Lecture 1

- Administrative:
  - 60% final exam + 30% labs + 10% Seminar

- Actual Lecture:
  - Compiler vs. Interpreter

    Compiler takes the whole code and simulate it while an interpreter takes line by line

Structure of Compiler:

- the output of each phase is the input of the next one

Phase 1: Scanning

Taking the source program as text and transform them into tokens.

ST = Symbol Table → the place where the identifiers are stored

Phase 2: Parsing (Syntactical Analysis)

We have a set of sintactic rules for each language (declarations, if, ...)

If everything is correct → a tree will be constructed.

Phase 3: Semantic Analysis

Verifies the type.

Puts the info of types into the nodes of the syntax tree.

Phase 4: Intermediary Code Generation

Intermediary code - very simple

Register allocation

Phase 5: Intermediary Code Optimizer.

Phase 6: Object Code Generation.

This process can be reverse.

Symbol table Management and error Handling interact with each phase (with specific ones)

CHAPTER 1: SCANNING

Program Internal Form = the list of created tokens.

Ex:

Separator

Mary has a little lamb.

Tokens: Mary
has
a
little
lamb
.

The separators - we ignore are called "White Spaces"

Ex: if (x==y) { x=y+2 }

Tokens: if
(
x
==
y
)
{
x
=
y
+
2
}

Why does it know that == are together.

Look ahead → for some characters the compiler looks ahead to determine the correct tokens (eg. ==, != ...)

After we detect the tokens we want to know what they are.

Reserved Words - 1 meaning
Key word - can have multiple meanings
(can be redefined)

Reserved Words
Delimiter /Separator
Operator
Identifier
Constant

$\left.\right\}$ Classes of tokens

Identifiers - We cannot create a list of all identifiers so we create some rules

Seminar 1

07.10.2024

- At least 2 activities (HW or class activity) for the point (final grade)

BNF (Backus - Naur Form)

→ 4 languages constructs

↳ non-terminals (written between <>)

↳ terminals (no special delimiters)

↳ Meta-linguistic connectors

a. ::= equals by definition

b. | alternative (OR)

General shape of BNF definition:

<construct> ::= expr_1 | expr_2 | .... | expr_n

(expr_i - combination of terminals and/or nonterminals)

e.g. <letter> ::= a | b | ... | z | A | .... | Z

<letter_sequence> ::= <letter> | <letter><let_sequence>

(Specify, using BNF, all non empty seq. of letter)

e.g. 2) Specify both sign and unsigned integers with the following constraints:

- 0 has no sign
- numbers of at least 2 digits can't start with 0.

<integer> ::= 0 | <sign><unsigned> | <unsigned>

<sign> ::= - | +

<unsigned> ::= <nonzerodigit>

<nonzerodigit> ::= 1 | 2 | ... | 9

<digit_seq> ::= <digit> | <digit><digit_seq>

<digit> ::= 0 | <nonzerodigit>

EBNF (Extended BNF)

- the main element here is allowing us to use repetition.

Wirta's dialect:

New constructs :

- { } - repeat 0 or more times
- [ ] - optionality
- ( ) - math grouping
- (* *) - comments
- rules end with .

Changes to the concrete syntax:

- Nonterminals loose <> → written as they are
- Terminals are written between " "
- ::= becomes =

in Sample mini-language
↳ at identifiers :

identifier ::= letter {letter | digit}

letter ::= "A" | "B" | . . . | "Z" | "a" | . . . | "z"

digit ::= "0" | "1" | . . . . | "9"

Seminar 2

The Scanning Algorithm

Input: source.txt + tokens.txt

output: PIF + ST + Lexical Errors (if any)

Source.txt Example:

```
program test
begin
    var a: integer;
        b: integer;
        c: string;
    a := 1;
    b := a+1;
    c := "message";
    write (c);
end
```

PIF: table with 2 cols.

token	ST_pos
program	-1
id	0
begin	-1
var	-1
id	1
:	-1
integer	-1
;	-1
id	2
⋮	
id	1
:=	-1
const	4
⋮	
write	-1
(	-1
id	3
)	-1
end	-1

ST - Here we store only identifiers and constants

ST_pos	symbol
0	test
1	a
2	b
3	c
4	1
5	"message"

- first we search to see if the token is Here becaus in the ST we shouldn't have duplicates

e.g. for hash func:

Sum of ascii codes % no of positions in hash table

Seminar 3
- Grammars -

1. Given the grammar $G = (N, \Sigma, P, S)$

$N = \{S, C\}$

$\Sigma = \{a, b\}$

$P: S \to ab \mid aCSb$

$C \to S \mid bSb$

$CS \to b,$

prove that $w = ab(ab^2)^2 \in L(G)$

$(ab)^2 = abab \neq aabb = a^2b^2$

$\Rightarrow w = ab(abb)^2 = ab\,abb\,abb$

$S \underset{2}{\Rightarrow} aCSb \underset{4}{\Rightarrow} abSbSb \overset{2}{\underset{1}{\Rightarrow}} ab\,abb\,abb$

$\Rightarrow S \Rightarrow w \Rightarrow S \overset{*}{\Rightarrow} w \Rightarrow w \in L(G)$

2. Given by the grammer $G = (N, \Sigma, P, S)$

$N = \{S\}$

$\Sigma = \{a, b, c\}$

$P: S \to a^2S \mid bc$, find $L(G)$

Let $L = \{a^{2k}bc \mid k \in \mathbb{N}\}$

Prove $L = L(G)$

I by double inclusion:

$L \overset{?}{\subseteq} L(G) \Leftrightarrow (\forall)\, w \in L,\ w \in L(G) \Leftrightarrow$

$\Leftrightarrow (\forall)\, k \in \mathbb{N},\ a^{2k}bc \in L(G)$

$P(n)$: $a^{2n}bc \in L(G)$ and prove that for $(\forall)\ n \in \mathbb{N}$, $P(n)$ is true

$$P(0) = a^0bc = bc \qquad S \underset{2}{\Rightarrow} bc \Rightarrow P(0)\ T$$

assume $P(n)$ is true and prove that $P(n+1)$ is T

$P(n)$: $a^{2n}bc \in L(G)$ (induction hypotesis) 1

$$S \overset{*}{\Rightarrow} a^{2n}bc$$

$P(n+1)$: $a^{2n+2}bc$

$$S \underset{1}{\Rightarrow} a^2S \overset{*}{\Rightarrow} a^2a^{2n}bc = a^{2n+2}bc$$

$$\Rightarrow S \overset{*}{\Rightarrow} a^{2n}bc \Rightarrow P(n+1)\ T \quad 2$$

From 1 and 2 $\Rightarrow$ $P(n)$ true for $(\forall)\ n \in \mathbb{N}$

$$\Rightarrow L \subseteq L(G)$$

II

$$S \underset{2}{\Rightarrow} bc = a^0bc$$

$$S \underset{1}{\Downarrow} a^2S \underset{2}{\Rightarrow} a^2bc$$

$$a^2S \underset{1}{\Downarrow} a^4S \Rightarrow a^4bc$$

$$a^4S \underset{1}{\Downarrow} a^6S \ \ldots\ldots$$

For the exam is enough to prove I, generate the tree above and explain (see notes on discord)

3. Find a grammar that generates

$$L = \{ 0^n, 1^n, 2^m ; \ n, m \in \mathbb{N}^* \}$$

$G = \{N, \Sigma, S, P\}$

$N = \{A, B, S\}$

P: $A \to 0A1 \mid 01$

$B \to 2B \mid 2$

$S \to AB$

I $L \subseteq L(G)$ $(\forall)$ $m, n \in \mathbb{N}^*$

$$S \underset{1}{\Rightarrow} AB \overset{n}{\underset{(a)}{\Rightarrow}} 0^n 1^n B \underset{(b)}{\Rightarrow} 0^n 1^n 2^m \Rightarrow$$

$$S \overset{n+m+1}{\Rightarrow} 0^n 1^n 2^m \in L(G)$$

(a) $A \overset{n}{\Rightarrow} 0^n 1^n$, $n \in \mathbb{N}^*$

(b) $B \overset{m}{\Rightarrow} 2^m$, $m \in \mathbb{N}^*$

Seminar 4

Finite Automata (FA)

alphabet

1. Given the FA : $M = (Q, \Sigma, \delta, q_0, F)$

States; transition function; init states; Set of final States

$$Q = \{q_0, q_1, q_2, q_3, q_f\}, \quad \Sigma = \{1, 2, 3\}, \quad F = \{q_f\}$$

δ	1	2	3
q_0	$\{q_0, q_1\}$	$\{q_0, q_2\}$	$\{q_0, q_3\}$
q_1	$\{q_1, q_f\}$	$\{q_1\}$	$\{q_1\}$
q_2	$\{q_2\}$	$\{q_2, q_f\}$	$\{q_2\}$
q_3	$\{q_3\}$	$\{q_3\}$	$\{q_3, q_f\}$
q_f	$\emptyset$	$\emptyset$	$\emptyset$

$\delta(q_0, 1) = \{q_0, q_1\}$ → another way of representing FA

General Configuration: $(\overset{\in Q}{q}, \overset{\in \Sigma^*}{x})$

Σ^* = set of all posible sequences of Σ including the empty one

Initial Config: $(q_0, \overset{\in \Sigma^*}{w})$

Final Configs.: $(\underset{\in F}{q_f}, \varepsilon)$ ← empty sequence

Direct transition:

$$(\underset{\in Q}{p}, ax) \vdash (\underset{\in Q}{q}, x)$$, $a \to$ sympol, $x \in \Sigma^*$

$$\iff q \in \delta(p, a)$$

$p \xrightarrow{a} q$ (Representation in a graph

Transition of order n : $\vdash^n$

One or more transitions: $\vdash^+$

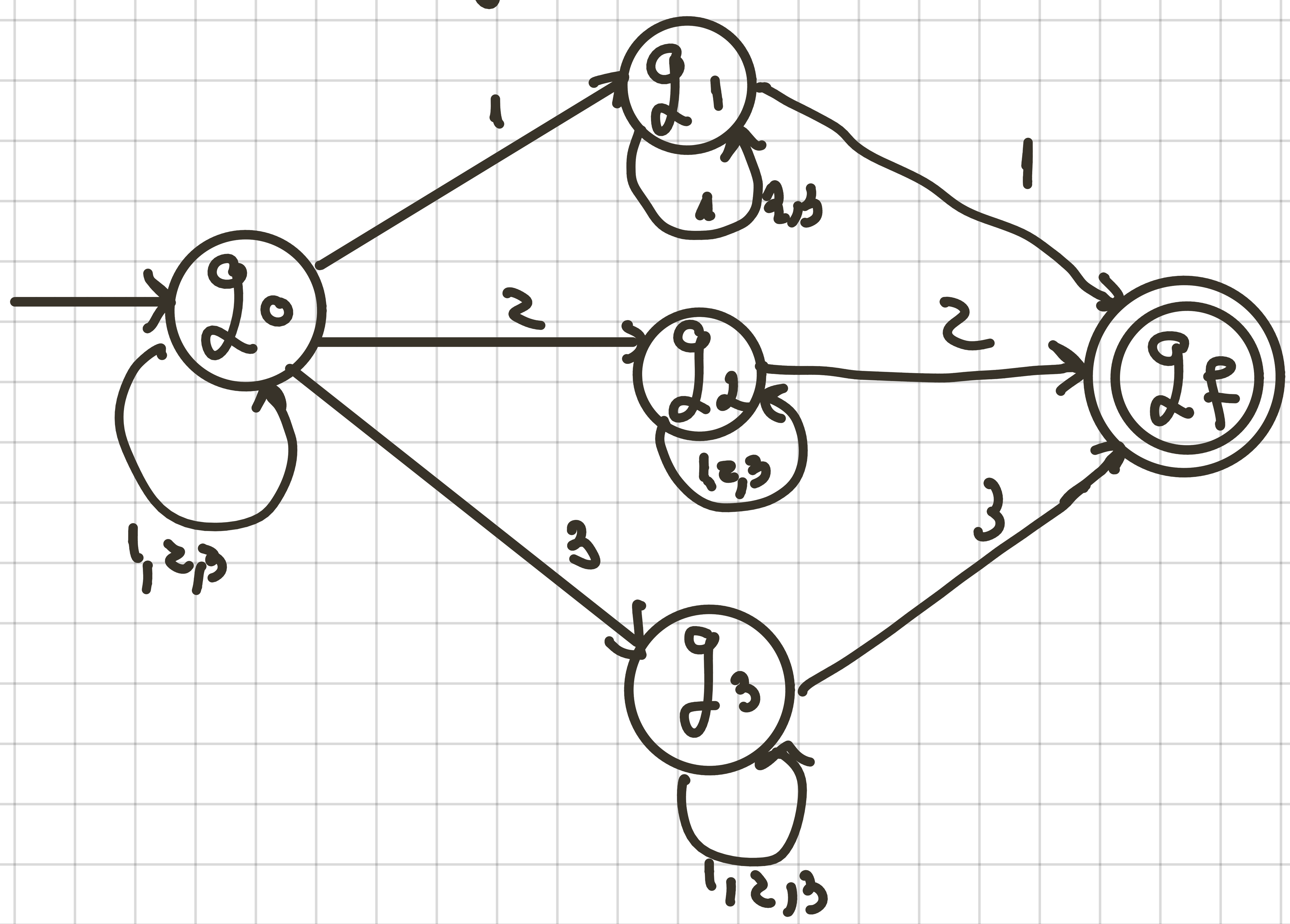
Zero or more -"- $\vdash^*$

$$L(M) = \{ w \in \Sigma^* \mid (q_0, w) \vdash^* (q_f, \varepsilon), q_f \in F \}$$

(definition of the problem

With the given FA :

Prove that $12321 \in L(M)$

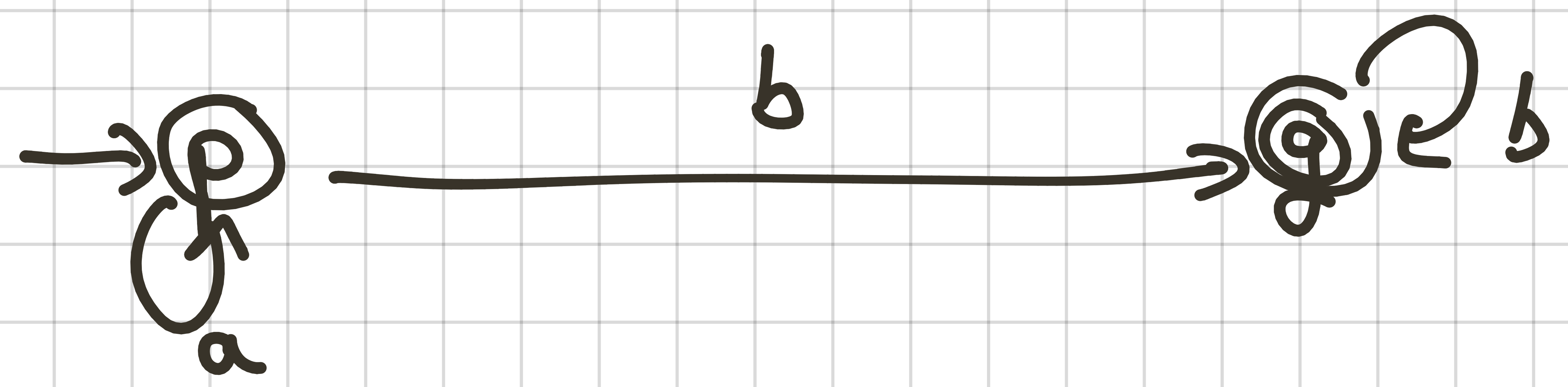


Solution: ? $(q_0, 12321) \vdash^* (q_f, \varepsilon)$

$(q_0, 12321) \vdash (q_1, 2321) \vdash (q_1, 321) \vdash (q_1, 21) \vdash$
$(q_1, 1) \vdash (q_f, \varepsilon)$

$\Rightarrow (q_0\ w) \vdash^5 (q_f, \varepsilon) \Rightarrow w = 12321 \in L(M)$

2.

M:  ? $L(M)$

Let $L = \{ a^n b^m \mid n \in N, m \in N^* \}$

Prove that $L = L(M)$

Induction:

I. . $L \subset L(M) \iff$ for any $n \in \mathbb{N}, m \in \mathbb{N}^*$, $a^n b^m \in L(M)$

Let $n \in \mathbb{N}$, $m \in \mathbb{N}^*$ - fixed

$$(p a^n, b^m) \underset{\text{based on } a}{\vdash^{n}} (p, b^m) \vdash^{m} (q, b^{m-1})$$

a) $(p, b^n) \vdash^{n} (p, \varepsilon)$, $(\forall)\ n \in \mathbb{N}$

b) $(q, b^m) \vdash^{m} (q, \varepsilon)$, $(\forall)\ m \in \mathbb{N}$

$$\underset{(b)}{\vdash^{m-1}} (q, \varepsilon) \Rightarrow (p, a^n b^m) \vdash^{n+m} (q, \varepsilon)$$

$$\Rightarrow a^n b^m \in L(M)$$

Now we need to prove a) or b)

a) $P(n) : (p, a^n) \vdash^{n} (p, \varepsilon)$, $n \in \mathbb{N}$

I. $P(0) = (p, a^0) \vdash^{0} (p, \varepsilon)$ A by def.

II. $P(k) \to P(k+1)$

$P(k) : (p, a^k) \vdash^{k} (p, \varepsilon)$

$P(k+1) : (p, a^{k+1}) \underset{P(k)}{\vdash^{k}} (p, a) \vdash (p, \varepsilon)$

$\Rightarrow$ we have $k+1$ steps $\Rightarrow P(k) \to P(k+1)$

From (I) and (II) $\Rightarrow P(n)$ true for any $n \in \mathbb{N}$

II. $L(M) \subseteq L$?

....

(3) Provide the automata for the following:

a) integer numbers

b) variable declarations

c) $L = \{0^n 1^m 0^q \mid n, m \in \mathbb{N}^*, q \in \mathbb{N}\}$

d) $L = \{0, (01)^n \mid n \in \mathbb{N}\}$

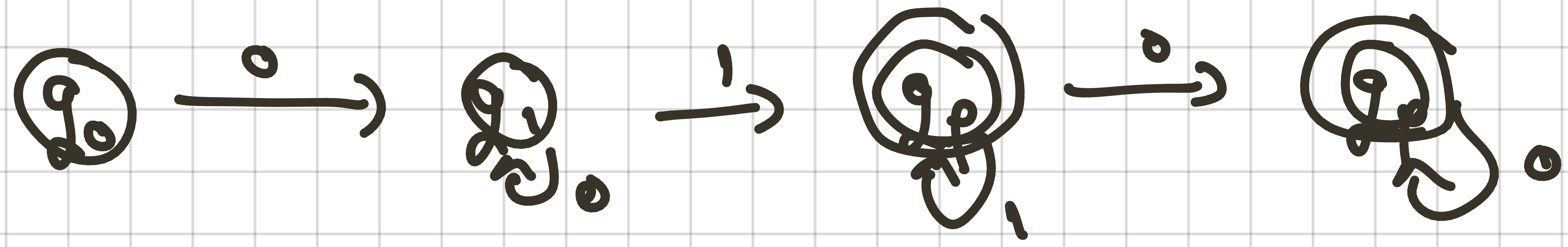
e) $L = \{c^{3n} \mid n \in \mathbb{N}^*\}$

f) $\Sigma = \{0,1\}$ - lang. containing all seq. with at least 2 cons. 0's.

Solutions:

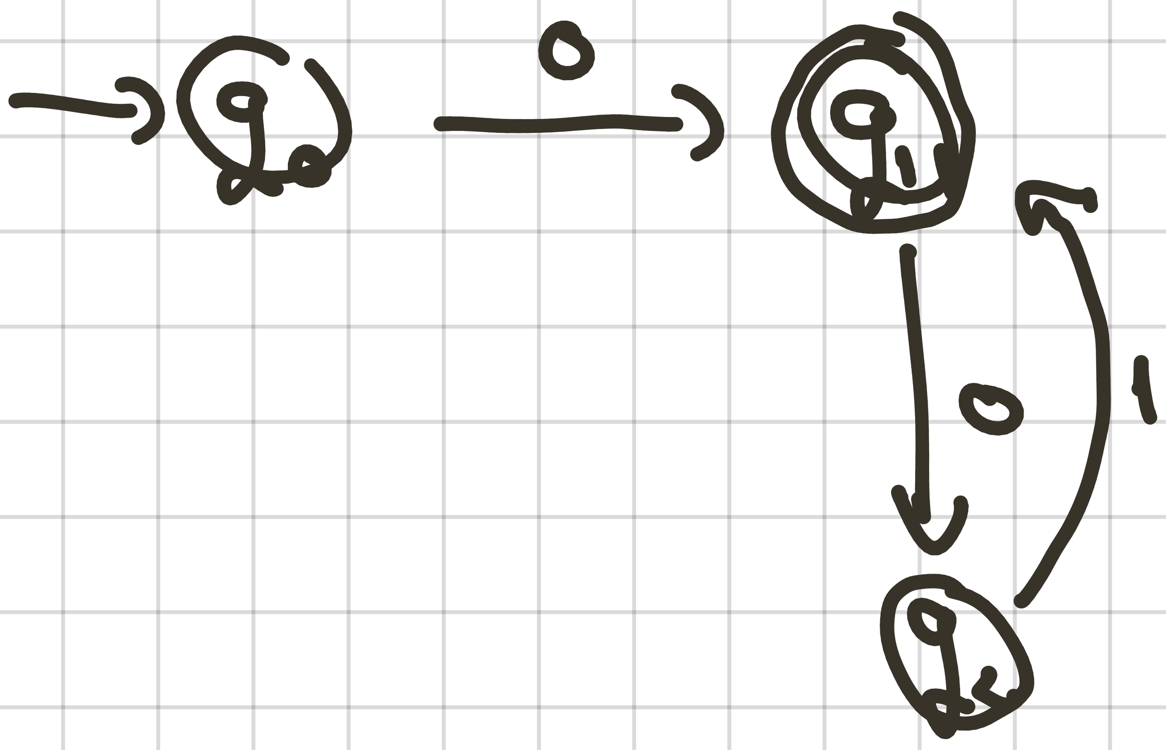
c) 

OR

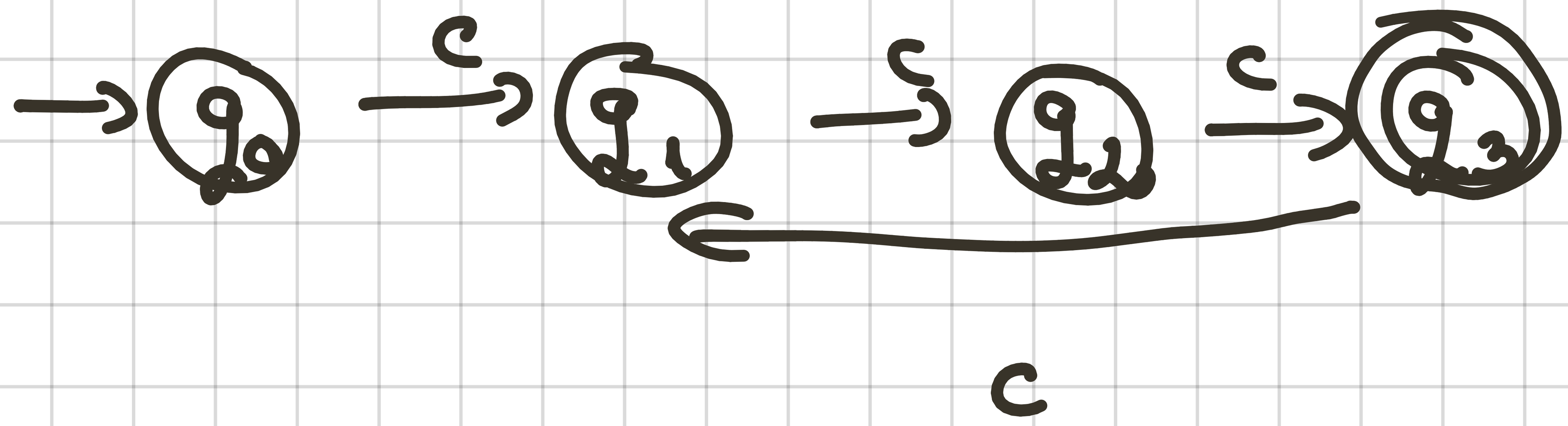


We can have multiple finals

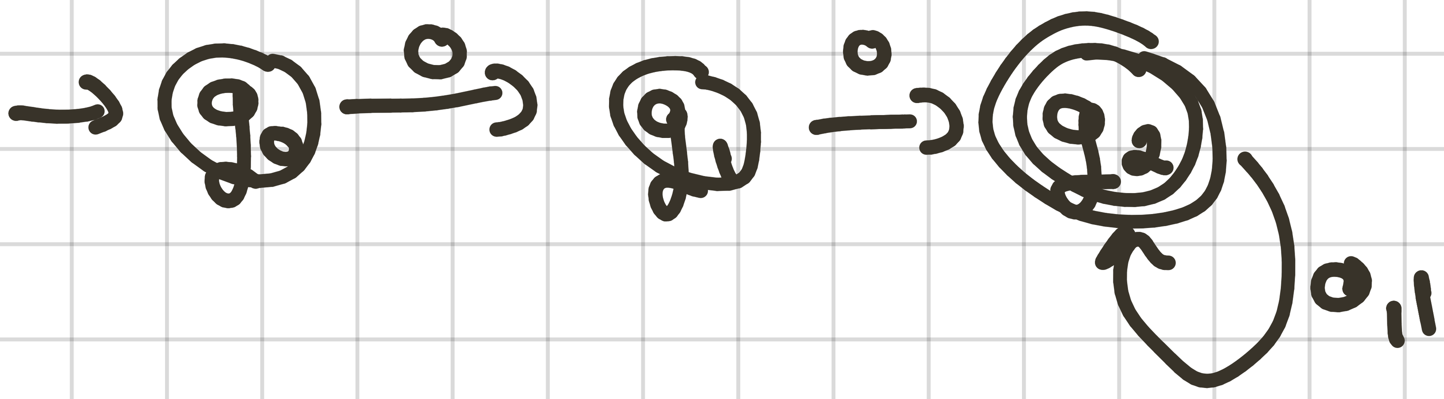
d)



e)



f)



Seminar 5

Regular Grammars:

I. RG $\Leftrightarrow$ FA (Finite Automata)

Right Linear Grammar: - Context free grammar
The right hand side can have at most 2 symbols.

$G = (N, \Sigma, P, S)$ RLG $\Leftrightarrow (\forall) p \in P: A \to aB$ or $A \to b$,
where $A, B \in N$ and $a, b \in \Sigma$

Regular Grammar

$G = (N, \Sigma, P, S)$ RG $\Leftrightarrow$
- G - RLG
- $A \to \varepsilon \notin P$ with exception of $S \to \varepsilon \in P$

Ex1. $G = (\{S, A\}, \{a, b\}, P, S)$

P: $S \to aA \mid \varepsilon$

$A \to aA \mid bA \mid a \mid b$

? $\Leftrightarrow$ FA

$Q = \{S, A, K\}$ $\qquad \Sigma = \{a, b\}$

$q_0 = S.$ $\qquad F = \{K, S\}$

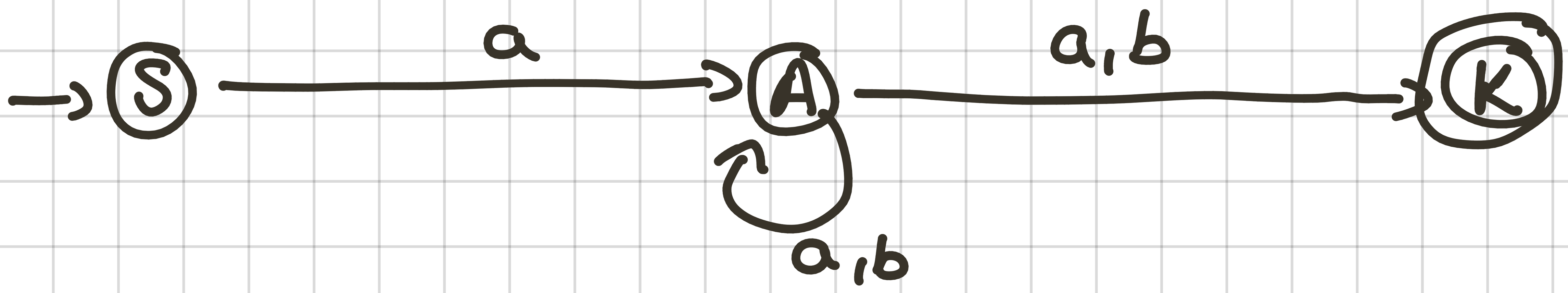
$A \in \delta(S, a)$

$A \in \delta(A, a)$

$A \in \delta(A, b)$

$K \in \delta(A, a)$

$K \in \delta(A, b)$



(Dacă S nu apare în dreapta și ε este producție pentru S atunci e RG)

$S \to \varepsilon$ nu se reprezintă grafic, asta doar indică faptul că starea inițială poate fi și finală.

Dacă ε nu apare în dreapta pt S, S poate apărea în dreapta pt. celelalte.

Ex 2. $M = (Q, \Sigma, q_0, F)$

$Q = \{p, q, r\}$

$\Sigma = \{0, 1\}$

$q_0 = p$

$F = \{r, p\}$

δ	0	1
p	p	q
q	q	r
r	r	r

? ⟺ RLG

$G = \{N, \Sigma, P, S\}$.

$S = p$.

$N = \{p, q, r\}$

$\Sigma = \{0, 1\}$

P :

$p \to 0p \mid 1q \mid 0 \mid \varepsilon$

$q \to 0q \mid 1r \mid 1$

$r \to 0 \mid 1 \mid 1r \mid 0r$

This is why it's not RG but RLG

II. RG $\Leftrightarrow$ RE (Regular Exspression)

RE: $01(0+1)^*$ (also wrote like $0(0+1)^*1$

this means 0 or more

? $\Leftrightarrow$ RG

$0 \to \{0\}$ $\qquad$ $0+1 \to \{0,1\}$

$\varepsilon \to \{\}$ $\qquad$ $(0+1)^* \to \{\varepsilon, 0, 1, 01, 11, 00, 10 \ldots\}$

$\emptyset \to \emptyset$

$0: G_1 = (\{S_1\}, \{0\}, \{S_1 \to 0\}, S_1)$

$1: G_2 = (\{S_2\}, \{1\}, \{S_2 \to 1\}, S_2)$

$01: G_3 = (\{S_1, S_2\}, \{0,1\}, \{S_1 \to 0S_2, S_2 \to 1\}, S_1)$

$0+1: G_4 = (\{S_1, S_2, S_3\}, \{0,1\}, \{S_1 \to 0, S_2 \to 1, S_3 \to 0|1\}, S_3)$

Here S_1 and S_2 are unreachable (See lecture 5)

$\Downarrow$

$G_4' = (\{S_3\}, \{0,1\}, \{S_3 \to 0|1\}, S_3)$

$(0+1)^*: G_5 = (\{S_3\}, \{0,1\}, \{S_3 \to 0S_3 \mid 1S_3 \mid \varepsilon\}, S_3)$

! Not Regular grammer

$01(0+1)^* = G_6(\{S_1, S_2, S_3\}, \{0,1\}, \{S_1 \to 0S_2,\ S_2 \to 1S_3,\ S_3 \to 0S_3 \mid 1S_3 \mid \varepsilon\}, S_1)$

! Not Regular

$$G_6' = (\{S_1, S_2, S_3\}, \{0,1\}, \{S_1 \to 0S_2,\ S_2 \to 1S_3 \mid 1,\ S_3 \to 0S_3 \mid 1S_3 \mid 0 \mid 1\}, S_1)$$

Regular

From RG to RE

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

P: $S \to aA$

$A \to aA \mid bB \mid b$

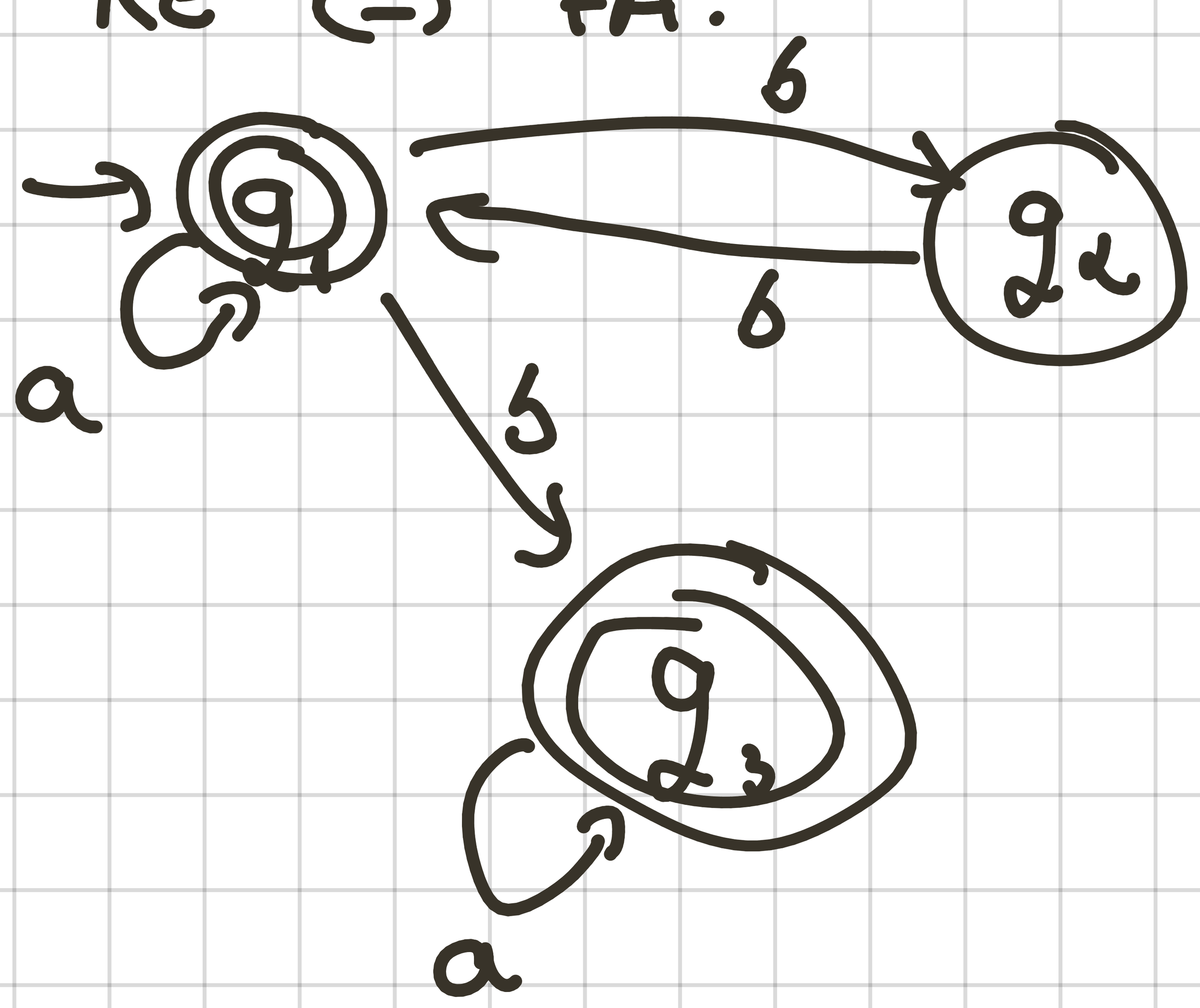
$B \to bB \mid b$

? $\Leftrightarrow$ RE

$$\begin{cases} S = aA \Leftrightarrow a^+b^+ \\ A = aA + bB + b \Leftrightarrow A = aA + b^+ \Leftrightarrow a^*b^+ \\ B = bB + b \Leftrightarrow B = b^*b = b^+ \end{cases}$$

$\Rightarrow$ Sol $= a^+b^+$

5. RE $\Leftrightarrow$ FA.



? $\Leftrightarrow$ RE

$X = Xa + b$, $X = ba^*$

$$\begin{cases} q_1 = q_1 a + q_2 b + \varepsilon \\ q_2 = q_1 b \\ q_3 = q_1 b + q_3 a \end{cases}$$

$$\Leftrightarrow \begin{cases} q_1 = q_1 a + q_1 b^2 + \varepsilon = q_1 (a+b^2) + \varepsilon = \varepsilon (a+b^2)^* = (a+b^2)^* \\ q_2 = q_1 b \Leftrightarrow b(a+b^2)^* \\ q_3 = q_1 b + q_3 a \Leftrightarrow q_3 = b(a+b^2)^* + q_3 a \end{cases}$$

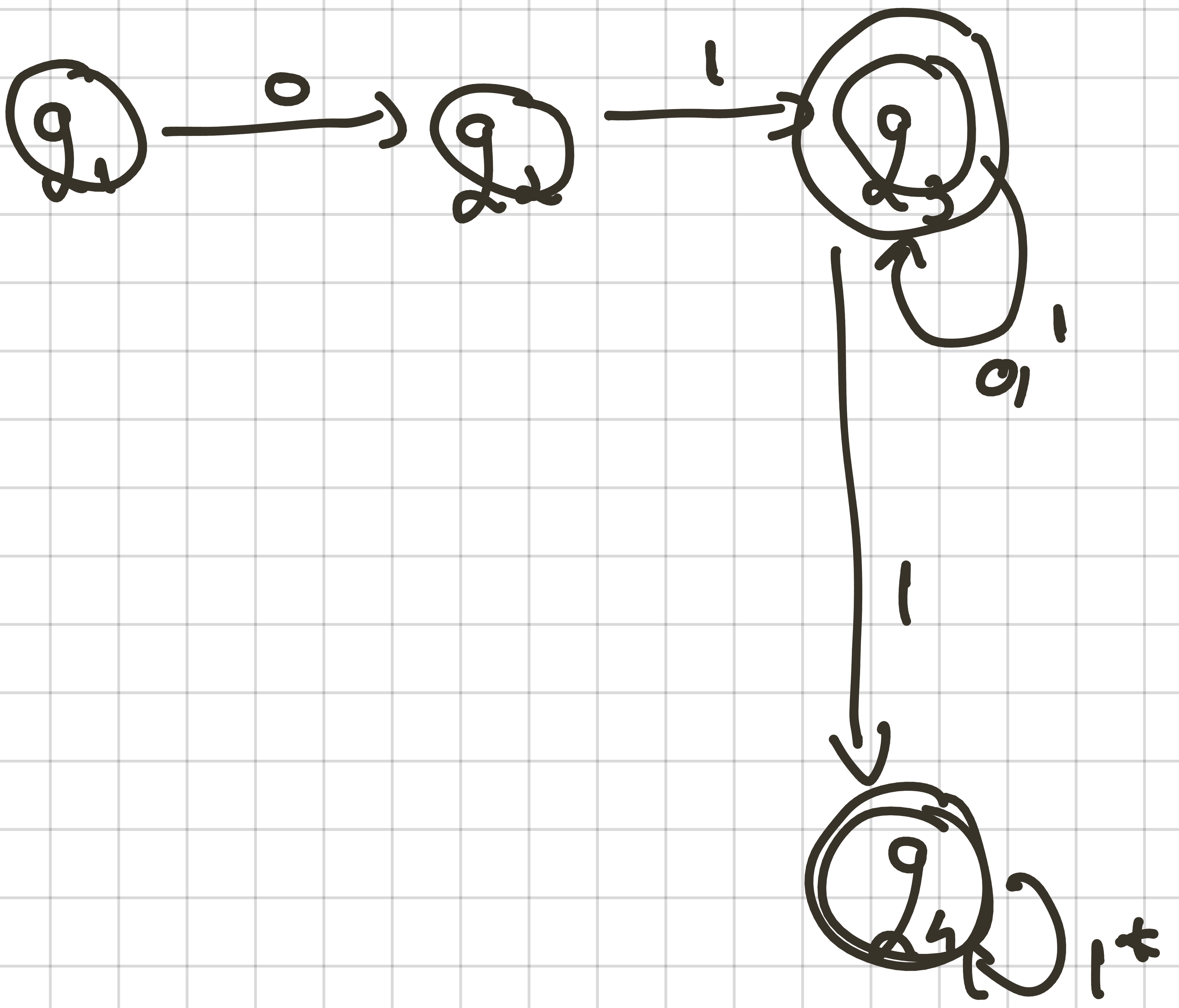
$= (a+b^2)^* b a^*$

Sol: $(a+b^2)^* + (a+b^2)^* ba^* =$

$$= \boxed{(a+b^2)^* (\varepsilon + ba^*)}$$

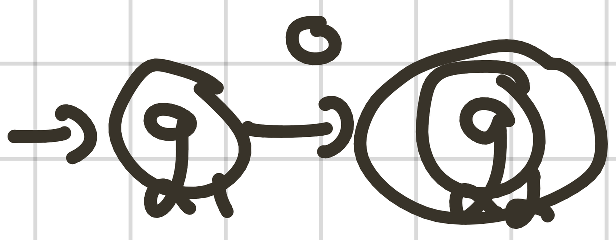
6. RE $\quad 01(1+10)^* \; 1^*$

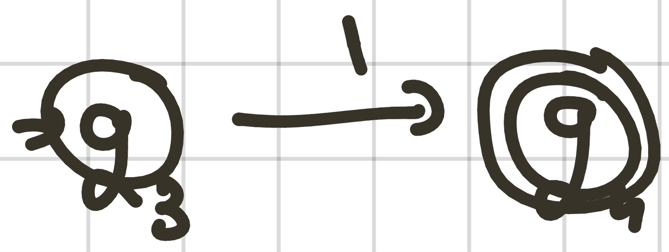
? ⇔ FA.

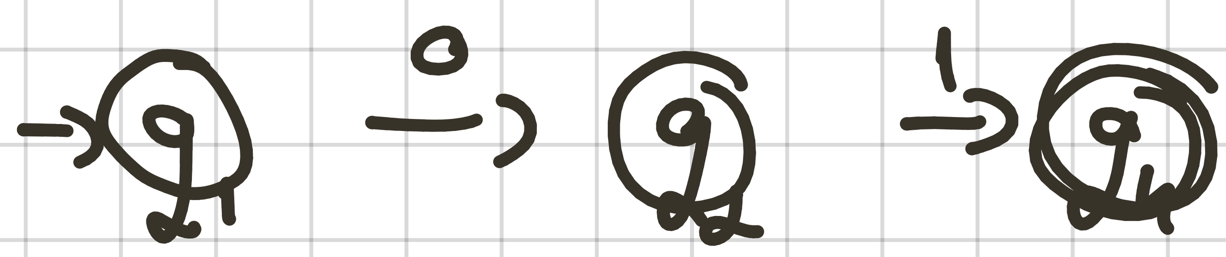


(Not using the rules)

Using the rules:

0 : 

1: 

01 : 

Seminar 9

LR(0)

Ex: $G = (\{S', S, A\}, \{a, b, c\}, P, S')$

P: $S' \to S$

(1) $S \to aA$

(2) $A \to bA$

(3) $A \to c$

$w = abbc$

1. Canonical collection of states:

$S_0 = closure(\{[S' \to .S]\})$

(we start the analysis in a state where we have nothing in the stack)

$= \{[S' \to .S], [S \to .aA]\}$

For each new state we need to compute the goto function.

$S_1 = goto(S_0, S) = closure(\{[S' \to S.]\}) = \{[S' \to S.]$

$goto(S_0, A) = closure(\{\}) = \{\}$

$S_2 = goto(S_0, a) = closure(\{[S \to a.A]\}) = \{[S \to a.A], [A \to .bA], [A \to .c]\}$

$goto(S_0, b) = \{\}$

$goto(S_0, c) = \{\}$

$s_3 = goto(s_2, A) = closure(\{[S \to aA.]\}) =$
$= \{[S \to aA.]\}$

$s_4 = goto(s_2, b) = closure(\{[A \to b.A]\}) =$
$\{[A \to b.A], [A \to .bA], [A \to .c]\}$

$goto(s_4, A) = \{[A \to b.A], [A \to .bA], [A \to .c]\} = \{\}$

$s_5 = goto(s_2, c) = closure(\{[A \to c.]\})$
$= \{[A \to c.]\}$

$s_6 = goto(s_4, A) = closure(\{[A \to bA.]\}) =$
$= \{[A \to bA.]\}.$

$goto(s_4, b) = closure(\{[A \to b.A]\}) = s_4$

$goto(s_4, c) = closure(\{[A \to c.]\}) = s_5$

Next Step : Fill in the parsing table

2. LR (0) Parsing Table:

	Action	goto a	b	c	A	S
S_0	shift	S_2				S_1
S_1	accept					
S_2	shift		S_4	S_5		S_3
S_3	reduce 1					
S_4	shift		S_4	S_5		S_6
S_5	reduce 3					
S_6	reduce 2					

3. Parse the sequence : abbc

Work Stack	Input Stack	Output Band
$\$ s_0$	abbc\$	eps (empty)
$\$ s_0 a \; 2$	bbc\$	eps
$\$ S_0 a \; S_2 b \; S_4$	bc\$	eps
$\$ S_0 a S_2 b S_4 b \; S_4$	c\$	eps
$\$ S_0 a S_2 b S_4 b S_4 b S_5 c$	\$	eps
$\$ S_0 a S_2 b S_4 b S_4 A S_6$	\$	3
$\$ S_0 a S_2 b S_4 A S_6$	\$	23
$\$ S_0 a S_2 A S_3$	\$	223
$\$ S_0 S S_1$	\$	1223
\$	\$	

Seminar 10

SLR : $G = \{\{S', E, T\}, \{+, id, const, (,)\}, P, S'\}$

P: $S' \to E$

(1) $E \to T$

(2) $E \to E + T$

(3) $T \to (E)$

(4) $T \to id$

(5) $T \to const$

$w = id + const$

SLR item $\Leftrightarrow$ LR(0) item : $[A \to \alpha . \beta]$

$s_0 = closure(\{[S' \to E]\}) = \{[S' \to .E], [E \to .T],$
$[E \to .E + T], [T \to .(E)],$
$[T \to .id], [T \to .const]\}.$

$s_1 = goto(s_0, E) = closure(\{[S' \to E.], [E \to E. + T]\})$
$= \{[S' \to E.], [E \to E. + T]\}$

$s_2 = goto(s_0, T) = closure(\{[E \to .T]\}) = \{[E \to T.]\}$

$s_3 = goto(s_0, id) = closure(\{[T \to .id]\}) =$
$= \{[T \to id.]\}$

$s_4 = goto(s_0, const) = closure(\{[T \to .const]\}) =$
$= \{[T \to const.]\}$

$s_5 = goto(s_0, () = closure(\{[T \to .(E)]\})$

$= \{[T \to (.E)], [E \to .T],$
$[E \to .E + T], [T \to .(E)],$
$[T \to .id], [T \to .const]\}$

$s_6 = goto(s_1, +) = closure(\{[E \to E+.T]\})$
$= \{[E \to E+.T], [T \to .(E)], [T \to .id],$
$[T \to .const]\}$

$s_7 = goto(s_5, E) = closure(\{[T \to (E.)], [E \to E.+T]\}) =$
$= \{[T \to (E.)], [E \to E.+T]\}$

$goto(s_5, T) = closure(\{[E \to T.]\}) = s_2$

$goto(s_5, ( ) = s_5$

$goto(s_5, id) = s_3$

$goto(s_5, const) = s_4$

$s_8 = goto(s_6, T) = \{[E \to E+T.]\}$

$s_9 = goto(s_7,)) = \{[T \to (E).]\}$

$goto(s_6, () = s_5$

$goto(s_6, id) = s_3$

$goto(s_6, const) = s_4$

$goto(s_7, +) = s_6$

$FOLLOW(E) = \{\varepsilon, +,)\}$

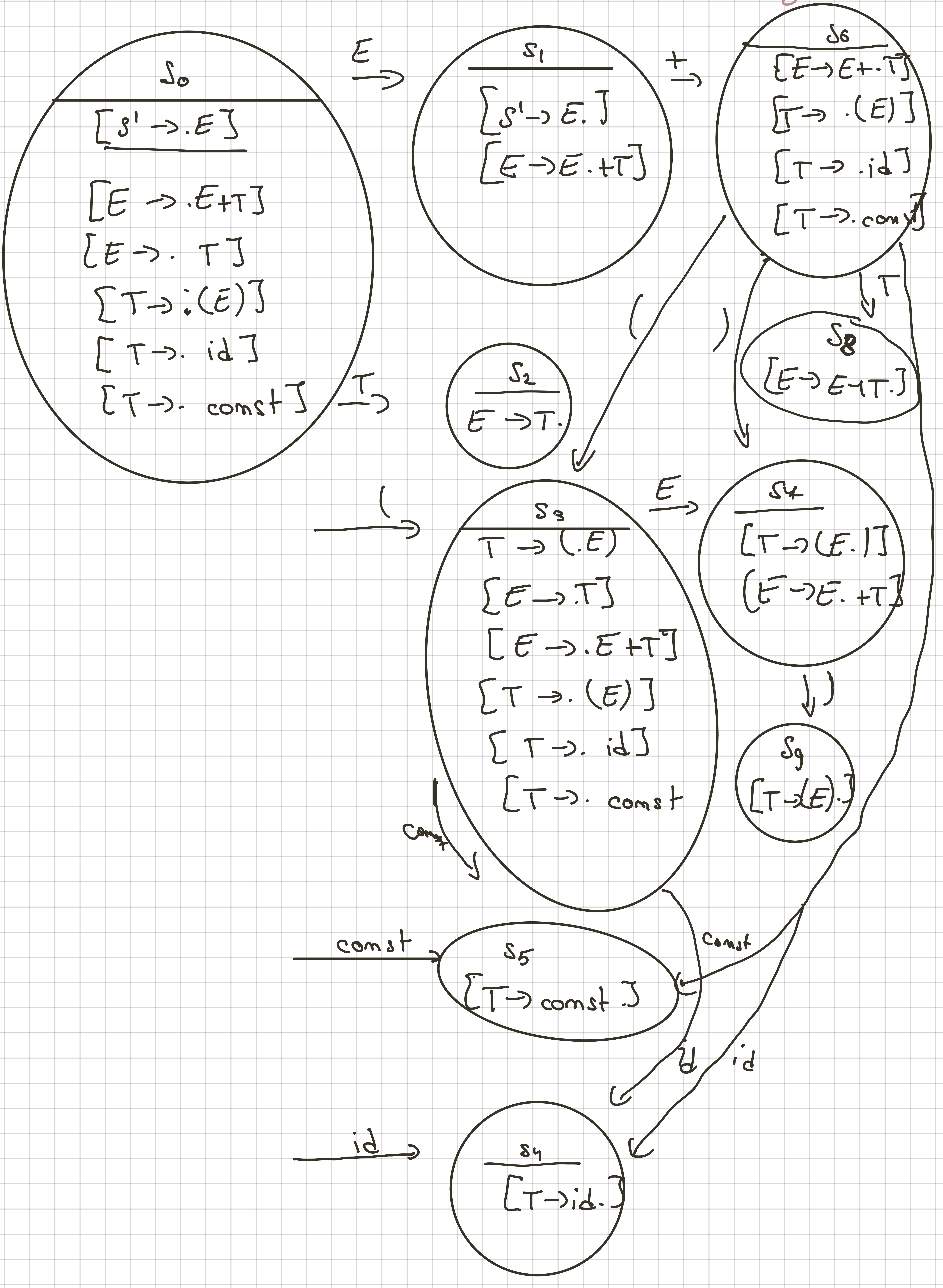
$FOLLOW(T) = \{\varepsilon, +,)\}$

	Action						goto	
	+	id	const	(	)	$	E	T
s_0		shift s_3	shift s_4	shift s_5			s_1	s_2
s_1	S, S_6					A		
s_2	R_1				R_1	R_1		
s_3	R_4				R_4	R_4		
s_4	R_5				R_5	R_5		
s_5		shift s_3	shift s_4	shift s_5			s_7	s_2
s_6		shift s_3	shift s_4	shift s_5				s_8
s_7	shift s_6				shift s_9			
s_8	R_2				R_2	R_2		
s_9	R_3				R_3	R_3		

Parse the sequence

Working Stack	Input Stack	Output Stack
$0	id + const $	ε
$0id3	+const $	ε
$0T2	+ const $	4
$0E1	+const $	14
$0E1+6	const $	14
$0E1+6const4	$	14
$0E1+6T8	$	514
$0E1	$	2514
Accept		

Building the canonical collection graphically



Seminar 11

PDAs.

FA: $M = (Q, \Sigma, \delta, q_0, F)$

PDA: $M = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$

stack alpha

initial stack symbol

$\delta : Q \times \Sigma \to P(Q)$ $\to$ for FA

$\delta : Q \times (\Sigma \cup \{\varepsilon\}) \times \Gamma \to P(Q \times \Gamma^*)$ $\to$ For PDA

$$L_f(M) = \{ w \in \Sigma^* \mid (q_0, w, Z_0) \vdash^* (q_f, \varepsilon, \gamma), q_f \in F, \gamma \in \Gamma^* \}$$

$$L_\varepsilon(M) = \{ w \in \Sigma^* \mid (q_0, w, Z_0) \vdash^* (q, \varepsilon, \varepsilon), q \in Q \}$$

Ex. of representation:

$$p \xrightarrow[\substack{a, X \to X\alpha \\ \varepsilon, X \to Y}]{} q$$

Ex 1. $L_1 = \{0^n 1^{2n} \mid n \in \mathbb{N}^*\}$ $L_1' = \{0^n 2^n \mid n \in \mathbb{N}\}$

$$M = (\{q_0, q_1\}, \{0,1\}, \{Z_0, x\}, \delta, q_0, Z_0, \emptyset)$$

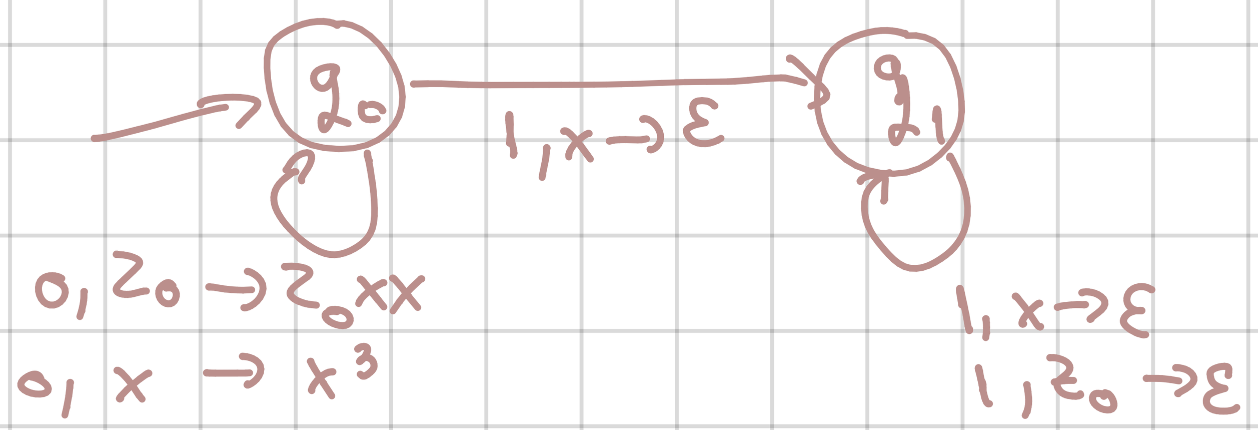
$$\delta(q_0, 0, Z_0) \ni (q_0, Z_0 x x)$$

$$\delta(q_0, 0, x) \ni (q_0, x^3)$$

$$\delta(q_0, 1, x) \ni (q_1, \varepsilon)$$

$$\delta(q_1, 1, x) \ni (q_1, \varepsilon)$$

$$\delta(q_1, 1, Z_0) \ni (q_1, \varepsilon)$$



Ex 2: $w = 0^2 1^2$ (Same PDA as before)

$$(q_0, 0^2 1^4 Z_0) \vdash (q_0, 01^4, Z_0 xx) \vdash (q_0, 1^4, x^4 Z_0)$$
$$\vdash (q_1, 1^3, x^3 Z_0) \vdash (q_1, 1^2, x^2 Z_0) \vdash (q_1, 1, x Z_0)$$
$$\vdash (q_1, \varepsilon, Z_0) \vdash (q_1, \varepsilon, \varepsilon)$$

$w_2 = 0^3 1^4$: There are more 0 than the sequence should have.

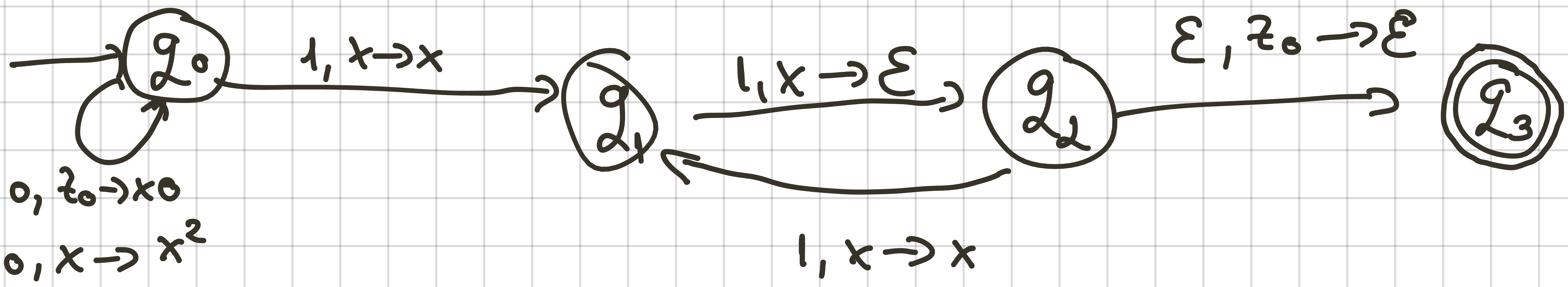
$$(q_0, 0^3 1^4, Z_0) \vdash^3 (q_0, 1^4, x^6 Z_0) \vdash (q_1, 1^3, x^5 Z_0) \vdash^3$$
$$(q_1, \varepsilon, \underline{x^2} Z_0)$$

$w_3 = 01^3$

$$(q_0, 01^3, Z_0) \vdash (q_0, 1^3, x^2 Z_0) \vdash (q_1, 1^2, x Z_0)$$
$$\vdash (q_1, 1, \underline{Z_0}) \vdash (q_1, 1, \underline{\varepsilon})$$ Too many 1's

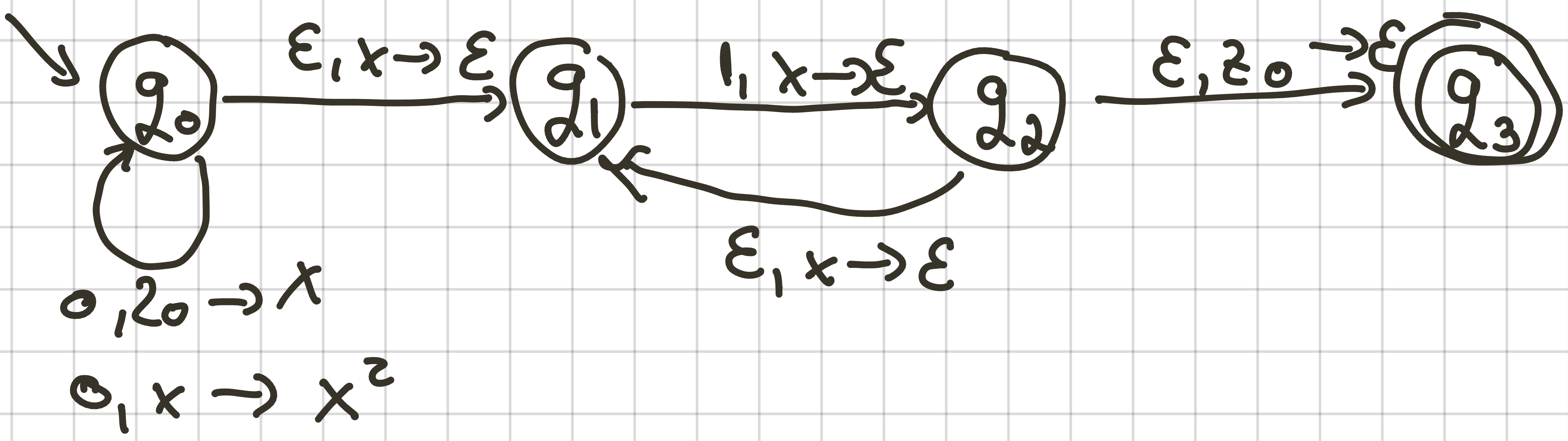
$\Rightarrow$ Sequence not accepted.

Ex 3: Tot L_1, $0^n 1^{2n}$



Ex 4:

$L_2 = \{0^{2n} 1^n \mid n \in \mathbb{N}^*\}$



Hw:

$L_3 = \{w, w^R \mid w \in \{a, b\}^+\}$

$L_4 = \{ 0^n 1^m 2^n \mid n, m \in \mathbb{N}^*$

$L_5 = \{ 0^n \ 1^m \mid m, n \in \mathbb{N}^*, \ m > n\}$.

Seminar 13

Attribute Grammars:

(G, A, R)

Context free grammar — G

Attributes — A

Set of rules. — R

Ex1: Define a Attr. grammar to compute the no of vowels in a letter string.
non empty

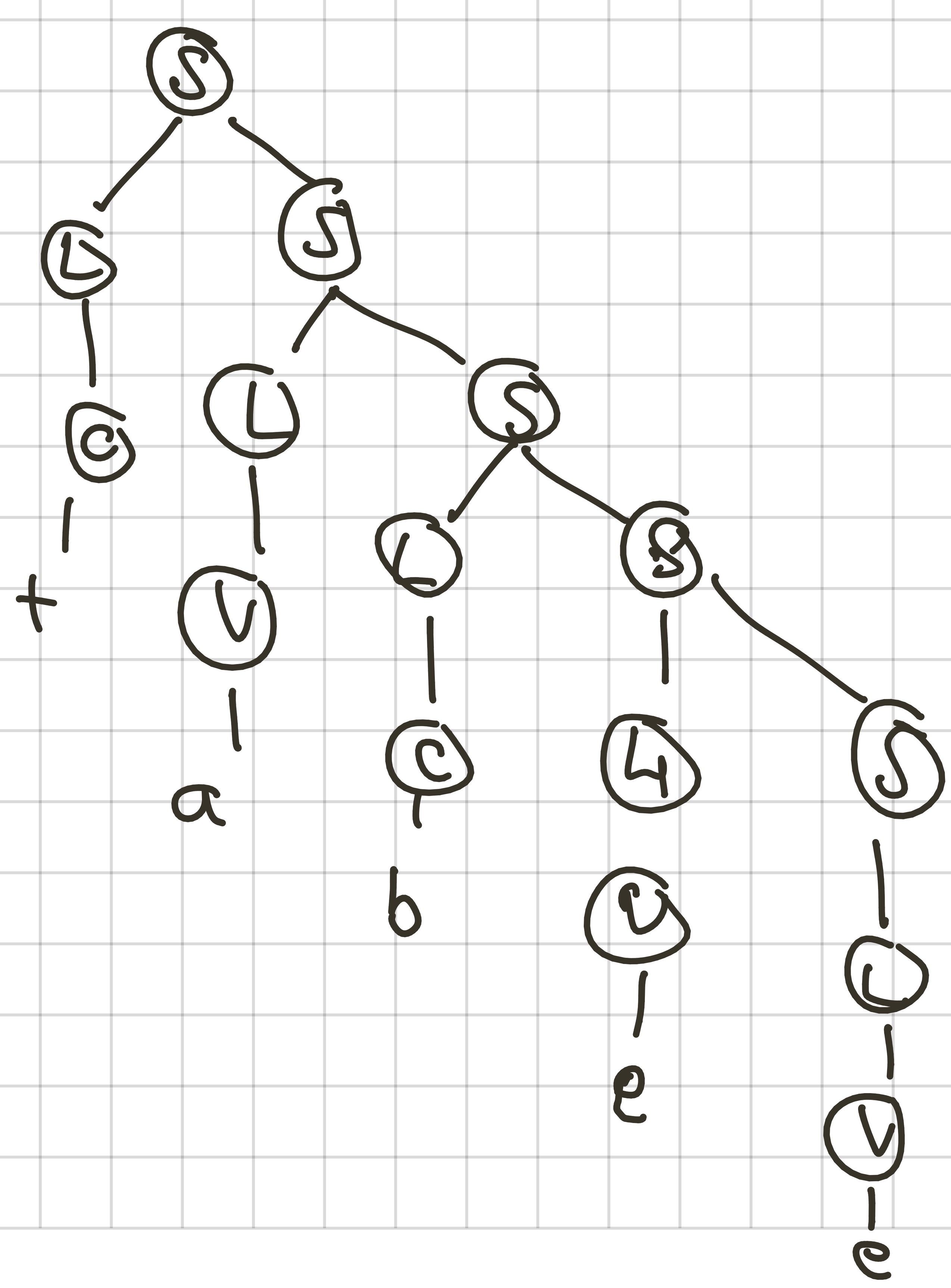
1. Define a CFG for a letter string.

G = ({

no. is the chosen attribute

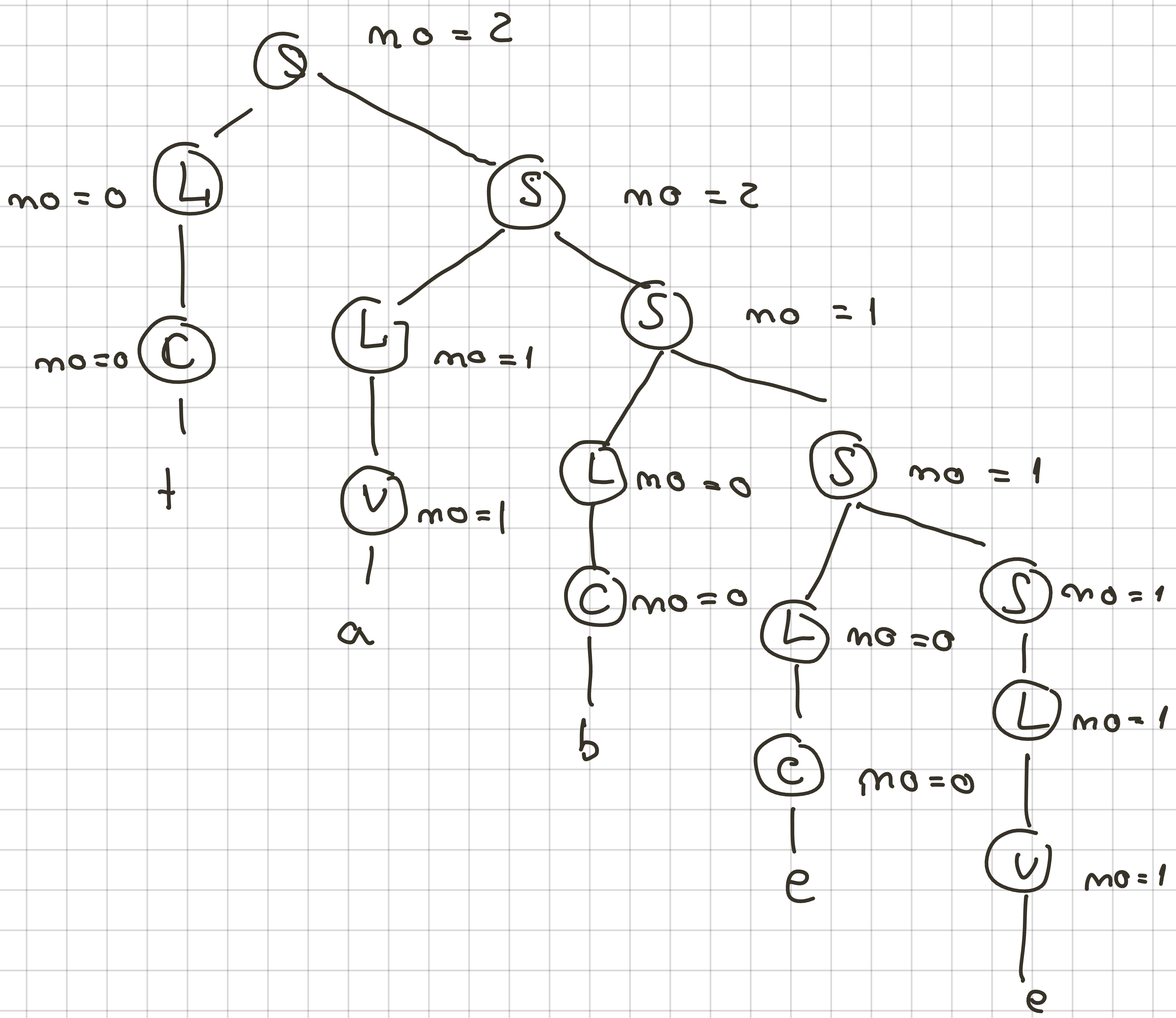
P:	$S \to L$	$\{S.no = L.no\}$
	$S \to SL$	$\{S_1.no = S_2.no + L.no$
	$L \to V$	$\{L.no = V.no\}$
	$L \to C$	$\{L.no = C.no$
	$V \to a \mid e \mid i \mid o \mid u$	$\{V.no = 1\}$
	$C \to b \mid c \mid d \ldots$	$\{C.no = 0\}$

w = table



$G = (\{S, L, V, C\}, \{a \dots z\}, P, S)$

w = table



Ex 2: Compute the value of an attribute exp $(+, -, *, /, (,))$

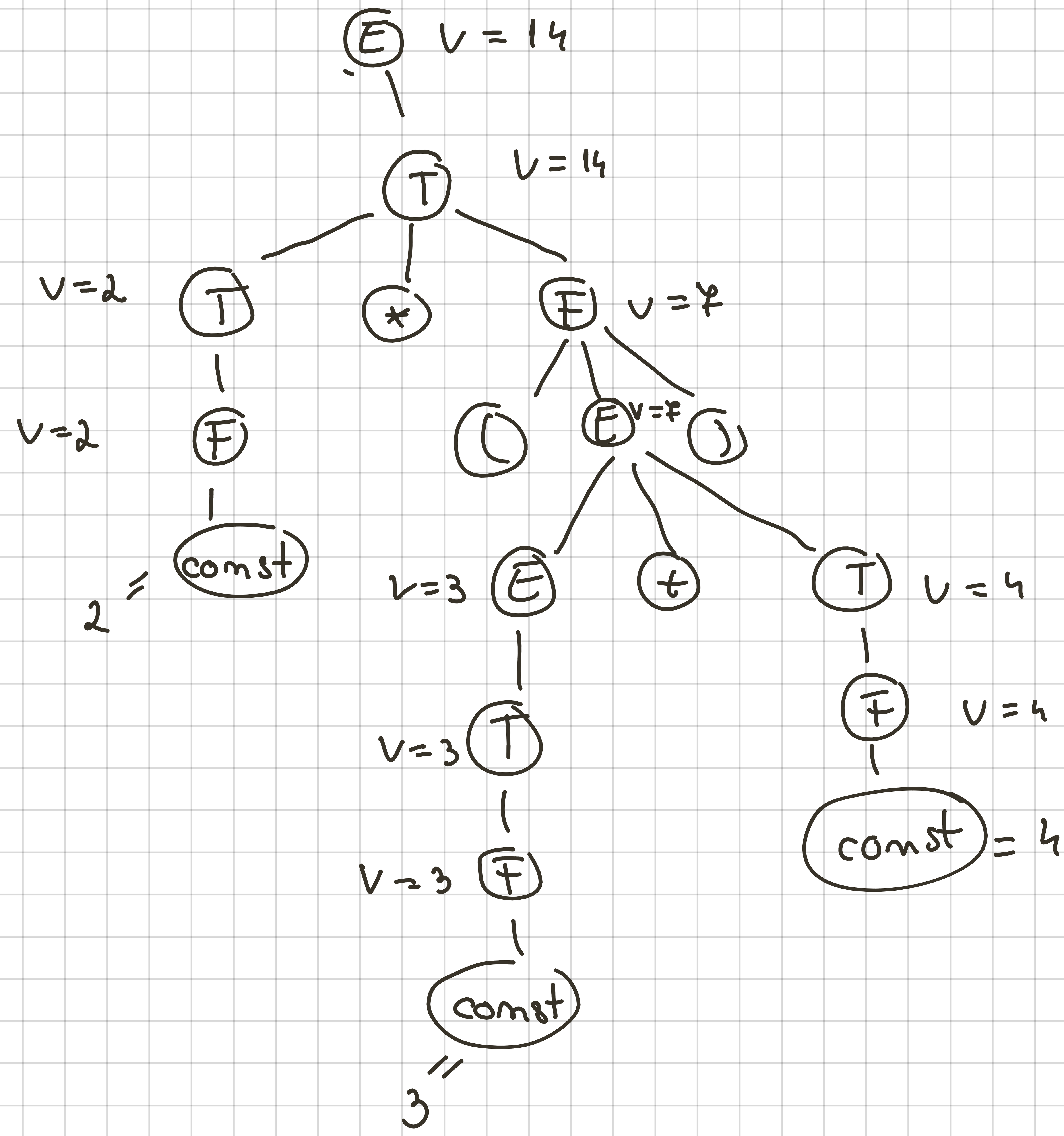
$G = ( )$

Ar fi bine de reținut asta

P:

$E \to E + T$	$E_1 . v = E_2 . v + T . v$
$E \to E - T$	$E_1 . v = E_2 . v - T . v$
$E \to T$	$E . v = T . v$
$T \to T * F$	$T_1 . v = T_2 . v * F . v$
$T \to T / F$	$T_1 . v = T_2 . v / F . v$
$T \to F$	$T . v = F . v$
$F \to (E)$	$F . v = E . v$
$F \to const$	$F . v = const . v$

w : 2 * (3 + 4)



Ex 3: if a no is divisible by 3

s = sum of digit div

$G = (\ )$

P: $N \to Z$	$N.s = Z.s$	
$N \to D$	$N.s = D.s$	
$N \to DS$	$N.s = D.s + S.s$	$N.div = N.s : 3 = 0$
$N \to ZS$	$N.s = Z.s + S.s$	
$S \to D$	$S.s = D.s$	
$S \to Z$	$S.s = Z.s$	
$S \to DS$	$S_1.s = D.s + S_2.s$	
$S \to ZS$	$S_1.s = Z.s + S_2.s$	
$Z \to 0$	$Z.s = 0$	
$D \to 1$	$D.s = 1$	
$D \to 2$	$D.s = 2$	
....		
$D \to 9$	$D.s = 9$	

TBC ....

3 Address code

1. if $(a > f)$ OR c AND $(d > e)$
   then $a \leftarrow -1$
   else $a \leftarrow b * c + 4$
   endif

2. while $(a < b)$ do
   $a \leftarrow a + 1$
   $b \leftarrow b * b$
   end while

3. for $i \leftarrow 1, n,$ do
   $a \leftarrow a + i$
   end for

1.

no	op	arg1	arg2	result
1	>	a	b	t_1
2	>	d	e	t_2
3	AND	c	t_2	t_3
4	OR	t_1	t_3	t_4
5	goto	t_4		(10)
6	*	b	c	t_5
7	+	t_5	4	t_6
8	←	t_6		a
9	goto			(12)
10	@	1		t_7
11	←	t_7		a
12				

2.

no	op	arg1	arg2	result
1	<	a	b	t_1
2	!	t_1		t_2
3	GOTO	t_2		(9)
4	+	a	1	t_3
5	←	t_3		a
6	*	b	b	t_4
7	←	t_4		b
8	GOTO			(1)
9				

3.

no	op	arg 1	arg 2	result
1	←	1		i
2	>=	i	n	t_1
3	!	t_1		t_2
4	GOTO	t_2		(9)
5	+	a	i	t_3
6	←	t_3		a
7	+	i	1	i
8	GOTO			(2)
9				