C B Z_0)$ |-- $(s, z(x)+x), FCB)\ C B Z_0)$

|-- $(q_z, (x)+x), FCB)\ C B Z_0)$ |-- $(s, (x)+x), ACB)\ C B Z_0)$

|-- $(q_(, x)+x), ACB)\ C B Z_0)$ |-- $(s, x)+x), E)CB)\ C B Z_0)$

|-- $(q_x,)+x), E)CB)\ C B Z_0)$ |-- $(s,)+x), ACB)CB)\ C B Z_0)$

|-- $(q_), +x), ACB)CB)\ C B Z_0)$ |-- $(q_), +x), CB)\ C B)\ C B Z_0)$ |--

. . . |-- $(q_), +x),)\ C B)\ C B Z_0)$

|-- $(s, +x), C B)\ C B Z_0)$ |-- $(q_+, x), CB)\ C B Z_0)$

|-- $(q_+, x), B)\ C B Z_0)$ |-- $(s, x), TB)\ C B Z_0)$ |-- $(q_x,), TB)\ C B Z_0)$

|-- $(s,), ACB)\ C B Z_0)$ |-- $(q_), e, ACB)CB Z_0)$ . . . |-- (s, e, CBZ_0) |-- . . .

(s, e, Z_0) |-- (f, e, Z_0) **TOMBALA !!!!!!!**

EXAMPLE 1

$L_1 = \{a^n b^m \; ; \; n > m \geq 0 \}$

CFG :

$S \rightarrow AC \; ; \; A \rightarrow aA \mid a \; ; \; C \rightarrow aCb \mid e$

PDA (algorithmic):

$(q_0, e, Z_0) \rightarrow (q, SZ_0)$ **initial**

$(q, e, S) \rightarrow (q, AC)$

$(q, e, A) \rightarrow (q, aA) \; ; \; (q, e, A) \rightarrow (q, a)$

$(q, e, C) \rightarrow (q, aCb) \; ; \; (q, e, C) \rightarrow (q, e)$

$(q, a, a) \rightarrow (q, e)$

$(q, b, b) \rightarrow (q, e)$

$(q, e, Z_0) \rightarrow (f, Z_0)$ **final**

PDA (direct logic): ***f,f'*** final states

$(q_0, a, Z_0) \rightarrow (f, Z_0)$

$(f, a, Z_0) \rightarrow (f, aZ_0)$

$(f, a, a) \rightarrow (f, aa)$

$(f, b, a) \rightarrow (f', e)$

$(f', b, a) \rightarrow (f', e)$

Note that at state ***f'*** *: (i) if top of the stack is* Z_0 *it cannot consume any further* **b** *inputs since no such transitions are defined ; (ii) if top of the stack is* **a** *then it continues popping* **a**'*s until top becomes* Z_0 *; or all* **b's** *are consumed !*

is this a DPDA ? YES

EXAMPLE 2

#a's+#B's = #b's+#A's in every sentence of derivation except for $S \rightarrow Sa$; aS productions to place excess a's

$L_2 = \{ \#a\text{'s} \geq \#b\text{'s} \}$

CFG :

$P : S \rightarrow aAS \mid bBS \mid Sa \mid aS \mid e$;
$A \rightarrow aAA \mid b$;
$B \rightarrow bBB \mid a$

PDA *(algorithmic):*

$(q_0, e, Z_0) \rightarrow (q, SZ_0)$ *initial*

$(q, e, S) \rightarrow (q, aAS)$; etc

$(q, e, A) \rightarrow (q, aAA)$; etc

$(q, e, B) \rightarrow (q, bBB)$ etc

$(q, a, a) \rightarrow (q, e)$

$(q, b, b) \rightarrow (q, e)$

$(q, e, Z_0) \rightarrow (f, Z_0)$ *final*

PDA *(direct logic):*

$(q_0, a, Z_0) \rightarrow (f, aZ_0)$

$(f, a, Z_0) \rightarrow (f, aZ_0)$

$(f, a, a) \rightarrow (f, aa)$

$(q_0, b, Z_0) \rightarrow (q_0, bZ_0)$

$(q_0, b, b) \rightarrow (q_0, bb)$

$(q_0, a, b) \rightarrow (q_0, e)$

$(f, b, Z_0) \rightarrow (q_0, bZ_0)$

$(f, b, a) \rightarrow (f, e)$

is this a DPDA ? YES