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

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

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 **: $G1 = (V_1, T_1, R_1, S)$.***

Do this by the following inductive method :

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

***Induction** : If for a production $A \rightarrow \alpha$ 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 $X \rightarrow \alpha$ and $C \rightarrow \alpha X \beta$

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

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

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

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

***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

$S \rightarrow A\ B \mid a$

$B \rightarrow b$

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 :*

$S \rightarrow a$

$B \rightarrow b$

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 $S \rightarrow A\ B$ *and then eliminate the* ***non-reachable*** *symbol* ***B*** *and*

the associated production $B \rightarrow b$

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* $S \Rightarrow_G \ldots \Rightarrow_G \alpha_j \ldots \Rightarrow_G w$, *be a derivation of* w *in* **G** *then*

$S \Rightarrow_{G2} \ldots \Rightarrow_{G2} \alpha_j \ldots \Rightarrow_{G2} w$ *,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 : $A \rightarrow e$

Definition

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

Compute all nullable variables inductively

Basis : A is nullable if $A \rightarrow e$;

Induction : If $B \rightarrow C_1 C_2 \ldots C_n$ 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:* $A \rightarrow X_1 X_2 \ldots X_m$ *where* $k \leq m$ *of the* X_j

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

Include in $\boldsymbol{R'}$, $\boldsymbol{2^k}$ *productions where each nullable* $\boldsymbol{X_j}$ *is present or absent; (except when* $\boldsymbol{m=k}$ *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 **:**

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

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

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

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

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

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

*(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* $\boldsymbol{G' = (V,T,R',S)}$ *from* $\boldsymbol{G}$

(1) Compute all ***unit pairs*** *of* $\boldsymbol{G}$

(2) Include all ***non-unit productions*** *of* $\boldsymbol{R}$ *in* $\boldsymbol{R'}$ *and in addition*

for each unit pair $\boldsymbol{(A,B)}$ *add to* $\boldsymbol{R'}$ *the production* $\boldsymbol{A \rightarrow \alpha}$ *if* $\boldsymbol{B \rightarrow \alpha}$ *is*

a non-unit production in $\boldsymbol{R}$

Theorem $\boldsymbol{L_{G'} = L_G}$

Chomsky Normal Form (CNF)

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

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

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

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 $W \to X_1 X_2 \dots X_n$, *if* $X_i \in T$ *then replace* X_i *with a new variable* Λ_i *in this production and add the new production* $\Lambda_i \to X_i$

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

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

$E \to T \mid E+T$

$T \to F \mid T*F$

$F \to I \mid (E)$

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

$J \to 0J \mid 1J \mid e$

Eliminate null production $J \to e$

$E \to T \mid E+T$

$T \to F \mid T*F$

$F \to I \mid (E)$

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

$J \to 0J \mid 1J \mid 0 \mid 1$

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

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

Eliminate unit pairs

$E \to T \to F \to I$

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

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

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

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

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

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

$J \to 0J \mid 1J \mid 0 \mid 1$

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

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

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

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

Example (Chomsky Normal Form , continued)

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

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

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

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

$zero \rightarrow 0$

$one \rightarrow 1$

$X \rightarrow x$

$[\rightarrow ($

$] \rightarrow)$

$mult \rightarrow *$

$add \rightarrow +$

$A \rightarrow X\ one$

$B \rightarrow [\ E$

$C \rightarrow T\ mult$

$D \rightarrow E\ add$

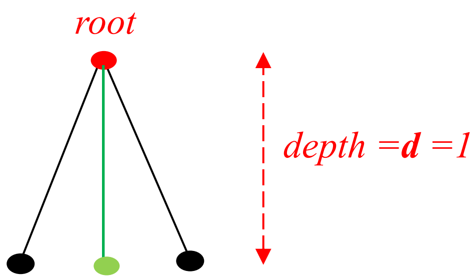
$A \rightarrow X\ one \quad B \rightarrow [E \quad C \rightarrow T\ mult \quad D \rightarrow E\ add$

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

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

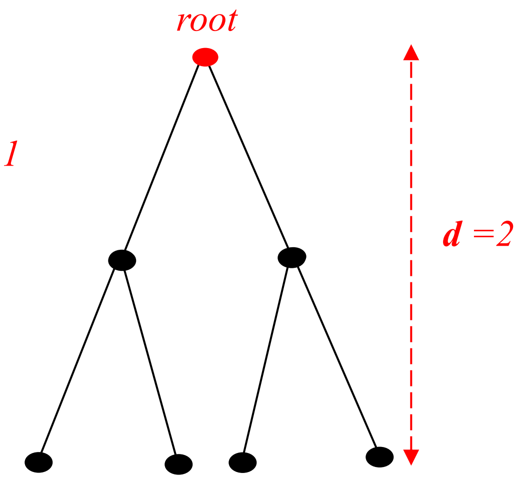
$F \rightarrow 0 \mid 1 \mid one\ J \mid X\ zero \mid X\ one \mid X\ one\ J \mid [E]$

A word on Binary Trees

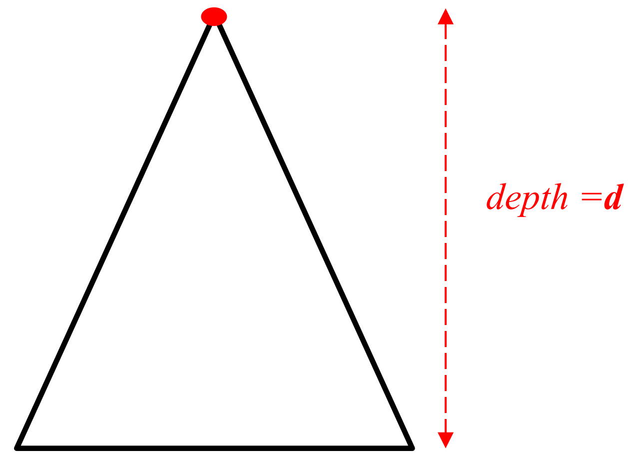


*at most **2** chidren*
*for a **full** binary tree*

could be 1 if not full



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



*at most 2^d children 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 !*

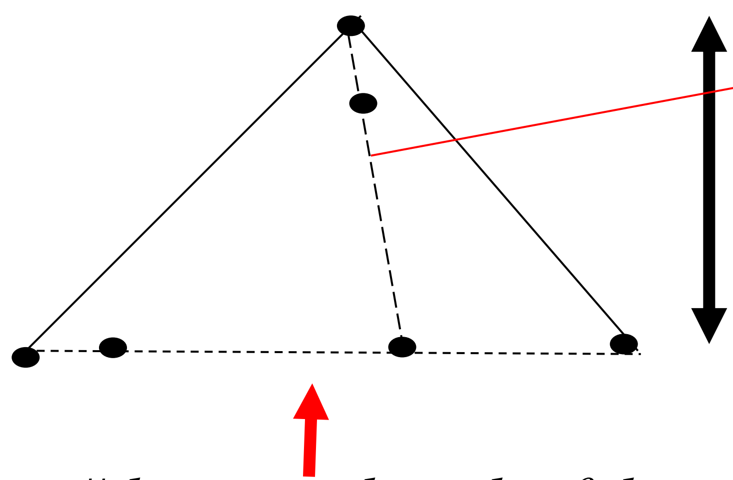
The Pumping Lemma for CFGs

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

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

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

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



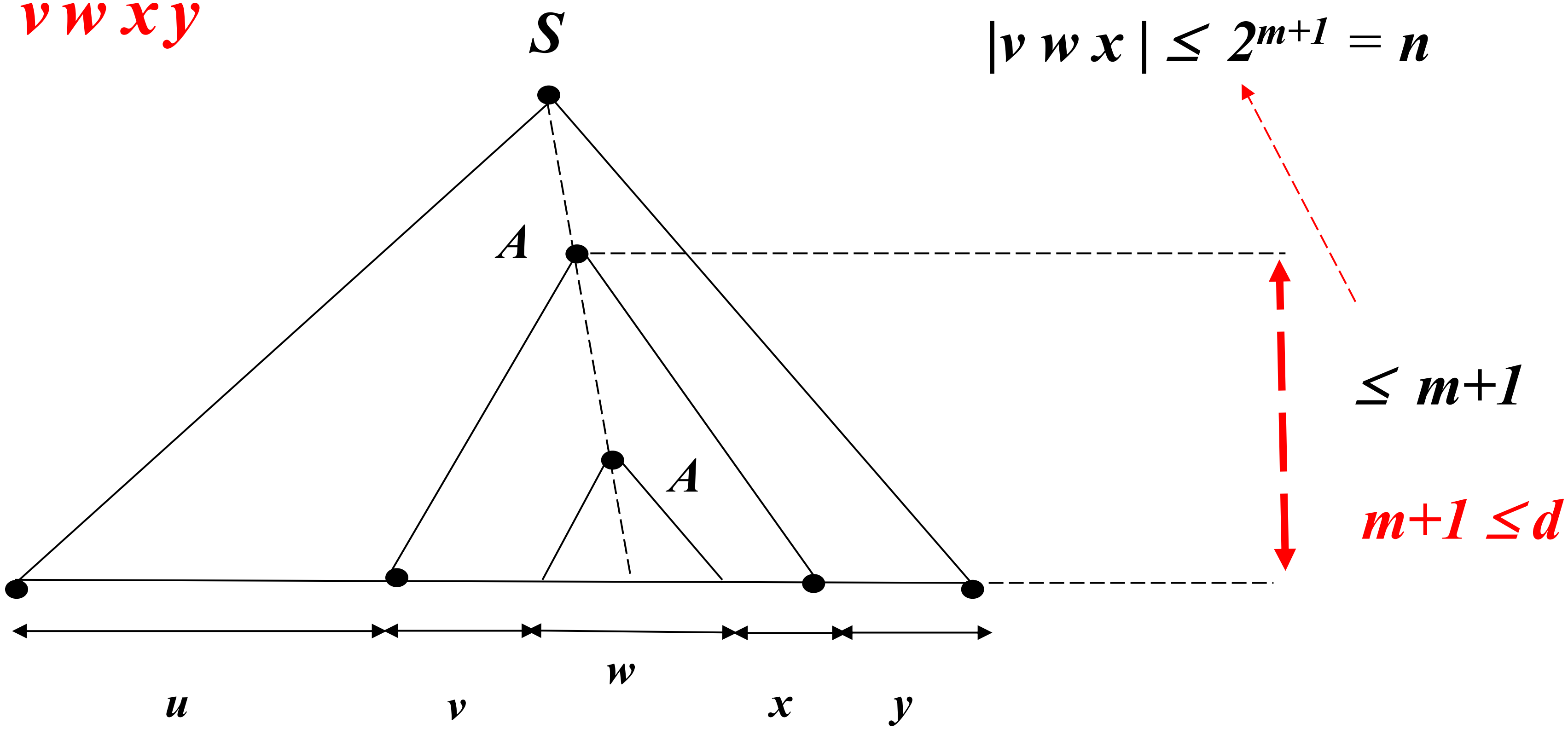
longest path length = d $\Rightarrow$
d ***<u>variable</u> nodes on the path***

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

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

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

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



$A \Rightarrow^* v\,A\,x$ *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* $i \geq 0$

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

1 – Let 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 $z=a^nb^nc^nd^n \in L$. Then by PL $a^nb^nc^nd^n = uvwxy$ 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 $vwx=a^ib^j$ or $=b^ic^j$ or $= c^id^j$ where $0 < i+j \leq n$

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

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

If (ii) holds then $uwy = a^mb^kc^n d^n$ or $= a^nb^mc^k d^n$ or $= a^nb^nc^md^k$ 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* $z = a^n b^n a^n b^n \in L$. *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 $uwy \notin L$.

Properties of Context Free Languages

Theorem 1 (Substitution)

Let $\mathbf{L}$ be a CFL over an alphabet $\boldsymbol{\Sigma}$ and for each $\mathbf{a} \in \boldsymbol{\Sigma}$ let $\mathbf{L(a)}$ be a CFL over an alphabet $\boldsymbol{\Sigma_a}$. Then the language :

$$\mathbf{L_s := \{ u \in \Sigma^* \mid u = l(a_1).l(a_2) \ldots l(a_n) \;;\; a_1.a_2 \ldots a_n \in L \;;\; l(a_k) \in L(a_k)}$$

for $\mathbf{k = 1, \ldots, n \}}$ is a CFL over the alphabet $\cup_{a \in \Sigma} \boldsymbol{\Sigma_a}$

Proof : Let $\mathbf{G = (V, \Sigma, R , S)}$ generate $\mathbf{L}$ and let each $\mathbf{G_a = (V_a , \Sigma_a , R_a , S_a)}$ generate $\mathbf{L_a}$ then the grammar $\mathbf{G'}$ that replaces each '**a**' in the productions of $\mathbf{G}$ by $\mathbf{S_a}$ and incorporate all the variables and productions of $\mathbf{G_a}$'s in $\mathbf{G'}$ generates $\mathbf{L_s}$ …

Theorem 2

$S \to a|b$ *that generates* L *is replaced by* $S \to S_1 \,|\, S_2$

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

$S \to ab$ *that generates* L *is replaced by* $S \to S_1 S_2$

Proof : *Use substitution theorem for* ***(i)-(iii)*** *!*

(i) $L_1 \cup L_2 \to$ *Choose* $L = \{ a , b \}$ *and* $L_a = L_1$ *and* $L_b = L_2$

(ii) $L_1 L_2 \to$ *Choose* $L = \{ a\, b \}$ *and* $L_a = L_1$ *and* $L_b = L_2$

(iii) $M^*(M^+) \to$ *Choose* $L = \{ a \}^*$ $(L = \{ a \}^+)$ *and* $L_a = M$

(iv) *Construct* G_R *by reversing each production in* G *.*

Then each leftmost derivation of w *in* G *has a symmetric rightmost derivation in* G_R *that generates* w^R

$S \to aS \,|\, e$ *that generates* L *is replaced by* $S \to S_M S \,|\, e$

$S \to aS \,|\, a$ *that generates* L *is replaced by* $S \to S_M S \,|\, S_M$

Theorem 3

If $\boldsymbol{L}$ *is a CFL and* $\boldsymbol{R}$ *a regular language then* $\boldsymbol{L \cap R}$ *is a CFL*

Proof : *Let the PDA* $\boldsymbol{P}$ *accept* $\boldsymbol{L}$ *and let the DFA* $\boldsymbol{A}$ *accept* $\boldsymbol{R}$.

Then the product automaton $\boldsymbol{P \times A}$ *which is a PDA accepts* $\boldsymbol{L \cap R}$

What is a product automaton $\boldsymbol{P \times A}$ *?*

Let δ_P *and* δ_A *be the transition functions of* $\boldsymbol{P}$ *and* $\boldsymbol{A}$ *then :*

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

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

where $\boldsymbol{q,q'}$ *and* $\boldsymbol{r,r'}$ *are elements of the state sets* $\boldsymbol{Q}$ *of* $\boldsymbol{P}$ *and* $\boldsymbol{R}$ *of* $\boldsymbol{A}$

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*** *)*

$((q,r),\, u.v\,,\gamma) \mid\!\!-\!\!-^*_{PxA} ((q',r'),v,\, \gamma')$ ***iff***

$(q,\, u.v\,,\gamma) \mid\!\!-\!\!-^*_{P} ((q',v,\, \gamma') \wedge r' = \delta_A E(r,u)$

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

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

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

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

Theorem 4

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

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

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

where **K** *is the maximum of length among all productions.*

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

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

where **K** *is the maximum of length among all transitions.*

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

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

inputs *stack* *input trans.* *stack transitions*

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*

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

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

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

$\delta(p_2, e, Y_2) = \{(p, Y_1 Y_2)\}$ *then new* $K = 2$ *and* $|\delta|.K \rightarrow |\delta|.O(K) = O(n)$,

$|Q| \rightarrow |Q| + |\delta|\, O(K) = O(n)$ *hence total complexity is* $O(n^2)$

Complexity of conversion to Chomsky Normal Form

(1) Elimination of non-generating and non-reachable symbols $O(n^3)$

no, of productions

(i) Elimination of non-generating variables.

size of productions

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

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

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

Every 'each' above is O(n)

(ii) Elimination of non-reachable variables.

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 : $O((|V|+|T|)^2) = O(n^2)$

Complexity of conversion to Chomsky Normal Form (Cont')

(2) *Eliminating the unit productions*

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

Warshall's *algorithm : complexity* $O(n^3)$

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

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

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

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

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

Complexity of conversion to Chomsky Normal Form (Cont')

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

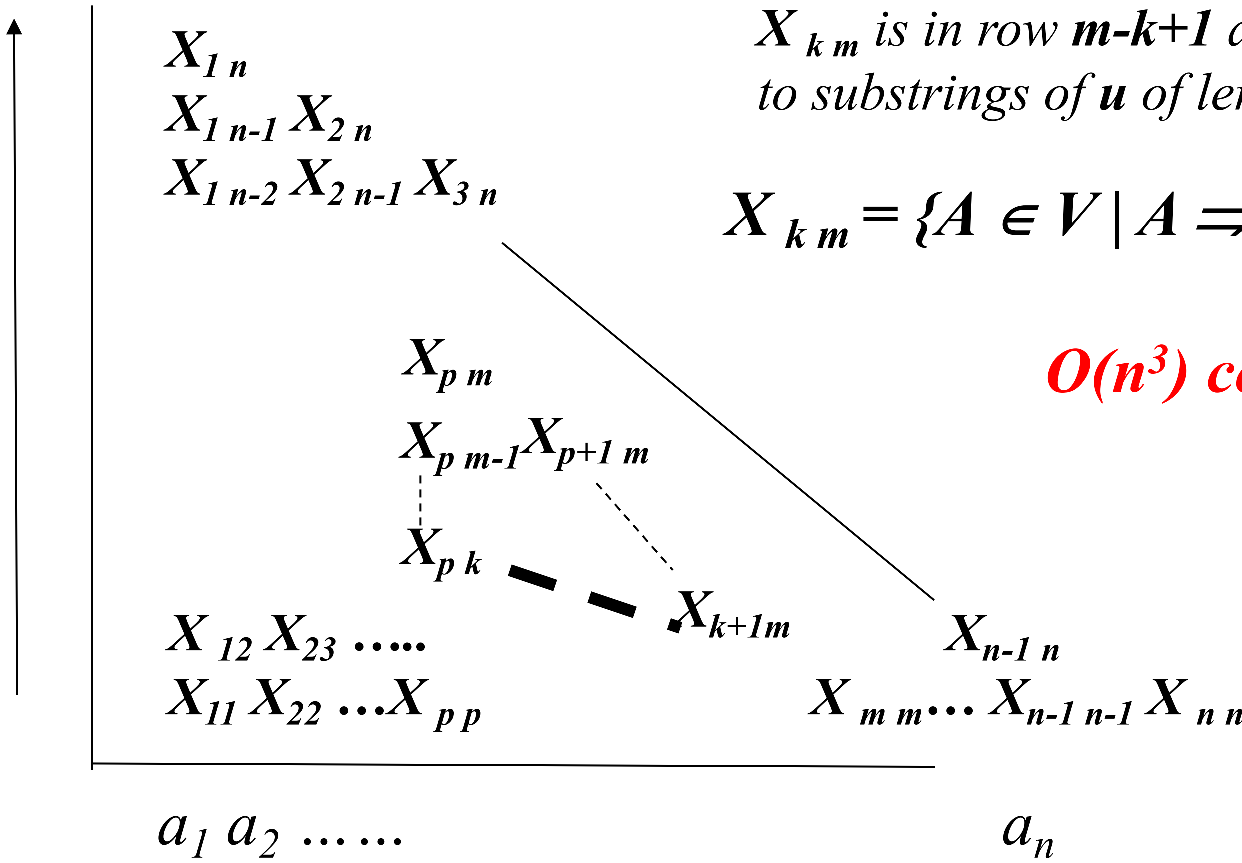
*size of new grammar **O(n)***

(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 $u = a_1 a_2 \ldots a_n$ a member of L_G?



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

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

$O(n^3)$ 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

$S \rightarrow aSb \mid e$

CNF :

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

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:*

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

$\delta(q_0, x, x) = \{ (q_0, e) \}$ for $x = a$ and $x = b$ **Non-determinism !!**

$\delta(q_0, e, S) = \{ (q_0, aSb), (q_0, e) \}$

$\delta(q_0, e, Z_0) = \{(f, SZ_1)\}$ *accepts* ***e=empty string***

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

$\delta(q_a, e, S) = \{(q_0, Sb)\}$ **Look-ahead for input a**

$\delta(q_0, a, S) = \{(q_a, S)\}$

$\delta(q_0, b, S) = \{(q_b, S)\}$ → **DPDA**

$\delta(q_b, e, S) = \{(q_b, e)\}$ **Look-ahead for input b**

$\delta(q_b, e, b) = \{(q_0, e)\}$

$\delta(q_0, b, b) = \{(q_0, e)\}$

$\delta(q_0, e, Z_1) = \{(f, Z_0)\}$

Top down parsing

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

(i) *If* $A \to c\,\alpha_1 \mid \dots \mid c\,\alpha_n$ *is a collection of productions of **G** where* ***c*** *is a terminal or a nonterminal then replace these productions by :*

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

The resulting grammar G_1 *yields the same language as **G***

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

$A \to \beta_1 B \mid \dots \mid \beta_m B$ *and* $B \to \alpha_1 B \mid \dots \mid \alpha_n B \mid e$.

The resulting grammar G_2 *yields the same language as **G***

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

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

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

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

$L_G = \{a^n b^n\}$ *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 :

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

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

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

$\delta(s, t, t) = \{(s, e)\}$ *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 $E \to E + T \mid T$ *by* $E \to TB$; $B \to +TB \mid e$

replace $T \to T * F \mid F$ *by* $T \to FC$; $C \to *FC \mid e$

(2) Fix common production start symbol :

replace $F \to I \mid (E) \mid I(E)$ *by* $F \to IA \mid (E)$ *and* $A \to (E) \mid e$

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

$E \to$ *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* $A \mid (E)$

$A \to (E) \mid e$

$E \to TB \to FCB \to IACB \mid (E)\,CB$

$T \to FC \to IAC \mid (E)\,C$

look-ahead works for this *GNF !*

extra 7 states required

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

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

$B \rightarrow +\ TB \mid e$

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

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

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

$A \rightarrow (E) \mid e$

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

V = any non-terminal variable

non-deterministic transitions

This transition is not used in the following example

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

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

$B \rightarrow +\ TB \mid e$

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

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

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

$A \rightarrow (E) \mid e$

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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