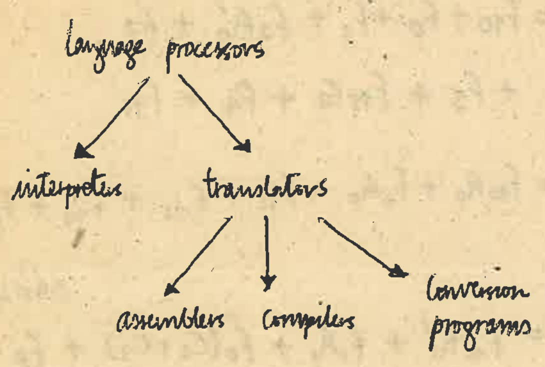
EE442 240282

Assemble - and - go
Compile - and - go

open subroutine = macro
closed subroutine = subroutine



parse

formal system : An uninterpreted calculus.
- alphabet
- axioms
- rules of inference

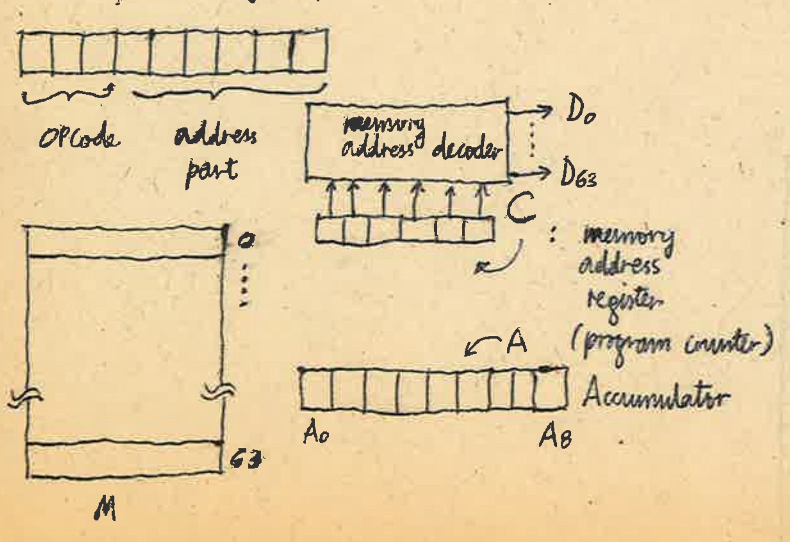
Machine Structure 250282

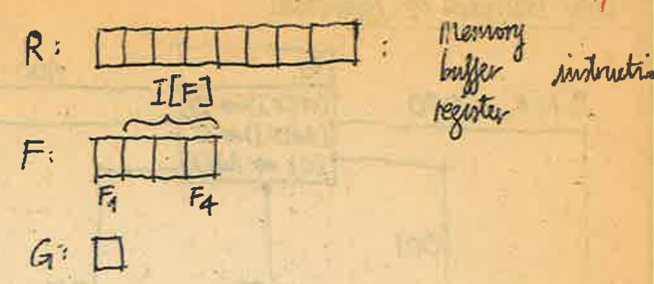
Symbols : (Example from CHU Digital Computer Design Fundamentals)

( ) : "contents of"
[ ] : "part of"
< > : "as addressed by"
$\Rightarrow$: "transfer into"

A very simple M/C

9 bit Word length
2's complement system





Instructions :

ADD	000UVWXYZ	add (M<UVXWYZ>) to (A), leave the result in A
SUB	001UVWXYZ	Subtract " from (A), "
JPN	010UVWXYZ	if $A_0 = 1$ jump to location UVWXYZ
STR	011UVWXYZ	(A) $\Rightarrow$ M<UVWXYZ>
JMP	100UVWXYZ	Jump to location UVWXYZ
SHR	1011000YZ	
SHL	1010100YZ	
CLR	1010010YZ	Clear A
STP	1010001YZ	STOP

030382

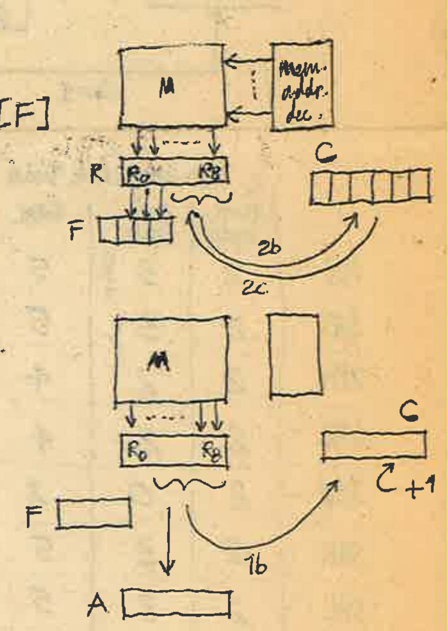
Instruction cycle

1) $(M<C>) \Rightarrow R$
2) a) $(Op[R]) \Rightarrow I[F]$
b) $(Adr[R]) \Rightarrow C$
c) $(C) \Rightarrow Ad[R]$

Execution cycle (ADD)

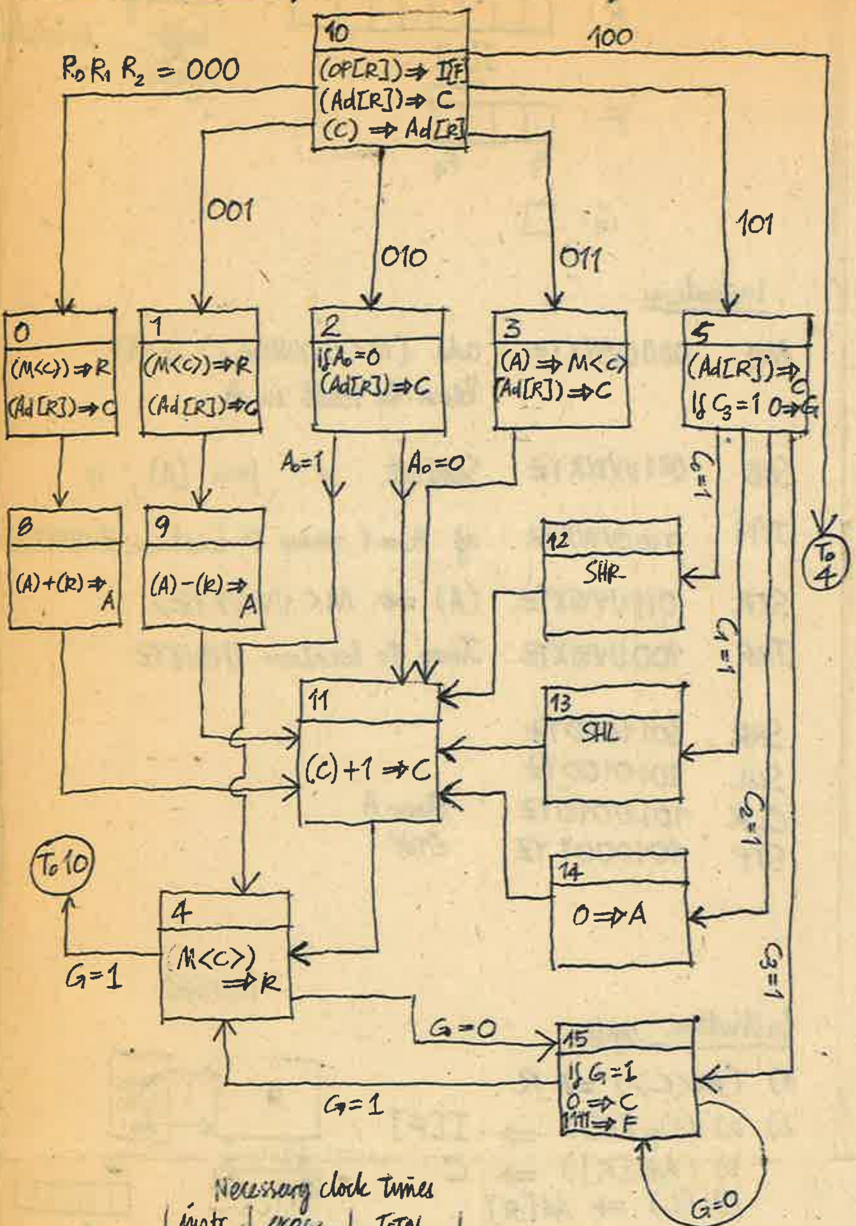
1) a) $(M<C>) \Rightarrow R$
b) $(Ad[R]) \Rightarrow C$

2) $(A) + (R) \Rightarrow A$
3) $(C) + 1 \Rightarrow C$



F_1	F_2	F_3	F_4	STATE	
0	0	0	0	0	ADD
0	0	0	1	1	SUB
0	0	1	0	2	JPN
0	0	1	1	3	STR
0	1	0	0	4	JMP
0	1	0	1	5	SHR, SHL, CLR, STP
0	1	1	0	6	—
0	1	1	1	7	—
1	0	0	0	8	add
1	0	0	1	9	subtract
1	0	1	0	10	instruction cycle
1	0	1	1	11	increment C
1	1	0	0	12	SHR
1	1	0	1	13	SHL
1	1	1	0	14	CLR
1	1	1	1	15	STP

STATE DIAGRAM OF COMPUTER



Necessary clock times

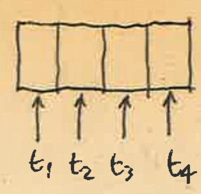
	instr. cycle	exec. cycle	TOTAL
ADD	2	3	5
SUB	2	3	5
JPN	2	2	4
STR	2	2	4
JMP	2	0	2
SHR	2	3	5
SHL	2	3	5
CLR	2	3	5
STP	2	2	4

4 10

All the variables are : $R_0 R_1 R_2 C_0 C_1 C_2 C_3 A_0 G$

PS $F_1F_2F_3F_4$		Next State ($\bar F_1 \bar F_2 \bar F_3 \bar F_4$) $R_1R_2R_3$ 000	001	010	011	100	101	110	110
1010	10	0 0000	1 0001	2 0010	3 0011	4 0100	5 0101	0110	0111
0000	0				8 1000				
0001	1				9				
0010	2	If A_0=0 11			If A_0=1 4				
0011	3				11				
0101	5	If C_0=1 12		C_1=1	13	C_2=1 14		C_3=1 15	
1000	8				11				
1001	9				11				
1100	12				11				
1101	13				11				
1110	14				11				
1111	15	If G=0 15				G=1 4			
0100	4	If G=0 15				G=1 10			
1011	11				4				
0110	6				–				
0111	7				–				

DESIGN OF F-REGISTER



(Decide to use T-FF's)

$$f_{10} = F_1 F_2' F_3 F_4'$$

$$t_1 = f_{10} + f_0 + f_1 + f_2 A_0' + f_3 + f_5 + f_{15} G + f_4 + f_{11}$$

$$t_2 = f_{10} R_0 + f_2 A_0 + f_{12} + f_{13} + f_{14} + f_4 G + f_{11}$$

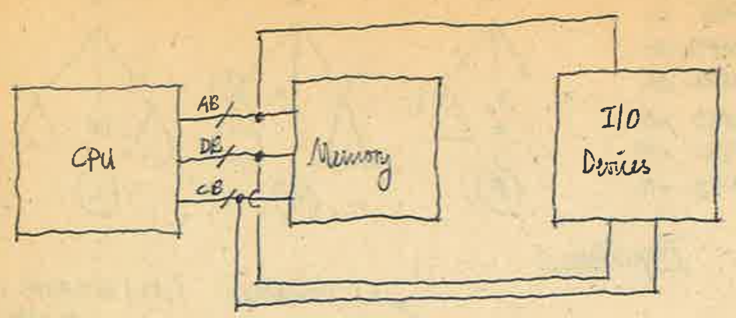
040382

$$t_3 = f_{10} R_1' + f_2 A_0 + f_5 (C_2 + C_3) + f_8 + f_9 + f_{12} + f_{13} + f_{15} G + f_4 + f_{11}$$

$$t_4 = f_{10} R_2 + f_2 A_0' + f_5 (C_0 + C_2) + f_8 + f_{12} + f_{14} + f_{15} G + f_4 G' + f_{11}$$

EE 402
MT1 17 NISAN
MT2 22 MAYIS

I/O: -442-



— Memory mapped I/O RARELY USED

— I/O mapped input/output

Trying to output certain data;

```
MVI A, FFH
OUT 01H
```

inputting :

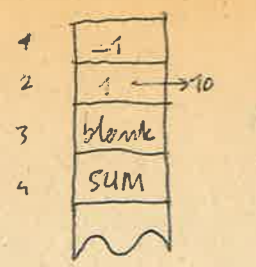
```
        :
BACK :  IN  01H
        ANI FFH
        JZ  BACK
        :
```

<u>8085 interrupt inputs</u>

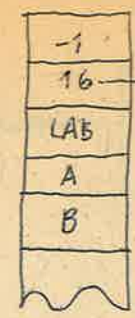
INTR
RST 5.5
RST 6.5 } maskable
RST 7.5
TRAP

SIM: Set interrupt mask.

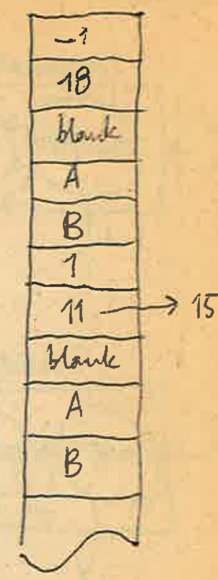
Expand DEFINE:

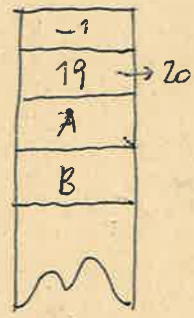


Expand FIND



Expand SUM





Continue expanding FIND

<u>EE442</u> : <u>Sample Program</u>: 070482

```
    MACRO $
    DEFINE &SUB
    SR     5,5
    MACRO
    &SUB   &Y, &Z
    L      2, &Y
    A      2, &Z
    BAL    9, &SUB
    MEND
    XR     7,7
    MEND
    DEFINE SUM
    MACRO
&LB FIND &X, &Y
    A    5, &X
    SUM  &X, &Y
&LB NOPR
    MEND
    NOP
LAB FIND A,B
    ---
    ---
A   DS  F
B   DS  F
```

MNT

```
1 DEFINE 1
2 SUM    11
3 FIND   16
```

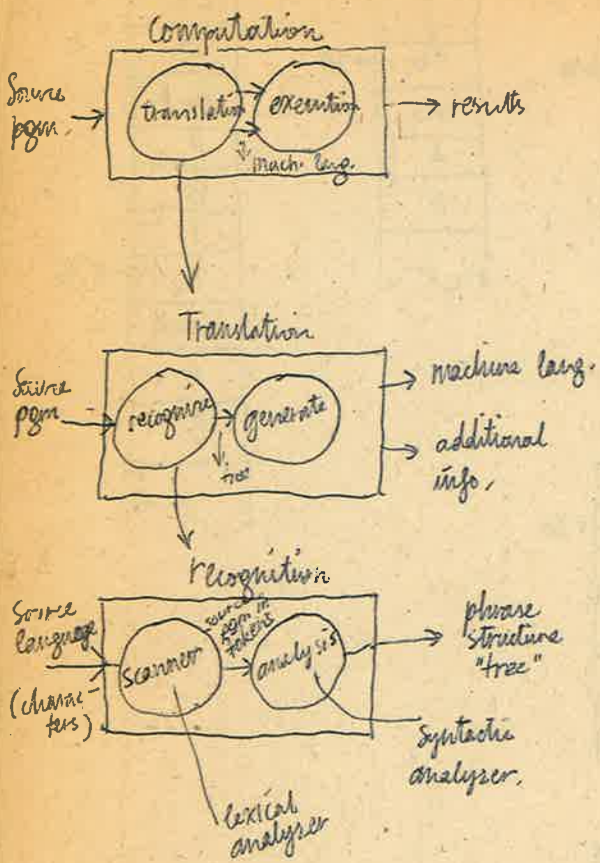
```
1  DEFINE  &SUB
2  SR      5,5
3  MACRO
4  #1      &Y, &Z
5  L       2, &Y
6  A       2, &Z
7  BAL     9, #1
8  MEND
9  XR      7,7
10 MEND
11 SUM     &Y, &Z
12 L       2, #1
13 A       2, #2
14 BAL     9, SUM
15 MEND
16 &LAB FIND &X, &Y
17       A    5, #1
18       SUM  #1, #2
19 #0    NOPR
20       MEND
```

270
850
135
216
2295.0
40
400
2335
2500

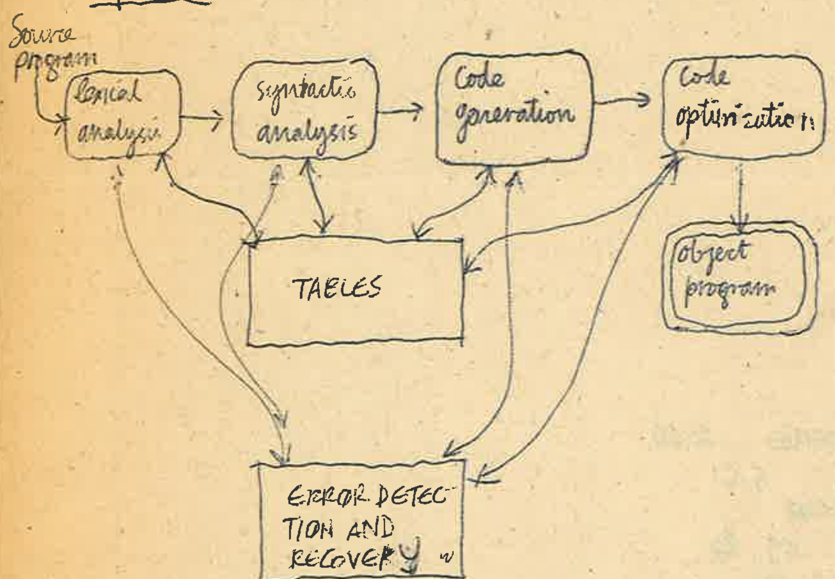
Page 143

Do

Compilers



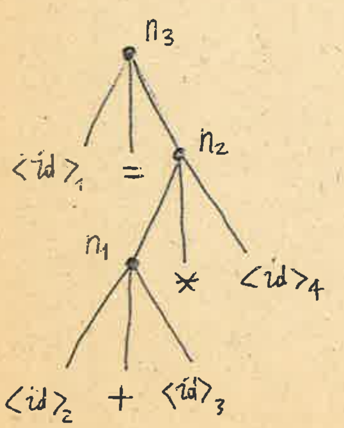
Sequence:



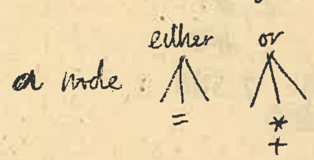
example:

VALUE = (DEGRE + PHASE) * 0.27

$\langle id \rangle_1 = (\langle id \rangle_2 + \langle id \rangle_3) * \langle id \rangle_4$ ← output of lexical analyzer

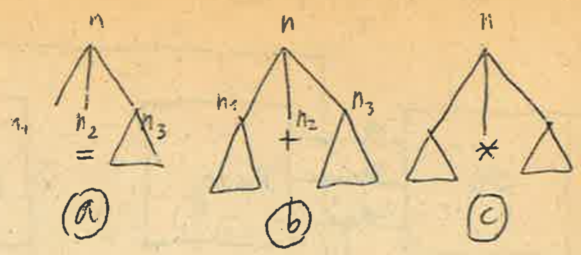


← output of syntactic analyzer



Translation example
About a simple m/c

```
LOAD  m
ADD   m
MPY   m
STORE m
LOAD  =m
ADD   =m
MPY   =m
```



algorithm:

for leaves:
(1). $\langle id \rangle$; a) Variable : C(n) = name of variable
b) Constant : C(n) = "=k"
(2) +, =, * C(n) : empty

(3) type (a) C(n) = 'LOAD' C(n_3) '; STORE' C(n_1)

(4) type (b) C(n) = C(n_3) '; STORE $' l(n)
'; LOAD' C(n_1) '; ADD $' l(n)

(5) type (c) C(n) = ...
'MPY $' l(n)

VALUE = (DEGRE+PHASE) * 0.27

$\langle id \rangle_1 = (\langle id \rangle_2 + \langle id \rangle_3) * \langle id \rangle_4$

Note : ; denotes "pass to the new line"

pgm for n_1

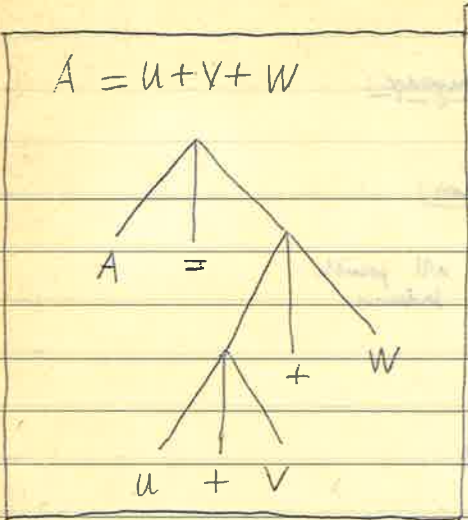
```
      PHASE
STORE $1
LOAD  DEGREE
ADD   $1
```

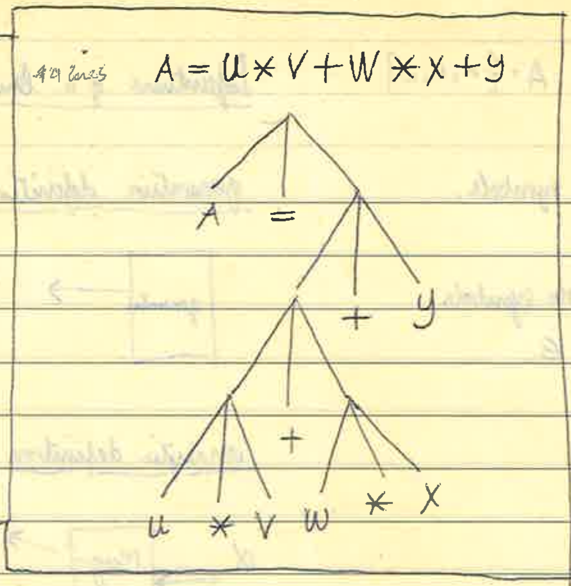
pgm for n_2

```
      =0.27
STORE $2
LOAD  PHASE
STORE $1
LOAD  DEGRE
ADD   $1
MPY   $2
```

n_3

```
LOAD  =0.27
STORE $2
LOAD  PHASE
STORE $1
LOAD  DEGRE
ADD   $1
MPY   $2
STORE VALUE
```





Code optimization: (Simplifying the program)

LOAD A } instead of ⇒ ① { LOAD B
ADD B } { ADD A

due to commutativity of addition

also there must be no transfer to that part of program.

LOAD A } ⇒ ② { LOAD B
MPY B } { MPY A

STORE A } ⇒ ③ ~~X~~
LOAD A }

with the condition that we will not use the value of A later.

LOAD A } ⇒ ④ ~~X~~
STORE B }

~~X~~ A

```
LOAD   =0.27
STORE  $2
LOAD   PHASE
STORE  $1
LOAD   DEGRE  ]⇒1   STORE $1  ] X
ADD    $1     ]     LOAD  $1  ]
MPY    $2           ADD   DEGRE
STORE  VALUE
```

```
LOAD   =0.27  ]⇒4  X
STORE  $2     ]
LOAD   PHASE
ADD    DEGRE
MPY    $2
STORE  VALUE
```

```
LOAD   PHASE
ADD    DEGRE
MPY    =0.27
STORE  VALUE
```

OPTIMIZED PROGRAM

CHOMSKY

150482

alphabet : Group of symbols. $A = \{a, b, c\}$

symbol :

string : Any finite sequence of symbols.

length : $\ell(\cdot)$ $\ell(abcaa) = 5$

empty string : string, containing no symbols.
denoted by λ or ϵ
$\ell(\lambda) = 0$

Concatenation : $\alpha\beta = abbbc$
(catenation) $\alpha = ab$
$\beta = bbc$

product : A, B : sets "cartesian product"

$AB = \{ ab \mid a \in A, b \in B \}$

$A = \{a, b, c\}$
$B = \{0, 1\}$
$AB = \{ a0, a1, b0, b1, c0, c1 \}$

$A = \{ab, c, caa\}$
$B = \{a, ac\}$
$AB = \{aba, abac, ca, cac, caaa, caaac\}$

$A^1 = A$ $A\{\lambda\} = A$
$A^0 = \{\lambda\}$ $A\varphi = \varphi$

Closure

$A^* = A^0 \cup A^1 \cup A^2 \cup \ldots$

$\{a, b, c\}^* = \{\lambda, a, b, c, aa, ab, ac, ba, bb, bc, ca, cb, cc, aaa, aab, aac, \ldots \ldots \rightarrow \}$

all strings that made by using elements of A

language L : (Given A) $L \subset A^*$

$L_1 = \{01, 001, 0001, \ldots\}$
$L_1 = \{0^n 1 \mid n \geq 1\}$

Definitions of a language:

generative definition:

generator → all possible sentences

analytic definition

α → recognizer → $\alpha \in L$
→ $\alpha \notin L$

grammar:

Rule #

① <Sentence> → <NP><VP> (noun phrase) (verb phrase)
② <NP> → <article><noun>
③ <VP> → <Verb>
④ <VP> → <verb><adverb>

$\underbrace{\text{The Student}}_{\text{noun phrase}}\ \underbrace{\text{studies hard}}_{\text{verb phrase}}$

⑤ <article> → the
⑥ <noun> → student
⑦ <verb> → studies
⑧ <adverb> → hard
⑨ <adverb> → slowly.

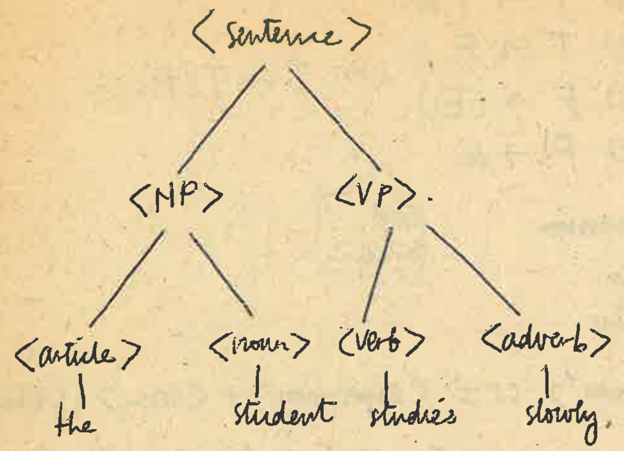
parantheses shows grammatical classes.

HW#1 :	CH3	PR 9a	CH4	2ab
		13		3
		14		4
		15a		5
		19		
		20		

Haftaya Persembe

$\boxed{\text{26 Mayıs MT 1 8 40}}$

<sentence> → <NP><VP>
<NP> → <article><noun>



<article> → the

Grammar : (N, T, Σ, P) (terms with brackets)

N : the set of nonterminals (<...>)
T : the set of terminals (the, student, ...)
$\Sigma \in N$ start symbol, sentence symbol. (<sentence>)

P : Set of of productions (rewriting rules, rules)
(<sentence> → <NP><VP>)

notation:

A, B, C ... : nonterminals
a, b, c ..., 0, 1, ... : terminals
α, β, γ ... : strings → $\alpha \in (N \cup T)^* - \lambda$

G_1 :

$N = \{\Sigma, A\}$
$T = \{0, 1\}$
$P = \Sigma \to A$ "we have the freedom of erasing A and writing 0A1 instead."
$A \to 0A1$
$A \to 01$

direct derivation

$\mu \Rightarrow \gamma$ "μ directly derives γ"
or
"γ is directly derived by ~~α~~"

$\Sigma \Rightarrow A \Rightarrow 01$
$\Sigma \Rightarrow A \Rightarrow 0A1 \Rightarrow 0011$
$\Sigma \Rightarrow A \Rightarrow 0A1 \Rightarrow 00A11 \Rightarrow 000111$

derivation

$\mu \stackrel{*}{\Rightarrow} \gamma$ "μ derives γ, if there is a chain of direct derivations"
($\mu \Rightarrow \dots \Rightarrow \dots \Rightarrow \gamma$)

ex: $\Sigma \stackrel{*}{\Rightarrow} A$
$\Sigma \stackrel{*}{\Rightarrow} 00A11$

Language generated by grammar G : "L(G)"

$$L(G) = \{ w \in T^* \mid \Sigma \stackrel{*}{\Rightarrow} w \}$$

example : $L(G_1) = \{ 0^n 1^n \mid n \geq 1 \}$

G_2 : $N = \{\Sigma, A, B\}$
$T = \{0, 1\}$

P : $\Sigma \to AB$
$A \to 0A$
$A \to 0$
$B \to 1B$
$B \to 1$

$\Sigma \Rightarrow \underline{AB} \Rightarrow 0\underline{B} \Rightarrow 01$
$\Sigma \Rightarrow \underline{AB} \Rightarrow 0AB \Rightarrow 00B \Rightarrow 001$

"a terminal string is called as a sentence"

$\therefore L(G_2) = \{ 0^n 1^m \mid n \geq 1, m \geq 1 \}$

Classification of Grammars (Due to N. Chomsky)
(N, T, Σ, P)

type 0 : Phrase structure grammars.
unrestricted grammars $\alpha \to \beta$
(too general, so we need some restrictions) $\alpha \in (N \cup T)^* - \lambda$
$\beta \in (N \cup T)^*$

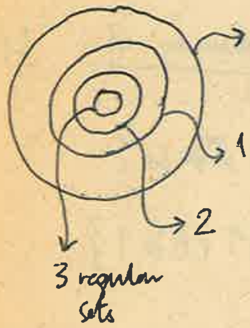
type 1 "context sensitive grammars":
$\alpha \to \beta$ $l(\alpha) \leq l(\beta)$

type 2 : $\alpha \to \beta$ $l(\alpha) \leq l(\beta)$
$\alpha \in N$
"Context free grammars"

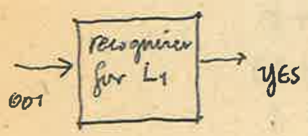
type 3 : "right linear"; (regular grammar grammars) $\alpha \to \beta$
$l(\alpha) \leq l(\beta)$
$\alpha \in N$

"left linear grammar": $A \to Ba$
$A \to a$

Definition: A language is $\left\{ \begin{array}{l} \text{type } 0 \\ 1 \\ 2 \\ 3 \end{array} \right\}$ if there is a grammar $\left\{ \begin{array}{l} \text{type } 0 \\ 1 \\ 2 \\ 3 \end{array} \right\}$ generating it



Recognizers for type 0 grammars : Turing machines
" " " 1 " : linear bounded automata
" " " 2 " : push-down automata
" " " 3 " : finite state machines



Type 0 greatly differs from the other ones :
Type 0 : contracting grammars : derived sentence may be shorter.

BNF :

$\longrightarrow$ $::=$

A, B, C <....>

G_2 written in BNF:

$\Sigma \to AB$ <Σ> ::= <A><B>

$\left.\begin{array}{l} A \to 0A \\ A \to 0 \end{array}\right\}$ <A> ::= 0<A> | 0

$\left.\begin{array}{l} B \to 1B \\ B \to 1 \end{array}\right\}$ <B> ::= 1<B> | 1

Meta language : A language is used for defining a language.

Example : G_5 $N = \{E, T, F\}$ (E used for start symbol)

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

P : 1) $E \to E+T$
2) $E \to T$ $\longrightarrow$ $E \to E-T$
3) $T \to T*F$
4) $T \to F$
5) $F \to (E)$ $\longrightarrow$ $T \to T/F$
6) $F \to a$

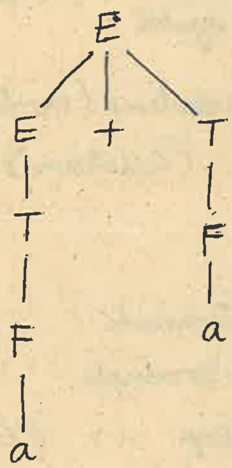
E expression
T term
F factor

BNF notation

<expression> ::= <expression> + <term> | <term>

"This Grammar generates all well formed arithmetic expressions"

$E \underset{1}{\Rightarrow} \underline{E}+T \underset{2}{\Rightarrow} \underline{T}+T \underset{4}{\Rightarrow} \underline{F}+T \underset{6}{\Rightarrow} a+\underline{T} \underset{4}{\Rightarrow} a+\underline{F} \underset{6}{\Rightarrow} a+a$

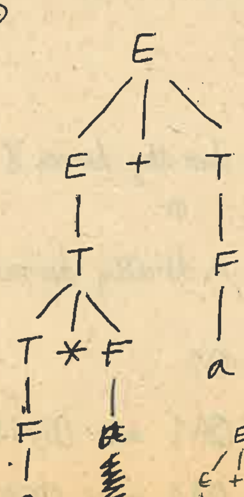


leftmost derivation
rightmost derivation
haphazard derivation

← There is only one tree to obtain a+a in this grammar.
"This is an unambiguous grammar"

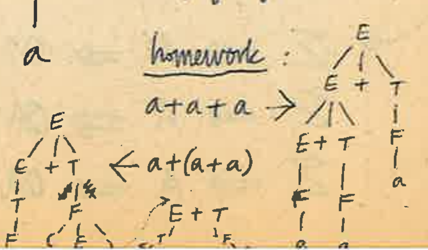
obtaining $a*a+a$

$E \underset{1}{\Rightarrow} E+T \underset{2}{\Rightarrow} \underline{T}+T \underset{3}{\Rightarrow} \underline{T}*F+T \underset{4}{\Rightarrow} \underline{F}*F+T \underset{6}{\Rightarrow} a*F+T \underset{6}{\Rightarrow} a*a+T \underset{4}{\Rightarrow} a*a+F \Rightarrow a*a+a$



ambiguity : If there is more than one tree for a string, there is ambiguity.

homework :
a+a+a →
← a+(a+a)



Simpler grammar to generate arithmetic expressions;

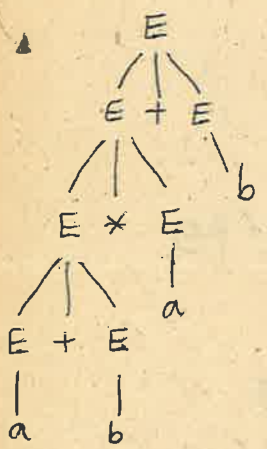
$E \to E+E$
$E \to E*E$
$E \to a|b$
} drawback: this grammar is ambiguous.

$a+b*a+b$

leftmost derivation - $E \Rightarrow E+E \Rightarrow E*E+E \Rightarrow E+E*E+E$
$\Rightarrow a+E*E+E \Rightarrow a+b*E+E \Rightarrow$
$a+b*a+E \Rightarrow a+b*a+b$

parse: Sequence of numbers in which the rules are used, denoted.

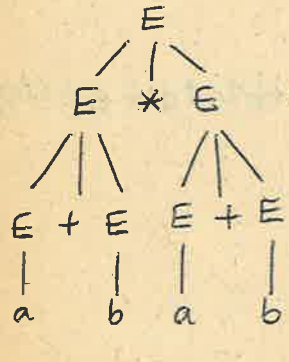
"If there are more than one leftmost derivations for a given sentence, this sentence and also this grammar is ambiguous."



$\equiv ((a+b)*a)+b$

another leftmost derivation of $a+b*a+b$:

$E \Rightarrow E*E \Rightarrow E+E*E \Rightarrow a+E*E$
$\Rightarrow a+b*E \Rightarrow a+b*E+E \Rightarrow$
$a+b*a+E \Rightarrow a+b*a+b$



$\equiv (a+b)*(a+b)$

"for every leftmost derivation there is a unique tree"

infix notation $m+m$ m/m

Jan Łukasiewicz : Polish expressions / notations

$+mm$: prefix notation

$mm+$: suffix notation (reverse polish)

reverse polish

$E \to EEO \mid L$ O: operator
$L \to a|b|c|\ldots|z$ unambiguous
$O \to +|*|-|/|\uparrow|,\ldots$

$a-b$ $\cancel{*b}$ power $a\uparrow b = a^b$

infix	reverse polish
a	a
a+b	ab+
a−b	ab−
a+b*c	abc*+
(a+b)*c	ab+c*
a+b+c	abc++ / ab+c+

Order of variables never change.
order of variables change.

$(a+b\uparrow c\uparrow d)*(e+f/g) \equiv abcd\uparrow\uparrow+efg/+*$

$\underbrace{a}\ \underbrace{bcd\uparrow\uparrow+}\ \underbrace{efg/+}*$

rank of a reverse polish expression:

r(variable) = 1
r(operator) = 1 − n

abc − −
1 1 1 −1 −1 = 1

theorem i) rank of a reverse polish expression is 1.

ii) any head of a reverse polish expression has rank greater or equal to 1.

abc − − a −

heads.

example:

ABC * + PQ/7 − +

A = 10
B = 2
C = 3
P = 20
Q = 5

how to evaluate?

↳ start from left, find first operator:

$A\underbrace{\boxed{BC*}}_{6}+PQ/7-+$

↳ $\underbrace{\boxed{A6+}}_{16}PQ/7-+$

↳ $16\underbrace{\boxed{PQ/}}_{4}7-+$

↳ 1647 − +

↳ 16 (−3) +

↳ 13 · II ·

bad thing:
we have destroyed the original notation.

280482

ABC * + PQ/7 − +

A = 10, B = 2, C = 3, P = 20, Q = 5

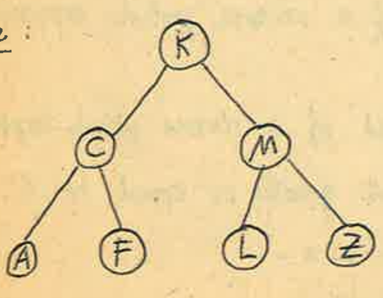
Not to destroy polish expression, we use STACK.

(processing)
traversing the trees:

	left-right	right-left
prefix	P,L,R	P,R,L
infix	L,P,R	R,P,L
postfix	L,R,P	R,L,P

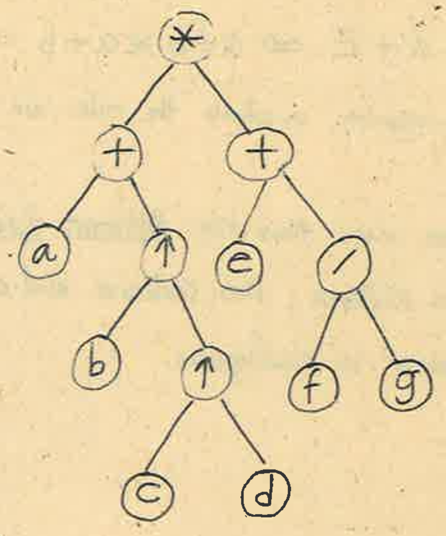
L: traverse the left branch.
P: traverse the node
R: traverse the right branch

example:



	L→R	R→L
prefix	KCAFMLZ	KMZLCFA
infix	ACFKLMZ	ZMLKFCA
postfix	AFCLZMK	ZLMFACK

example:



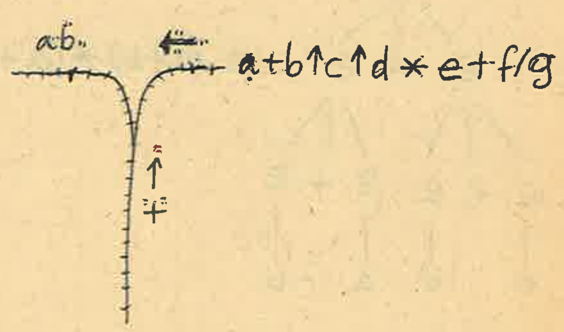
preorder (P,L,R)
polish   * + a ↑ b ↑ c d + e / f g

inorder (L,P,R)
infix   a + b ↑ c ↑ d * e + f/g

postorder (L,R,P)
rev. polish

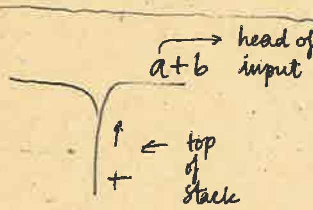
a b c d ↑ ↑ + e f g / + *

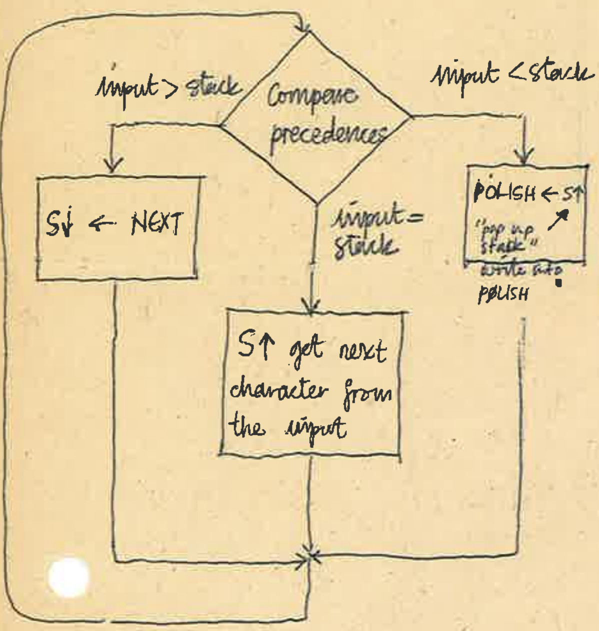
Algorithm for infix → reverse polish



algorithm:

op	head of input precedence f	top of stack precedence g	rank r
+ −	1	2	−1
* /	3	4	−1
↑	6	5	−1
variables	7	8	1
(	9	0	—
)	0	—	—

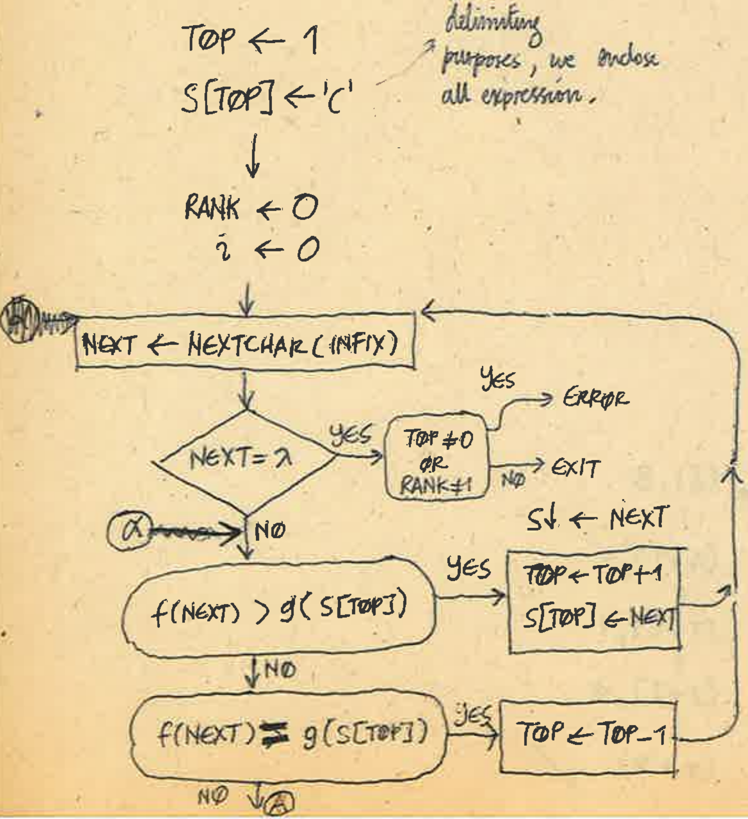


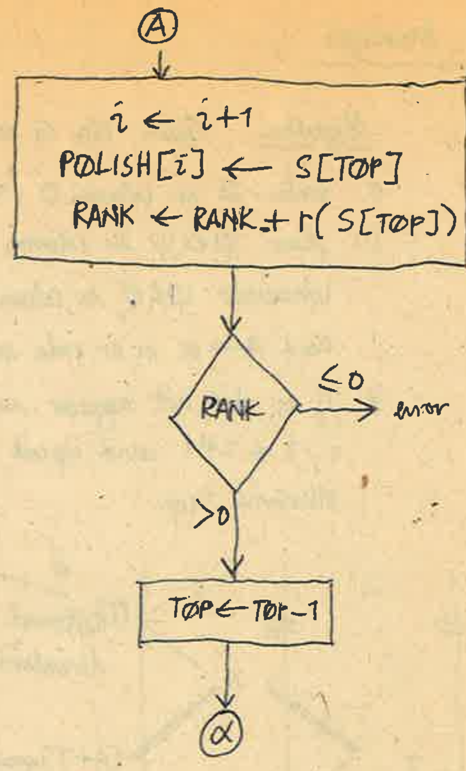


NAMES:

output array	POLISH
stack	S
input (character currently being scanned)	NEXT

NEXTCHAR (, ,)





example:

(a+b↑c↑d) * (e+f/g))

the other one is already in the stack.

f	NEXT	g	STACK	POLISH
		0	(	
9	(	0	((	
7	a	8	((a	
1	+	0	((	a
1	+	0	((+	a
7	b		((+b	

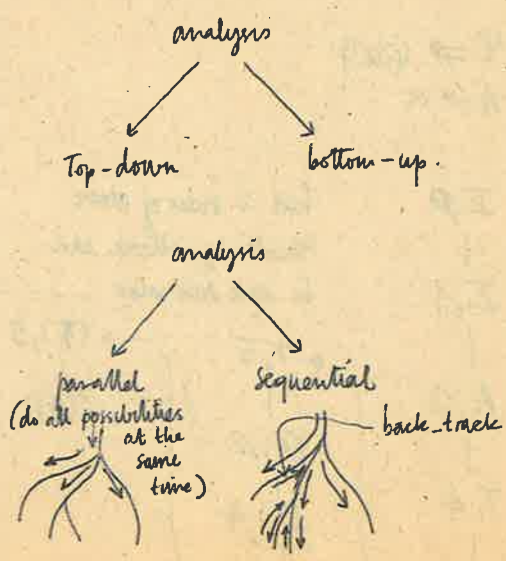
result:

abcd↑↑+efg/+*

SYNTAX ANALYSIS

given a sentence (string of nonterminals) and a grammar (set of rules) (context free grammars)

find its structure (derivation)



sequential analysis without backtracking:

a) Precedence grammars

b) LR(k) "left to right with max. k ahead"

Parallel Top-down analysis :

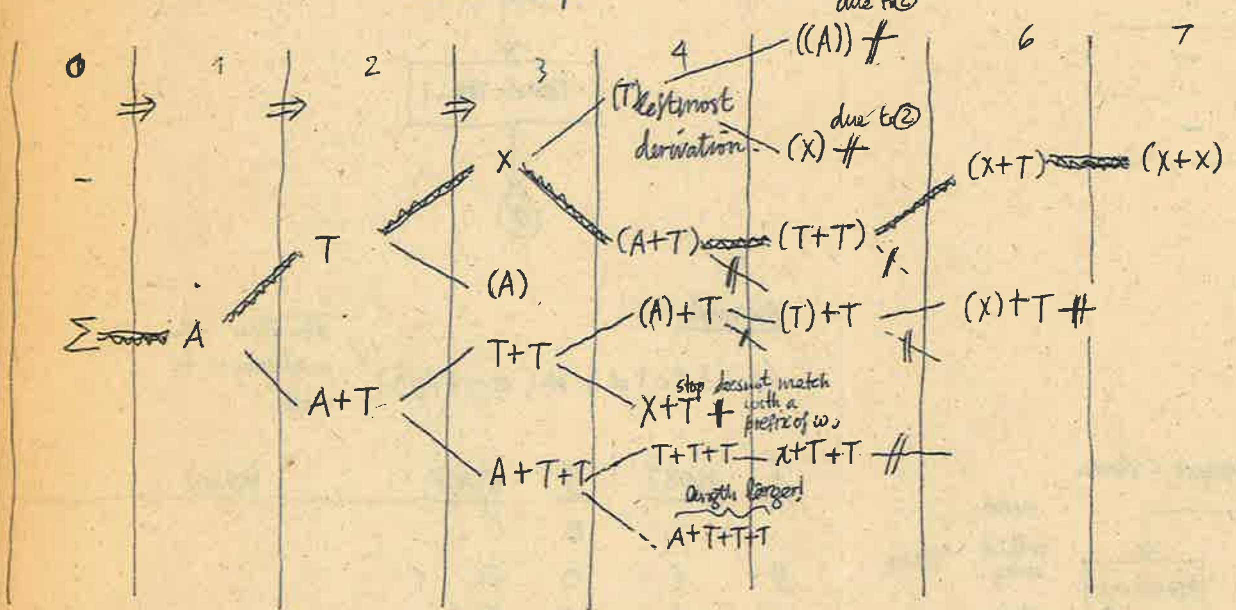
grammar:

$\Sigma \to A$
$A \to T$
$A \to A+T$
$T \to x$
$T \to (A)$

$w = (x+x)$

Algorithm Given cfg G and $w \in T^*$

1. Enter Σ in column 0 $i \leftarrow 1$
2. place $\varphi \alpha \psi$ in column i whenever $\varphi A \psi$ is column $i-1$ and $A \to \alpha$ is a rule in G
3. If w does not appear in column i, $i \leftarrow i+1$ and repeat step 2 otherwise Stop.



Criteria to cut down the number of possible dead-ends:

1. Perform only leftmost derivations.
2. φ match with a prefix of w. $\varphi \alpha \psi$
3. $\ell(\varphi\alpha\psi) > \ell(w)$ STOP that path.

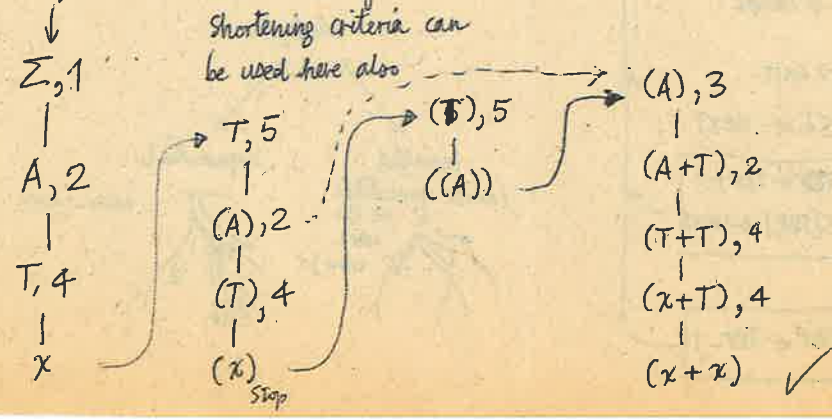
membership problem
given a G and an w
Decide if $w \in L(G)$.
Type 1 : possible.

Sequential top down analysis

1) $\Sigma \to A$
2) $A \to T$
3) $A \to A+T$
4) $T \to x$
5) $T \to (A)$

$\varphi A \psi \Rightarrow \varphi \alpha \psi$
$A \to \alpha$

first 2 rules of above Shortening criteria can be used here also



example :

<statement> → <left part><exp>
<left part> → <identifier> :=
<identifier> → ω | x | y | ...

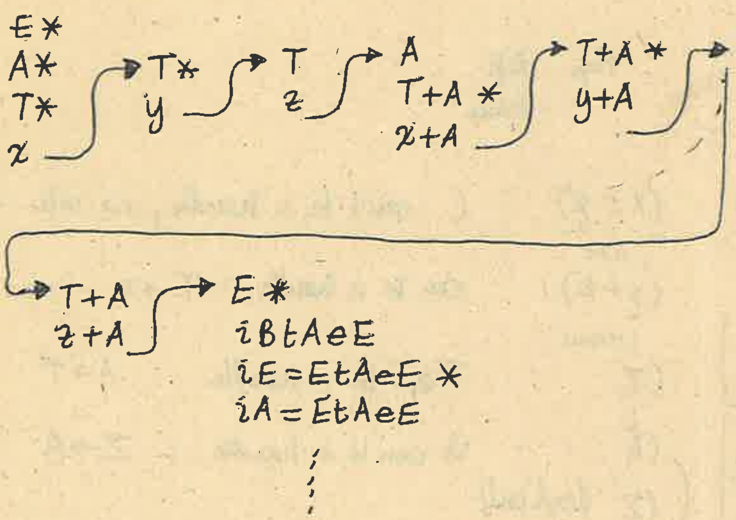
<boolean> B
<arithmetic> A
<expression> E
if : i
then : t
else : e

1) E → A
2) E → iBtAeE
3) B → E = E
4) A → T
5) A → T + A
6) T → x

7) T → y
8) T → z

if x = y then z else x + y

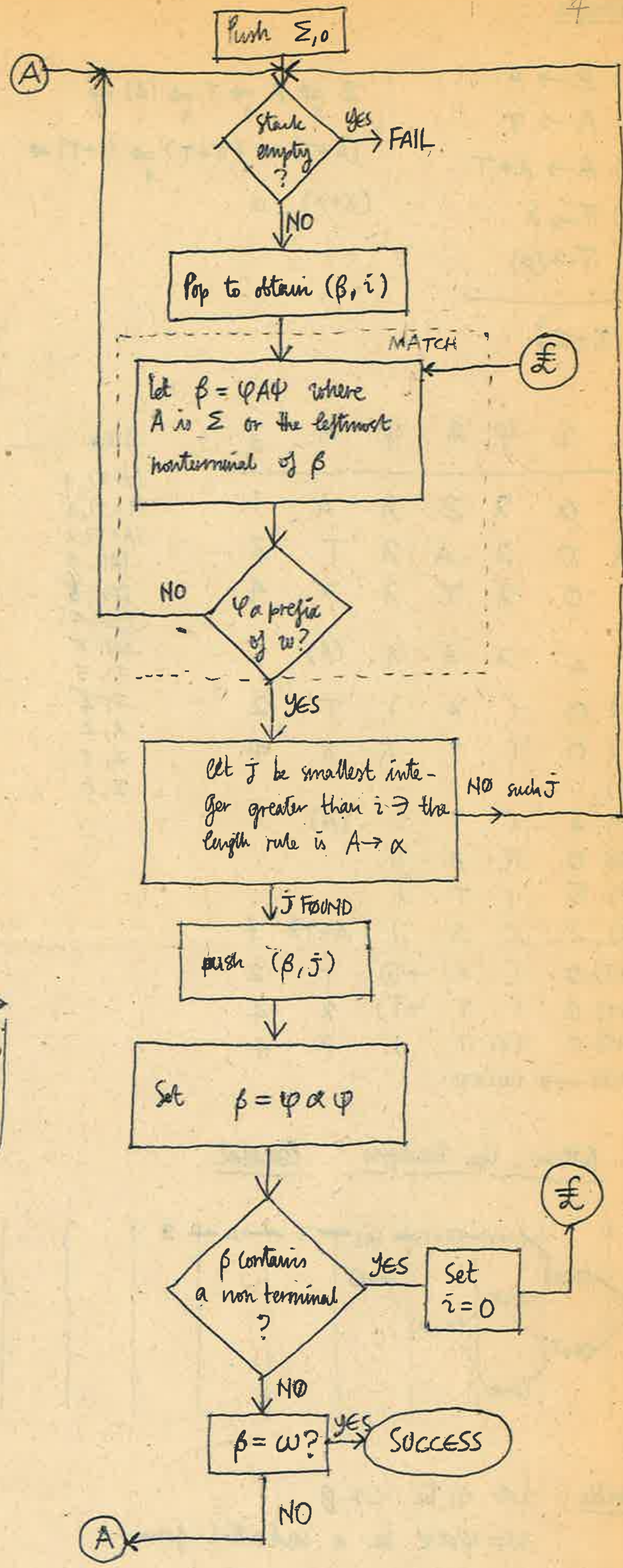
i x = y t z e x + y



about 40 steps we find

050582

Bottom-up analysis



Example :

1) $\Sigma \to A$
2) $A \to T$
3) $A \to A+T$
4) $T \to x$
5) $T \to (A)$

$(x+x)$

$$\Sigma \underset{1}{\Rightarrow} A \underset{2}{\Rightarrow} T \underset{5}{\Rightarrow} (A) \underset{3}{\Rightarrow} (A+T) \Rightarrow (T+T) \underset{4}{\Rightarrow} (x+T) \Rightarrow (x+x) \quad \square$$

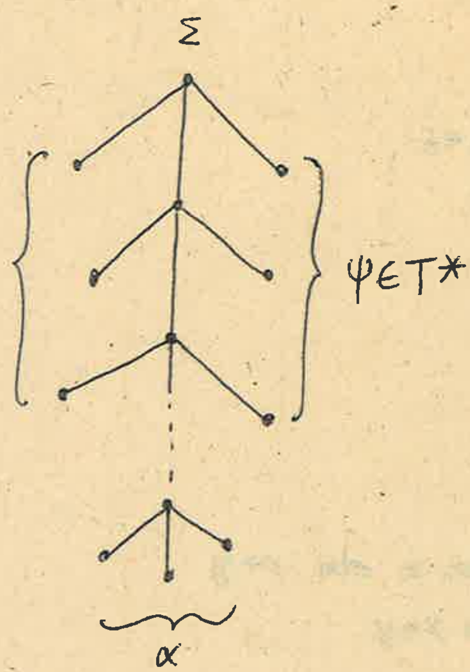
1. w in column 0 ; let $i=1$
2. place $\varphi A \psi$ in column i whenever $\varphi\alpha\psi$ appear in column $i-1$ and $A \to \alpha$ is a rule in G
3.

3. If Σ does not appear in column i let $i \leftarrow i+1$ repeat step 2
4. If Σ appears in column i STOP.

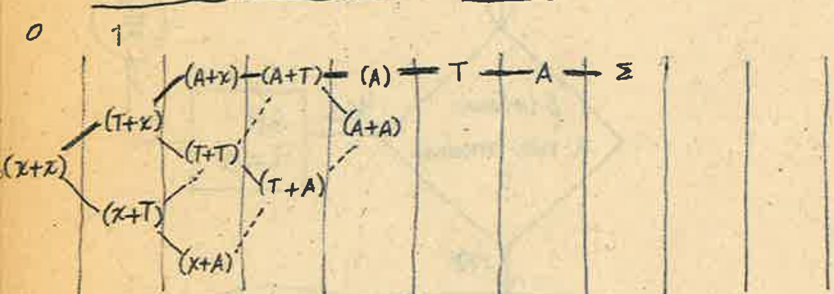
β	i	φ	A	ψ	α	j
Σ	0	λ	Σ	λ	A	1
A	0	λ	A	λ	T	2
T	0	λ	T	λ	x	4
x		.				
T	4	λ	A	λ	(A)	5
(A)	0	(	A	)	T	2
(T)	0	(	T	)	x	4
(x)						
(T)	4	(	T	)	(A)	5
((A))	0	((	A	))		
(T)	5	(	T	)		
(A)	2	(	A	)	A+T	3
(A+T)	0	(	A	+T)	T	2
(T+T)	0	(	T	+T)	x	4
(x+T)	0	(x+	T	)	x	4

(x+x) ⟶ success.

STACK
(x+T), 4
(T+T), 4
(A+T), 2
(A), 5
~~(T), 5~~
~~(T), 4~~
~~(A), 2~~
T, 5
~~T, 4~~
A, 2
Σ, 1
Σ, 0



Bottom - Up Analysis **Parallel**



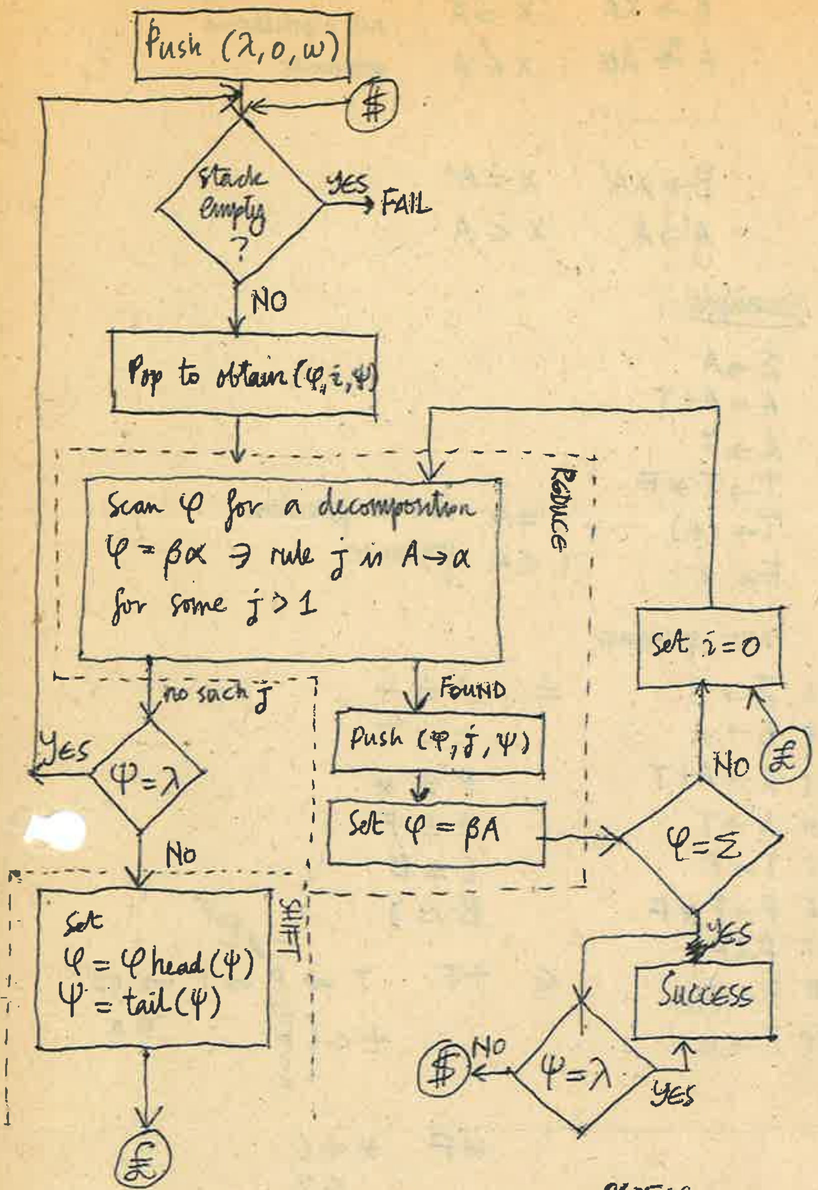
steps : shift
reduce

(x + x)	(can't be a handle, no rule $\to$ (
↓ shift	
(x + x)	can be a handle : $T \to x$
↓ reduce	
(T	T can be a handle : $A \to T$
↓	
(A	A can be a handle : $\Sigma \to A$
(Σ dead end!	
shift (A+	
(A+x	x can be a handle
(A+T	

a) (A+T T is a handle
(A+A

b) (A+T A+T is a handle
(A
(A) (A) is a handle
T T is a handle
A A is a handle
Σ STOP.

Handle : Let G be C.F.G
$w = \varphi\alpha\psi$ be a sentential form
if $A \to \alpha$ is a rule in G
and if $\psi \in T^*$
then α is called a potential (possible) handle
If, moreover, there is a derivation of w of the form
$\Sigma \overset{*}{\Rightarrow} \varphi A \psi \Rightarrow \varphi\alpha\psi$
then α is a handle.



060582

	↑	*	/	+	−	#
↑	⋗	⋗	⋗	⋗	⋗	⋗
*	⋖	⋗	⋗	⋗	⋗	⋗
/	⋖	⋗	⋗	⋗	⋗	⋗
+	⋖	⋖	⋖	⋗	⋗	⋗
−	⋖	⋖	⋖	⋗	⋗	⋗
#	⋖	⋖	⋖	⋖	⋖	=

$\doteq$

⋗

⋖

$\# 3 * 5 - 2 \uparrow 3 / 4 * 3 + 1 \#$

⋖ ↑ ⋗ ⋖ ⋗ ⋗ ⋗ ⋗

⟶

$\# \; 15 \; - \; 2 \uparrow 3 \; / \; 4 \ldots$

⋖ ⋖ · ⋗

$\# \; 15 \; - \; 8 \; \ldots$

⋖ ⋖ - - -

grammar $G = (N, T, P, \Sigma)$

⋖ ≐ ⋗ NUT

1) $X \lessdot Y$ if $\exists\ A \to \alpha X B \beta$ in $P \ni B \stackrel{+}{\Rightarrow} Y\gamma$

2) $X \doteq Y$ if $\exists\ A \to \alpha X Y \beta$ in P

3) $X \gtrdot a$ if $A \to \alpha B Y \beta$ is in P, $B \stackrel{+}{\Rightarrow} \gamma X$ and $Y \stackrel{*}{\Rightarrow} a\delta$

$\#$ $\quad \# \lessdot x \quad \forall x \ni \Sigma \stackrel{+}{\Rightarrow} X\alpha$

$Y \gtrdot \# \quad \forall Y \ni \Sigma \stackrel{+}{\Rightarrow} \alpha Y$

Example

$\Sigma \to S$

$S \to aSSb$

$S \to c$

	S	a	b	c	#
S	≐	⋖	≐	⋖	⋗
a	≐	⋖		⋖	
b		⋗	⋗	⋗	⋗
c		⋗	⋗	⋗	⋗
#	⋖	⋖		⋖	

PRECEDENCE TABLE

1. $\Sigma \to A$
2. $A \to T$
3. $A \to A+T$
4. $T \to x$
5. $T \to (A)$

φ	i	ψ	j	β	α	A
λ	0	(x+x)				
(		x+x)				
(x		+x)	4	(	x	T
(T		+x)	2	(	T	A
(A		+x)				
(A+		x)				
(A+x		)	4	(A+	x	T

(A+x4)
(T2+x)
(x4+x)
λ0 (x+x)

FLOYD	WIRTH/WEBER	KNUTH
precedence relations	simple precedence grammars	LR(K)

Books:

AHO - ULLMAN The theory of Parsing, Translation and compiling

Vol 1 CH1 / CH2

DENNING - DENNIS - QUALITZ
Machines, language, Qualitz.
CH10. Syntax analysis

Lewis / Rosenk... Compiler design theory

D GRIES Compiler construction

BRILLINGER Introduction to Data Structures and nonnumeric comp.

RUN $*METU/ASM370 ; DATA $ CRA

$\doteq$ $a \doteq S$
$S \doteq S$
$S \doteq b$

$\lessdot$ aS $S \overset{+}{\Rightarrow} aSSb$ $a \lessdot a$
$\overset{+}{\Rightarrow} c$ $a \lessdot c$

SS $S \lessdot a$
$S \lessdot c$

$\gtrdot$ SS $S \overset{+}{\Rightarrow} aSSb$ $b \gtrdot a$
$\overset{+}{\Rightarrow} c$ $b \gtrdot c$
$S \Rightarrow aSSb$ $c \gtrdot a$
$\Rightarrow c$ $c \gtrdot c$

Sb $S \Rightarrow aSSb$ $b \gtrdot b$
$S \Rightarrow c$ $c \gtrdot b$

$\# \lessdot X \ \forall X \ni \Sigma \overset{+}{\Rightarrow} X\alpha$
$Y \gtrdot \# \ \forall Y \ni \Sigma \overset{+}{\Rightarrow} \alpha Y$

$\# \lessdot S$ $S \gtrdot \#$
$\# \lessdot a$ $b \gtrdot \#$
$\# \lessdot c$ $c \gtrdot \#$

A grammar G, in which at most one Wirth-Weber precedence exists between any pair of symbols in NUT is called a precedence grammar.

thm: any simple precedence grammar is unambiguous.

Any Context free grammar can be transformed into Chomsky Normal Form and Chomsky Normal Form can be transformed into precedence grammar.

$B \to XA$ $X \doteq A$
$A \overset{*}{\Rightarrow} A\alpha$ $X \lessdot A$

∴ This grammar is not a precedence grammar.

$B \to XA'$ $X \doteq A'$
$A' \to A$ $X \lessdot A$

Example:

$\Sigma \to A$
$A \to A+T$
$A \to T$
$T \to T*F$
$T \to (A)$ $\longrightarrow$ $(\doteq A$ not a precedence
$F \to x$ $(\lessdot A$ grammar.

new grammar:

1 $\Sigma \to B$
2 $B \to A$
3 $A \to A+T$
4 $A \to T$
5 $T \to P$
6 $P \to P*F$
7 $P \to F$
8 $F \to (B)$
9 $F \to a$

$\doteq$ $A \doteq +$
$+ \doteq T$
$P \doteq *$
$* \doteq F$
$(\doteq B$
$B \doteq)$

$\lessdot$ +T $T \Rightarrow P \Rightarrow F \Rightarrow (B)$, $P \Rightarrow P*F$, $F \Rightarrow x$
$+ \lessdot \{P, F, (, x\}$

*F $* \lessdot ($
$\lessdot x$

(B $(\lessdot \{A, T, P, F, (, x\}$

$\gtrdot$ A+ $A \Rightarrow A+T$, $T \Rightarrow P \Rightarrow P*F \Rightarrow P*x$, $F \Rightarrow P*(B)$

$\# \lessdot \{B, A, T, P, F, (, x\}$

$\{T, P, F,), x\} \gtrdot +$

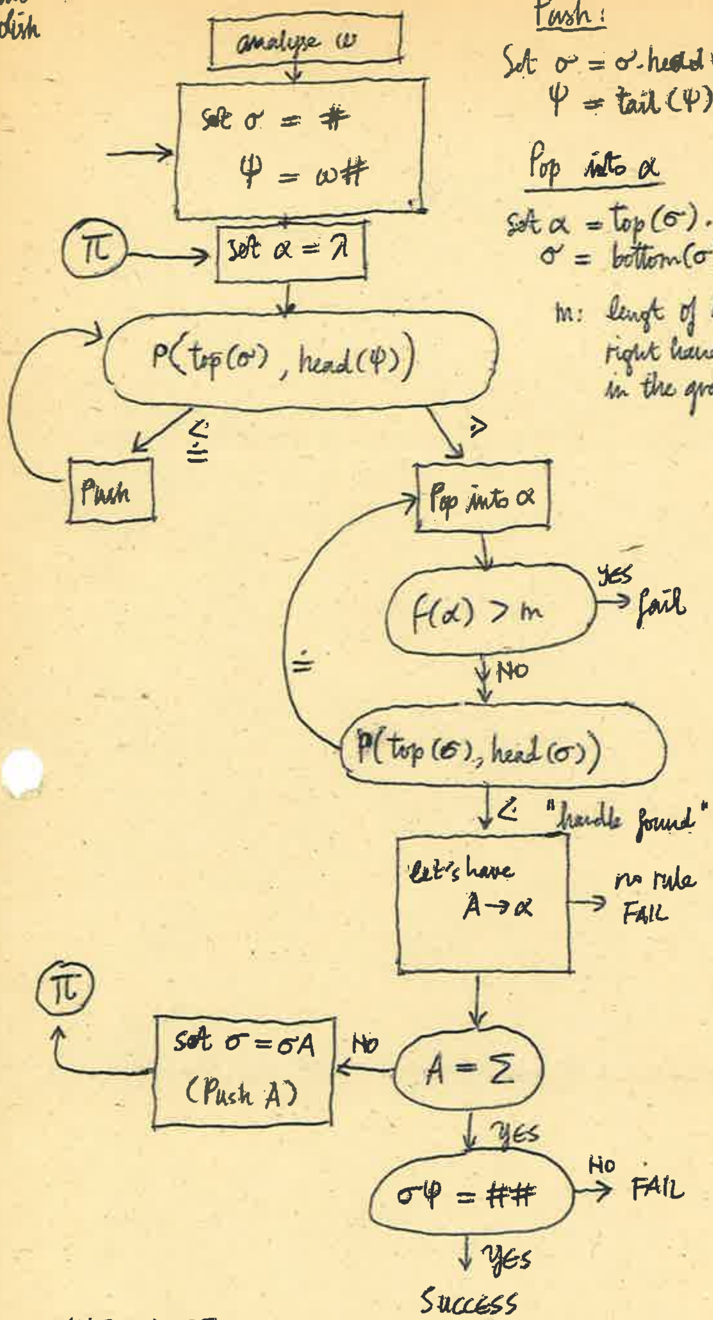
P* $\{F,), x\} \gtrdot *$

$\{B, A, T, P, F,), x\} \gtrdot \#$

B) $\{A, T, P, F,), x\} \gtrdot)$

like Conversion into polish

FINDING HANDLE AND PARSING



Push:

Set $\sigma = \sigma \cdot head(\psi)$

$\psi = tail(\psi)$

Pop into α

Set $\alpha = top(\sigma) \cdot \alpha$

$\sigma = bottom(\sigma)$

m: length of longest right hand side in the grammar.

CH8 PART I
PART II
PART III

Donovan P279