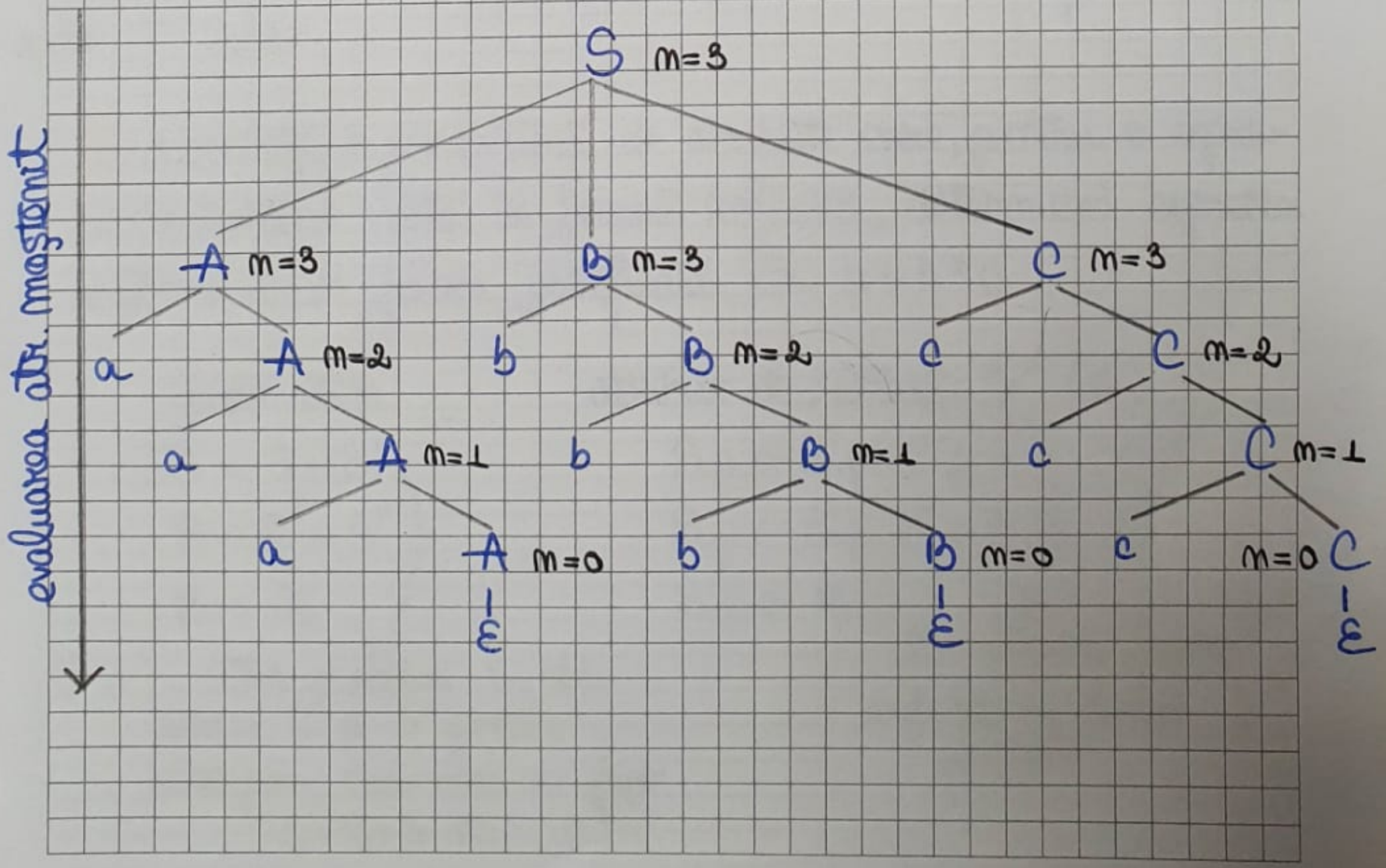
Seminar 13 LFTC

1. Fie limbajul $L = \{a^n b^n c^n \mid n$ - natural $\}$.
Dați o gramatică de atribute care îl generează.

Gramatica: atribut moștenit "m" (→):

$S \to ABC$	$A.m \leftarrow S.m \quad B.m \leftarrow S.m \quad C.m \leftarrow S.m$
$A \to aA'$	$A'.m \leftarrow A.m - 1$
$A \to \varepsilon$	$A.m = 0$
$B \to bB'$	$B'.m \leftarrow B.m - 1$
$B \to \varepsilon$	$B.m = 0$
$C \to cC'$	$C'.m \leftarrow C.m - 1$
$C \to \varepsilon$	$C.m = 0$

Pentru cazul particular $n=3$: (arborele de derivare)

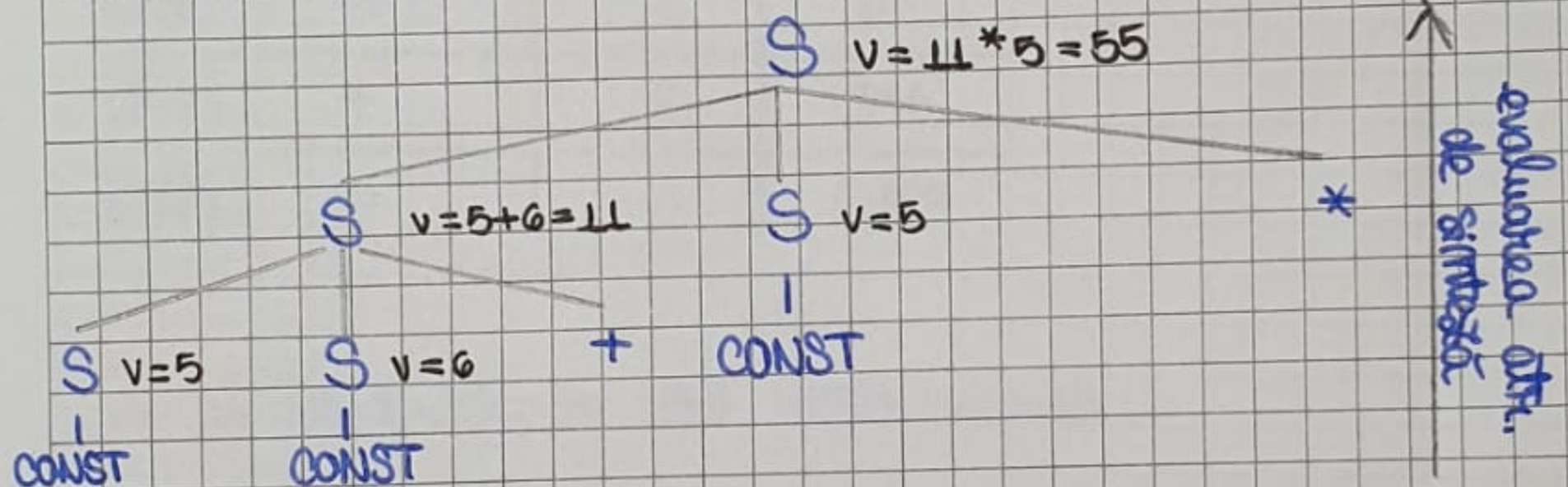


2. Descrieți o gramatică de atribute care determină valorile expresiilor aritmetice în forma postfixată
EX.: 5 6 + 5 *

Gramatica
$S \to S'S'' +$
$S \to S'S'' *$
$S \to CONST$

atribut de sinteză "v" ($\Leftarrow$)
$S.v \Leftarrow S'.v + S''.v$
$S.v \Leftarrow S'.v * S''.v$
$S.v \Leftarrow CONST$

Arborele de derivare:



3. Descrieți o gramatică de atribute care, pentru o expresie aritmetică dată în formă infixată, determină expresia aritmetică în forma postfixată. EX.: a+b*c

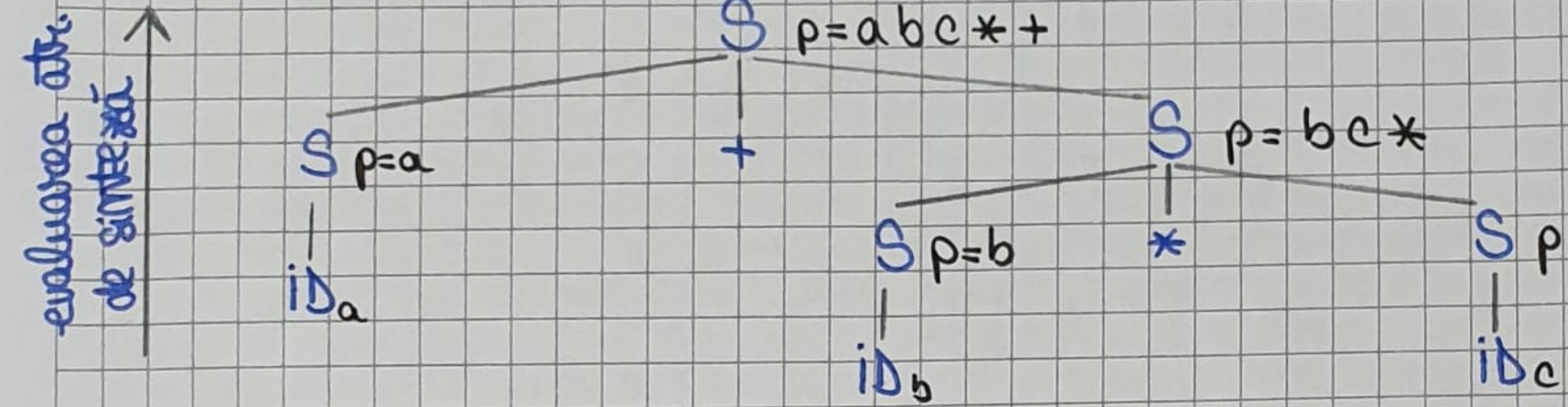
Gramatica
$S \to S' + S''$
$S \to S' * S''$
$S \to ID$

atribut de sinteză "p" ($\Leftarrow$)
$S.p \Leftarrow S'.p\ S''.p +$
$S.p \Leftarrow S'.p\ S''.p *$
$S.p \Leftarrow ID$

Obs.: e același S, dar punem S' și S'' pentru distingere (am avea de fapt $S \to S+S$ etc.)

Transformă forma infixată în forma postfixată

Arborele de derivare:



COD INTERMEDIAR

1. Fie secvența de instrucțiuni:

A := B+C*D

B := B+C*D

D := B+C*D

Traduceți în cod intermediar cu 3 adrese, reprezentare cvadruple și apoi în cod intermediar cu 3 adrese reprezentare triplete.

a) Reprezentare cvadruple:

Neoptimizat:

operator	arg1	arg2	rez
...	...	...	...
*	C	D	T1
+	B	T1	T2
:=	T2		A
*	C	D	T3
+	B	T3	T4
:=	T4		B
*	C	D	T5
+	B	T5	T6
:=	T6		D

Optimizat:

operator	arg1	arg2	rez
...	...	...	...
*	C	D	T1
+	B	T1	T2
:=	T2		A
:=	T2		B
+	B	T1	T3
:=	T3		D

b) Reprezentare Triplete:

...	operator	arg1	arg2
...	...	...	...
(340)	*	C	D
(371)	+	B	(370)
(372)	:=	A	(371)
(373)	*	C	D
(374)	+	B	(373)
(375)	:=	B	(374)
(376)	*	C	D
(374)	+	B	(376)
(378)	:=	D	(374)
...	...	...	...

2. a) Traduceți în cod intermediar:

```
a := 0;
for i := 1 to 5 do begin
        a := a + 1;
        i := i + 1;
end
```

b) Care este valoarea lui a la ieșirea din secvența de instrucțiuni? Dar valoarea lui i?

c) Dați gramatica atribuită care generează codul intermediar cu 3 adrese, reprezentare cvadruple, asociat.

d) Evaluați codul pentru ex. dat (arborele).

a) Cod intermediar cu 3 adrese, reprezentare cvadruple:

adr.	eticheta	operator	arg1	arg2	rez
...		...	...	...	...
		:=	0		a
		:=	1		i
	et_begin	g>	i	5	et_for
		+	a	1	T1
		:=	T1		a
		+	i	1	T2
		:=	T2		i
		+	i	1	i
		goto			et_begin
...		...	...	...	...
	et_for				

→ incrementarea lui i pt. for în Pascal se face în spate, ultima

b)

a	i	i>5
0	1	X
1	2	
	3	X
2	4	
	5	X
3	6	
	7	V

⇒ Răspuns: a=3
i=7

C - cod int. cu 3 adr. repr. cvadruple (atr. de sinteză)

v - variabilă

l - etichetă (label)

concatenare ↑

c) $li \to i\ ;\ li'$ — $li.c \Leftarrow i.c \parallel li'.c$

$li \to i$ — $li.c \Leftarrow i.c$

$i \to atr$ — $i.c \Leftarrow atr.c$

$i \to rep$ — $i.c \Leftarrow rep.c$

$atr \to ID := expr$ — $atr.c \Leftarrow expr.c \parallel$ | := | expr.v | | ID |

$expr \to ID$ — $expr.v \Leftarrow ID$

$expr \to CONST$ — $expr.v \Leftarrow CONST$

$expr \to expr' + expr''$ (1)

$rep \to$ for ID := CONST to CONST' do begin li end (2)

Pt. (1): $expr.v \Leftarrow$ new Temp()

$expr.c \Leftarrow expr'.c \parallel expr''.c \parallel$ | + | expr'.v | expr''.v | expr.v |

Pt. (2):

$rep.et1 \Leftarrow$ new Label() $rep.et2 \Leftarrow$ new Label()

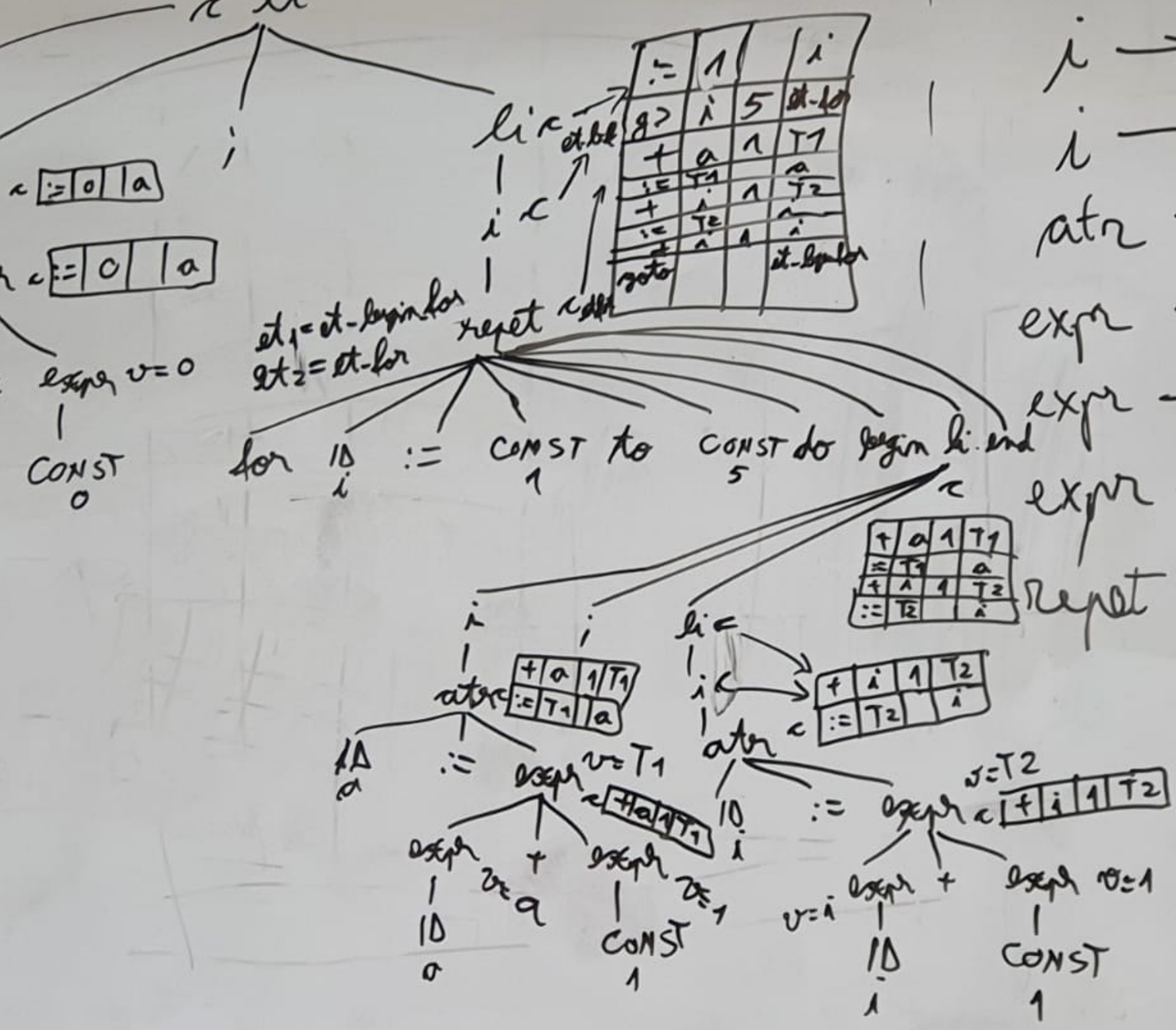
$rep.c \Leftarrow$ | := | CONST | ID | $\parallel rep.et1 \parallel$ | g> | ID | CONST' | rep.et2 | $\parallel$

$li.c \parallel$ | + | ID | 1 | ID | $\parallel$ | goto | | | rep.et1 | $\parallel rep.et2$

d) Poză :(

t: cu 3 adr. repr. cvadruple

arg 1	arg 2	rez
...	...	...
0		a
1		i
i	5	et-for
a	1	T_1
T_1		a
i	1	T_2
T_2		i
i	1	i
		et-begin-for
...	...	...



. $li \longrightarrow i;li'$

$li \longrightarrow i$

$i \longrightarrow atr$

$i \longrightarrow repet$

$atr \longrightarrow id := e$

$expr \longrightarrow id$

$expr \longrightarrow CONST$

$expr \longrightarrow expr' +$

$repet \longrightarrow for\ id :$