Week 8: Lecture Notes

Context free Grammars (CFG)

A context free grammar is $G = (V, T, P, S)$ where

- V is a finite set (non-empty) whose elements are called variables
- T is a finite set (non-empty) whose elements are called terminals
- $V \cap T = \phi$
- S is a special variable called the start variable
- P is a finite set of productions or rules whose elements are $A \to \alpha$ where A is a variable and α is a string of symbols from $(V \cup T)^*$

Example:

$G = (\{E\}, \{+, *, (,), id\}, P, E)$

where P represents the following set of rules

$E \to E+E$ $\quad$ $E \to (E)$

$E \to E*E$ $\quad$ $E \to id$.

Applying productions repeatedly, we can obtain more and more complicated expressions.

$\Rightarrow$ denotes act of deriving

$(id+id)*id$ consists of only terminal symbols, it is a word in the language generated by G.

$E \Rightarrow E+E$
$\Rightarrow (E)*E$
$\Rightarrow (E)*id$
$\Rightarrow (E+E)*id$
$\Rightarrow (E+id)*id$
$\Rightarrow (id+id)*id$

Example:

L_{pal} : languages of palindromes of 0's and 1's
- not regular

- $\varepsilon, 0, 1$ are palindromes
- if w is a palindrome, so are $0w0$ and $1w1$

Such recursively defined languages (L_{pal}) can be expressed formally as context free grammar (for palindromes) G_{pal}

$$G_{pal} = (\{E\}, \{0,1\}, P, E)$$

where P represents the set of five productions

$E \to E$ $\quad$ $E \to 0E0$

$E \to 1$ $\quad$ $E \to 1E1$

$E \to 0$

- The set of productions is the kernel of grammar and language specifications

- Note (Regarding production rules)

(i) Reverse substition is not permitted

e.g. if $S \to AB$ is a production, then we can replace S by AB but <u>we cannot replace</u> AB by S

(ii) No inversion operation is permitted

eg. if $S \to AB$ is a production, then it is not necessary that $AB \to S$ is a production.

Notations

1. The capita letters A, B, C, D, E and S denote variables. S is the start symbol
2. The lowercase letters a, b, c, d, e, digits, boldface strings are terminals.
3. The capital letters X, Y, Z denote symbols that may be either terminals or variables.
4. The lowercase letters u, v, w, x, y, z denote strings of terminals.
5. The lower case Greek letters α, β, γ denote strings of variables and terminals $(V \cup T)^*$

A CFG is often presented by simply listing its productions.

If $A \to \alpha_1, A \to \alpha_2, \ldots, A \to \alpha_k$ are productions for variable , then we may express them by the notation

$A \to \alpha_1 | \alpha_2 | \ldots | \alpha_k$ '|' stands for 'or'

Example:

$E \to E+E$, $E \to E*E$, $E \to (E)$, $E \to id$

can be written as

$E \to E+E \mid E*E \mid (E) \mid id$

Derivations and languages

- $G = (V, T, P, S)$ → CFG
- $\underset{G}{\Rightarrow}$ → relation on $(VUT)^*$

defined as follows:

if $A \to \beta$ is a production of P and $\alpha, \gamma \in (VUT)^*$

then $\alpha A \gamma \underset{G}{\Rightarrow} \alpha \beta \gamma$

$\alpha A \gamma$ directly derives $\alpha \beta \gamma$ in grammar G.

- Two strings are related by $\underset{G}{\Rightarrow}$ exactly when the second is obtained from the first by one application of some production

- $\overset{*}{\underset{G}{\Rightarrow}}$ relation on $(VUT)^*$

defined as follows:

Suppose that $\alpha_1, \alpha_2, \ldots \alpha_m, \in (VUT)^*$, $m \geq 1$ and

$\alpha_1 \underset{G}{\Rightarrow} \alpha_2$, $\alpha_2 \underset{G}{\Rightarrow} \alpha_3$, $\ldots$ $\alpha_{m-1} \underset{G}{\Rightarrow} \alpha_m$

Then $\alpha_1 \overset{*}{\underset{G}{\Rightarrow}} \alpha_m$ or α_1 derives α_m in grammar G

- $\overset{*}{\underset{G}{\Rightarrow}}$ is reflexive, transitive closure of $\underset{G}{\Rightarrow}$
- $\alpha \overset{*}{\underset{G}{\Rightarrow}} \alpha$ for each string $\alpha \in (VUT)^*$

- When it is clear which grammar G is involved, we write $\Rightarrow$ instead of $\underset{G}{\Rightarrow}$ and $\overset{*}{\Rightarrow}$ instead of $\overset{*}{\underset{G}{\Rightarrow}}$
- $\alpha \overset{i}{\Rightarrow} \beta$ if α derives β by exactly i steps

- The language generated by G is

$$L(G) = \{w \mid w \in T^* \text{ and } S \overset{*}{\underset{G}{\Rightarrow}} w\}$$

i.e. a string is in $L(G)$ if

i. The string consists of solely of terminals
ii. The string can be derived from S

Context free language (CFL)

L is a context free language (CFL) if $L = L(G)$ for some CFG G

Sentential form

A string $\alpha \in (VUT)^*$ is called a sentential form if $S \overset{*}{\Rightarrow} \alpha$

Equivalent grammar

G_1, G_2 are equivalent if $L(G_1) = L(G_2)$

Example:

Consider CFG $G = (V, T, P, S)$ where

$V = \{S\}$, $T = \{a, b\}$, $P = \{S \to aSb, S \to ab\}$. Find $L(G)$

$$L(G) = \{w \mid w \in T^* \text{ and } S \overset{*}{\underset{G}{\Rightarrow}} w\}$$

$$S \Rightarrow aSb \Rightarrow aaSbb \Rightarrow a^3Sb^3 \Rightarrow \dots \Rightarrow a^{n-1}Sb^{n-1}$$

(applying $S \to aSb$ $n-1$ times)

$$\Rightarrow a^n b^n \quad \text{(applying } S \to ab\text{)}$$

Thus, $S \overset{*}{\Rightarrow} a^n b^n \in T^* \Rightarrow a^n b^n \in L(G)$ for $n \geq 1$

Now, we need to show that $L(G) \subseteq \{a^n b^n \mid n \geq 1\}$

~~Let $\alpha \in L(G)$~~.

Let $w \in L(G)$. w is a terminal string i.e. $w \in T^*$

The only way w can be derived from start symbol S is

$$S \overset{*}{\Rightarrow} a^{n-1}Sb^{n-1} \Rightarrow a^n b^n \quad \text{for some } n \geq 1$$

- $S \overset{*}{\Rightarrow} a^{n-1}Sb^{n-1}$: applying $S \to aSb$ few no. of times
  - ⇓ number of S's remains same in the sentential form
  - ⇓ i.e. $L(G) \subseteq \{a^n b^n \mid n \geq 1\}$
- $\Rightarrow a^n b^n$: one application of $S \to ab$
  - ⇓ number of S's is reduced by 1 in the sentential form
  - ⇓ no S remains in the resulting string

Thus

$$L(G) = \{a^n b^n \mid n \geq 1\}$$

Example

Find a CFG generating the language of even length palindromes.

i.e. $L = \{ww^R \mid w \in \Sigma^* = \{a,b\}^*\}$

$G = (\{S\}, \{a,b\}, P, S)$,

P is defined as $S \to aSa \mid bSb \mid \varepsilon$

Arguments:

i. ε is a palindrome

ii. a, b are palindromes

iii. if x is a palindromes, then axa, bxb are palindromes.

Example:

Construct a CFG generating

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

$G = (\{S\}, \{a,b,c\}, P, S)$

where P is defined as $S \to c \mid aSa \mid bSb$

Argument:

Any string in L is generated by recursion as follows:

i. $c \in L$

ii. if $x \in L$, then $wxw^T \in L$

Example

A CFG for the regular language corresponding to the regular expression 00^*11^*,

$$L = \{ 0^i 1^j \mid i, j > 0 \}$$

idea:

The language is the concatenation of two languages

- language of strings of 0's
- language of strings of 1's

$G = (\{S, C, D\}, \{0, 1\}, P, S)$

where P is defined as $\{ S \to CD,\ C \to 0C \mid 0,\ D \to 1D \mid 1 \}$

Example:

CFG that generates sentences as composed of noun and verb phrases.

$S \to NP\ VP$

$NP \to \text{the}\ N$

$VP \to V\ NP$

$V \to \text{sings} \mid \text{eats}$

$N \to \text{cat} \mid \text{song} \mid \text{bird}$

"the bird sings the song" → accepted

"the song eats the cat" → also accepted

- All legal sentences are generated, not just those that are meaningful — content-free

Example:

Complement of $\{0^i 1^j : i,j > 0\}$

idea:

partition the set of strings into three failure cases

1. strings not of the right form : somewhere there is a 1 followed by a 0
2. only zeroes
3. only ones

$G = (\{S, A, B, C, D\}, \{0,1\}, P, S)$

where P has the following production rules:

$S \to A|B|C$

$A \to D01D$ (produces all strings with a 0 and 1 out of order)

$D \to 0D|1D|\varepsilon$ (produces all strings)

$B \to 0B|0$ (produces strings of 0's)

$C \to 1C|1$ (produces strings of 1's)

e.g.

Test whether $w \in L(G)$ where

$w = 101001$

$S \Rightarrow A \Rightarrow D10D \Rightarrow \varepsilon 10D = 10D$

$\Rightarrow 101D$

$\Rightarrow 1010D$

$\Rightarrow 10100D$

$\Rightarrow 101001D$

$\Rightarrow 101001\varepsilon$

$\Rightarrow 101001$

$= w$

$\therefore w \in L(G) = \{w | w \in T^* \text{ and } S \overset{*}{\Rightarrow} w\}$

Example:

Find a CFG G for all binary strings with an even number of 0's

$S \to 1S \mid 0A0S \mid \varepsilon$

$A \to 1A \mid \varepsilon$

($1A$ produces strings of 1's.)

How to decompose such strings?

1. 1st symbol starts with 1, followed by even no. of 0's
2. 1st symbol starts with 0, then go to the next 0, followed even number of 0's

e.g. $101000 \in L(G)$ as

$S \Rightarrow 1S \Rightarrow 10A0S \Rightarrow 101A0S \Rightarrow 101\varepsilon 0S = 1010S$

$\Rightarrow 10100A0 \Rightarrow 10100\varepsilon 0 \Rightarrow 101000$

- A language can be more than one grammar

$S \to S \mid 0T \mid \varepsilon$, $T \to T \mid 0S$ } also generates all binary strings with even no. of 0's

idea:

We decompose all binary strings with an even number of 0's as follows:

1. 1st symbol is 0, followed by odd number of
2. We use T to generate all binary strings with odd number of zeroes.
3. We use S to generate all binary string with even number of zeroes.

Production rules: $S \to 1S \mid 0T \mid \varepsilon$

$T \to 1T \mid 0S$

Example:

Let $G = (\{S, C\}, \{a, b\}, P, S)$ where P consists of $S \to aCa$, $C \to aCa \mid b$. Find $L(G)$.

$S \Rightarrow aCa \Rightarrow aba \quad \therefore aba \in L(G)$

$S \Rightarrow aCa \Rightarrow aaCaa \Rightarrow \dots \Rightarrow a^n C a^n \Rightarrow a^n b a^n$

$\therefore a^n b a^n \in L(G), \; n \geq 1$

$\therefore \{a^n b a^n \mid n \geq 1\} \subseteq L(G)$ ——— (1)

Now, to prove $L(G) \subseteq \{a^n b^* a^n \mid n \geq 1\}$

Any string $w \in T^* = \{a, b\}^*$ will be in $L(G)$ if

$$S \overset{*}{\Rightarrow} w$$

The only S-production is $S \to aCa$

if we apply $C \to b$, we get $aba \in T^*$

$$S \overset{*}{\Rightarrow} aba$$

if we apply $C \to aCa$, once or several times

we get $S \overset{*}{\Rightarrow} a^n C a^n$

$\Rightarrow a^n b a^n \in T^*$

(one application of $C \to b$)

Thus any derivation is of the form

$$S \overset{*}{\Rightarrow} a^n b a^n, \quad n \geq 1$$

$\Rightarrow L(G) \subseteq \{a^n b a^n \mid n \geq 1\}$ ——— (2)

Thus from (1) and (2)

$$L(G) = \{a^n b a^n \mid n \geq 1\}$$

Example:

Let $G = (\{S, A_1\}, \{0,1,2\}, P, S)$ where P consists of

$S \to 0SA_1 2$, $S \to 012$, $2A_1 \to A_1 2$, $1A_1 \to 11$. Find $L(G)$

We have

$S \Rightarrow 012 \quad \therefore \quad 012 \in L(G)$

$S \Rightarrow 0\underline{S}A_1 2 \Rightarrow 0012\underline{A_1}2 \Rightarrow 001\underline{A_1}22 \Rightarrow 001122$

$S \Rightarrow 0SA_1 2 \Rightarrow 00SA_1 2A_1 2 \Rightarrow 000SA_1 2A_1 2A_1 2$

$S \to 0SA_1 2$ applied $(n-1)$ times

$\overset{*}{\Rightarrow} 0^{n-1} S (A_1 2)^{n-1}$

$\Rightarrow 0^{n-1} 012 (A_1 2)^{n-1}$

$\Rightarrow 0^n 1\ 2A_1 (A_1)^{n-2} 2^{n-1}$

$\Rightarrow 0^n 1\ A_1 2\ A_1 (A_1)^{n-3} 2^{n-1}$

$\Rightarrow 0^n 1\ A_1 A_1\ 2A_1 (A_1)^{n-4} 2^{n-1}$

$\vdots$

$\Rightarrow 0^n 1\ A_1^{n-1} 2^n$

$\Rightarrow 0^n 1A_1\ A_1^{n-2} 2^n$

$\Rightarrow 0^n 11\ A_1^{n-2} 2^n$

$\Rightarrow 0^n 111\ A_1^{n-3} 2^n$

$\vdots$

$\Rightarrow 0^n 1^n 2^n$

$\therefore \quad 0^n 1^n 2^n \in L(G)$

$\Rightarrow \quad \{0^n 1^n 2^n \mid n \geq 1\} \subseteq L(G)$

Then we need to show that $L(G) \subseteq \{0^n 1^n 2^n \mid n \geq 1\}$

Example

Let $G = (\{S, A_1, A_2\}, \{a, b\}, P, S)$

where P consists of

$S \to a A_1 A_2 a$, $A_1 \to ba A_1 A_2 b$

$A_2 \to A_1 ab$, $A_1 \to baa$

$bA_2 b \to abab$

Test whether $w = baabbabaaaabbaba$ is in $L(G)$.

$S \Rightarrow \underline{a A_1} A_2 a$

$\Rightarrow baa \underline{A_2} a$

$\Rightarrow baa \underline{A_1} aba$

$\Rightarrow baa\, ba \underline{A_1} A_2\, baba$

$\Rightarrow baabbaa \underline{A_2}\, baba$

$\Rightarrow baabba \underline{aA_1} abbaba$

$\Rightarrow baabbabaaaabbaba = w$

$\therefore w \in L(G)$

Example

If the grammar G is given by the productions $S \to aSa \mid bSb \mid aa \mid bb \mid \varepsilon$, show that

i) $L(G)$ has no strings of odd length

ii) any string in $L(G)$ is of length $2n$, $n \geq 0$

iii) the number of strings of length $2n$ is 2^n

On application of each production (except $S \to \varepsilon$), a variable is replaced by 2 terminals and atmost one variable

Thus each step in any derivation increases the number of of terminals by 2 except that involving $S \to \varepsilon$.

Which explains (i) and (ii)

Now any string w of length $2n$ is of the form

$$a_1 a_2 \ldots a_n a_n a_{n-1} \ldots a_1$$

where each a_i is either a or b

$\Rightarrow$ number of $2n$-bit strings in $L(G)$ is 2^n

Example

Construct a context free grammar G generating all integers (with sign)

Let $G = (V, T, P, S)$ where

$V = \{ S, \langle sign \rangle, \langle digit \rangle, \langle integer \rangle \}$

$T = \{ 0, 1, \ldots, 9, +, - \}$

and P consists of

$S \rightarrow \langle sign \rangle \langle integer \rangle$

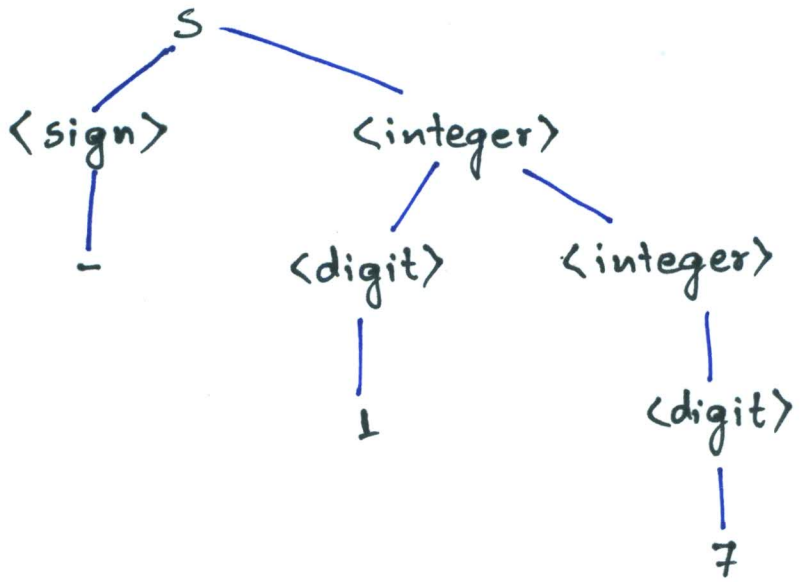
$\langle sign \rangle \rightarrow + \mid -$

$\langle integer \rangle \rightarrow \langle digit \rangle \langle integer \rangle \mid \langle digit \rangle$

$\langle digit \rangle \rightarrow 0 \mid 1 \mid 2 \mid \cdots \mid 9$

Then $L(G)$: the set of all integers.

e.g. derivation of -17



Derivation Trees / Parse Trees

- $G = (V, T, P, S)$ be a CFG. A tree is a **derivation tree or parse tree** for G if:

  i. Every vertex has a label, which is a symbol of
  $$V \cup T \cup \{\varepsilon\}$$

  ii. The label of the root is S

  iii. If a vertex is interior and has label A, then A must be in V

  iv. If vertex n has label A and vertices $n_1, n_2, \ldots, n_k$ are sons of vertex n, in order from the left with labels $X_1, X_2, \ldots, X_k$ respectively, then
  $$A \to X_1 X_2 \ldots X_k$$
  must be a production in P

  v. If vertex n has label ε, then n is a leaf and is the only son of its father

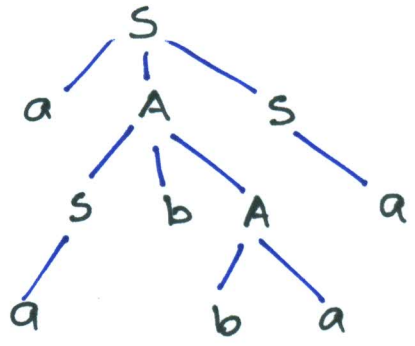
Example:

$G = (\{S, A\}, \{a, b\}, P, S)$

where consists of

$S \to aAS \mid a$

$A \to SbA \mid SS \mid ba$



A natural description of the sentential form

yield of the derivation tree $\longrightarrow$ aabbaa

- The yield of a derivation tree is the concatenation of the labels of the leaves without repetition in the left-to-right ordering.

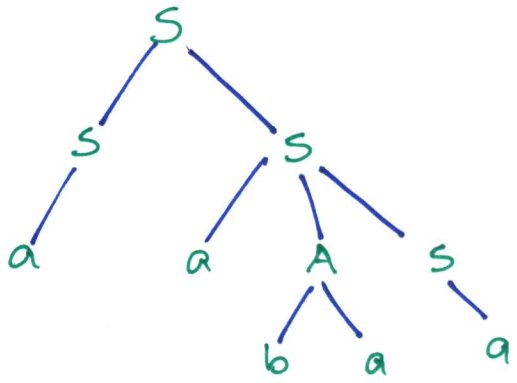
Example:

$G = (\{S, A\}, \{a, b\}, P, S)$

where P consists of

$S \rightarrow aAS \mid a \mid SS$

$A \rightarrow SbA \mid ba$



A parse tree showing the derivation

$S \overset{*}{\Rightarrow} aabaa$

- The yield of the derivation tree is a sentential form in G. i.e. yield $\in (V \cup T)^*$

Example:

$G = (\{E, I\}, T, P, E)$

where P consists of

$E \rightarrow I \mid E+E \mid E*E \mid (E)$

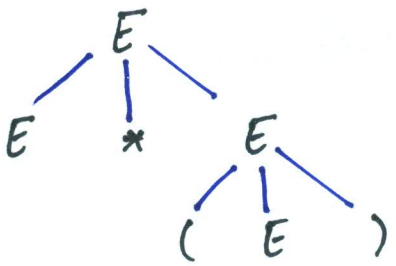
$I \rightarrow a \mid b \mid Ia \mid Ib \mid I0 \mid I1$

and

$T = \{+, *, (,), a, b, 0, 1\}$

(i) A parse tree showing the derivation

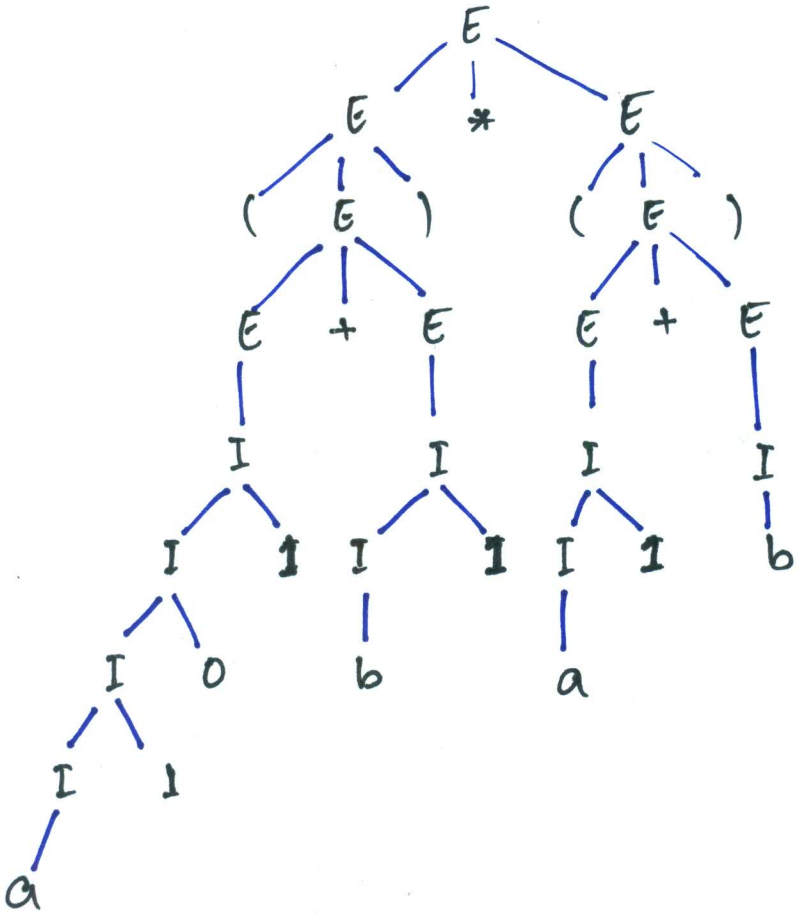
$E \overset{*}{\Rightarrow} E * (E)$



yield $\in (VUT)^*$

A parse tree showing the derivation

$E \overset{*}{\Rightarrow} (a101 + b1) * (a1 + b)$



yield $\in T^*$

Relationship between derivation trees and derivations

Theorem:

Let $G = (V, T, P, S)$ be a context-free grammar. Then $S \overset{*}{\Rightarrow} \alpha$ iff there is a derivation tree in grammar G with yield α

Proof:

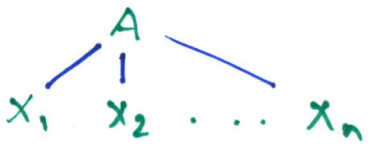
A-tree: A subtree of a derivation tree is called an A-tree if the label of its root is A

S-tree: derivation tree when S is the start symbol

We prove that $A \overset{*}{\Rightarrow} \alpha$ iff there is an A-tree with yield α.

(if) Suppose α is the yield of an A-tree. We prove by induction on K = the no. of internal vertices in the tree, that $A \overset{*}{\Rightarrow} \alpha$

Basis: $K=1$, i.e. only one internal vertex



$\therefore \alpha = X_1 X_2 \ldots X_n$ and

$A \to \alpha$ must be a production rule in G by definition of a derivative tree

Induction step:

Let us assume the result for all derivation trees with $\leq K-1$ internal vertices for some $K > 1$

Let α be the yield of an A-tree with K- interior vertices for some $K > 1$

Let the sons of the root A have labels $X_1, X_2, \ldots, X_n$ in order from the left

Then $A \to X_1 X_2 \ldots X_n$ is in P ($n \geq 1$ is any integer)

- $X_1, X_2, \ldots X_n$ cannot all be leaves (as $K > 1$)
- if i-th son of A is not a leaf, then it is the root of a subtree and X_i must be a variable.

X_i - tree with yield α_i say.

if i^{th} son of A is a leaf, let $\alpha_i = X_i$

Then $\alpha = \alpha_1 \alpha_2 \ldots \alpha_n$ as one can easily see that if $j < i$, then vertex j and all its descendants are to the left of vertex i and all its descendants.

Now subtrees must have fewer interior nodes than its tree does, unless the subtree is the entire tree.

$\therefore$ We have for each X_i-tree with yield α_i

$X_i \overset{*}{\Rightarrow} \alpha_i$ by induction as X_i- tree is not the entire tree.

Also, $X_i \Rightarrow \alpha_i$ if $X_i = \alpha_i$

$\therefore A \Rightarrow X_1 X_2 \ldots X_n \Rightarrow \alpha_1 X_2 \cdots X_n$

$\overset{*}{\Rightarrow} \alpha_1 \alpha_2 \cdots X_n \Rightarrow \ldots$

$\overset{*}{\Rightarrow} \alpha_1 \alpha_2 \ldots \alpha_n = \alpha$

Thus $A \overset{*}{\Rightarrow} \alpha$

(Only if)

Let $A \overset{*}{\Rightarrow} \alpha$

We must show that there is an A-tree with yield α

We do it by induction on K = the number of steps in $A \overset{*}{\Rightarrow} \alpha$

(Base)

K = 1

if $A \overset{*}{\Rightarrow} \alpha$ in single step, then $A \to \alpha$ is in.

Induction:

Assume the result is true for derivatives in $< K$ steps

Let $A \overset{K}{\Rightarrow} \alpha$

This can be splitted as $\underbrace{A \Rightarrow X_1 X_2 \ldots X_n}_{\text{implies } A \to X_1 X_2 \ldots X_n \text{ is in P}} \overset{K-1}{\Rightarrow} \alpha$

- In $X_1 X_2 \ldots X_n \overset{K-1}{\Rightarrow} \alpha$ either

  i. X_i is not changed throughout the derivation

  or ii. X_i is changed in some subsequent step.

Let α_i be the substring derived from X_i

Then $X_i = \alpha_i$ is in (i) and $X_i \overset{*}{\Rightarrow} \alpha_i$ is in (ii)

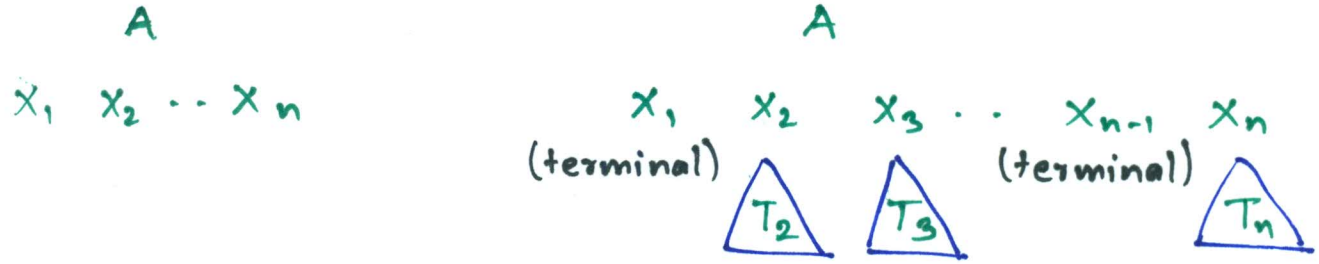
- As G is context free, in every step of derivation $X_1 X_2 \dots X_n \overset{*}{\Rightarrow} \alpha$, we replace a single variable by a string

  As $\alpha_1, \alpha_2, \dots \alpha_n$ accounts for all the symbols in α we have $\alpha = \alpha_1 \alpha_2 \dots \alpha_n$

- If X_i is a variable, then the derivation $X_i \overset{*}{\Rightarrow} \alpha_i$ has $< K$ steps as the entire derivation $A \overset{*}{\Rightarrow} \alpha$ takes K steps and the first step is surely not a part of $X_i \overset{*}{\Rightarrow} \alpha_i$

Hence by induction, for each X_i that is a variable, there is an X_i-tree T_i with yield α_i

Now, we construct an <u>A-tree</u> with <u>yield α</u> as follows:



First construct an A-tree with n leaves labeled $X_1, X_2, \dots X_n$ and no other vertex X

Replace each vertex with label X_i by tree T_i where X_i is non-terminal.

If X_i is a terminal vertex, no replacement is made.

The yield of the resulting A-tree is $\alpha_1 \alpha_2 \dots \alpha_n = \alpha$

Note:

The derivation tree does not specify the order in which we apply the productions for getting α.

So, the same derivation tree can induce several derivations (with the same yield)

Remark:

Let $A \overset{*}{\Rightarrow} w$ where $w \in T^*$ (a terminal string)

and $A \Rightarrow A_1 A_2 \cdots A_n$

Then we can write $w = w_1 w_2 \cdots w_n$

so that $A_i \overset{*}{\Rightarrow} w_i$