



Local Value Numbering \rightarrow sigur EXAMEN

\rightarrow me informă lo operare

cheie de căutare



#	val.	comparisō
0	f	f
1	#0 · #0	d
2	a	d
3	0	d
4	#1 + #2	d
5	b	d
6	2	d
7	8	d
8	#3 + #4	10
9	c	10
10	10	10
11	#1 · #5	d
12	d	d
13	e	d
14	#1 + #6	x
15	x	x
16	g	x
17	1	x
18	#7 · #8	x
19	j	x
20	#2 #7	y
21	y	y

$$\begin{aligned}
 a &:= f * f \\
 b &:= a + 0 \quad |a| \\
 c &:= 2 + 8 \quad |10| \\
 d &:= c * b \quad |10 \cdot a| \\
 e &:= f * f \quad |a| \\
 x &:= e + d \quad |a + d| \\
 g &:= b / d \quad |x| \\
 h &:= b + d \quad |x| \\
 i &:= g * \frac{1}{d} \quad |x| \\
 j &:= i \cdot 0 \quad |y| \\
 \end{aligned}$$

\rightarrow ordonare după value number

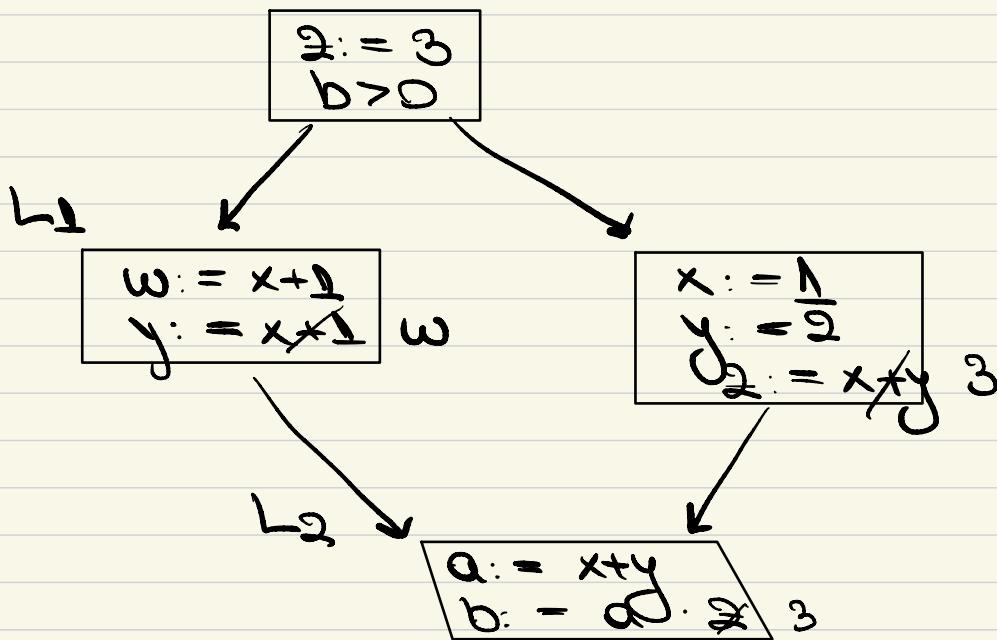
me e găzduit cu $g \neq 0 \Rightarrow$ me se poate face simplificare algebraică



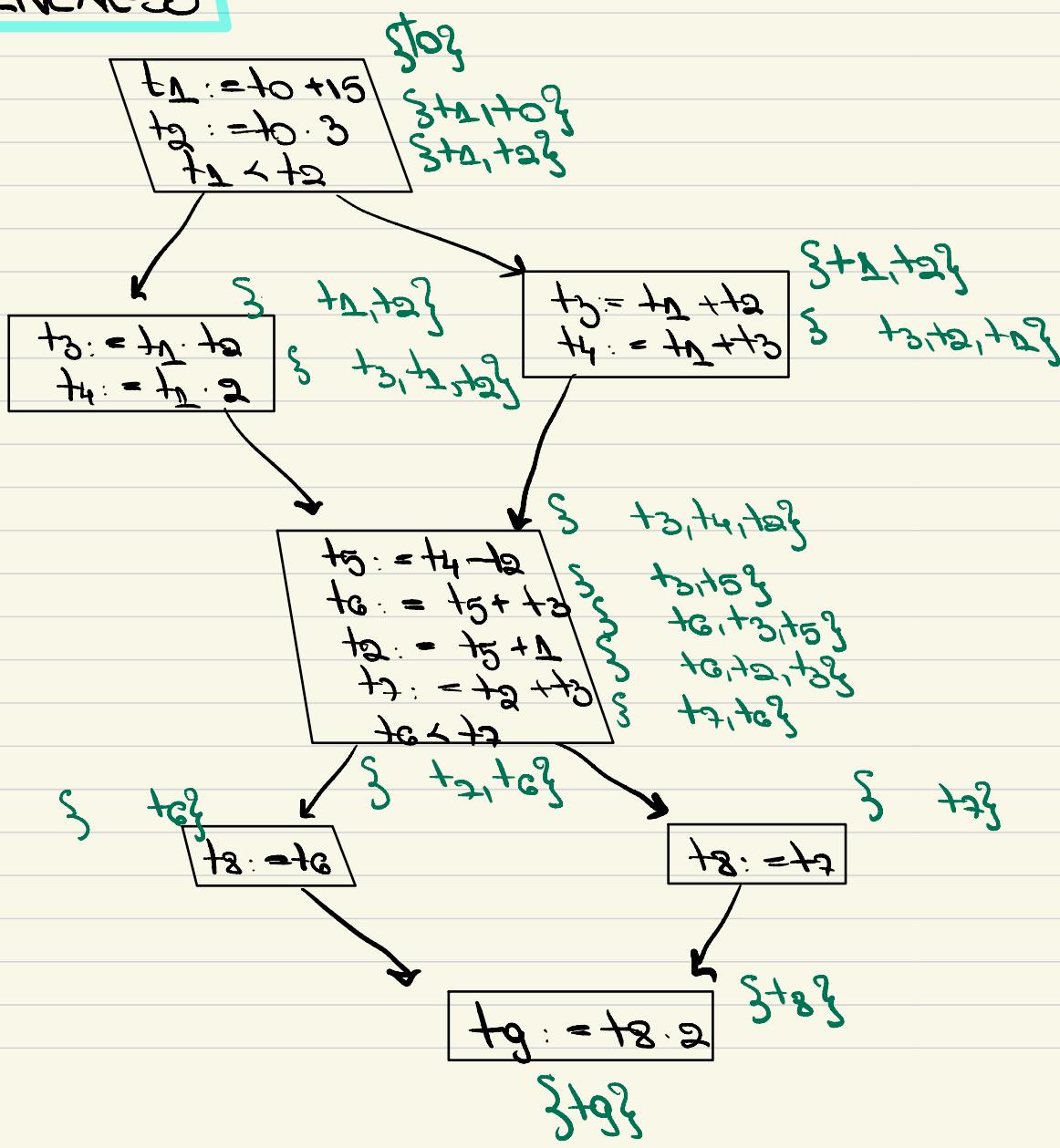
Dacă am avea 2 definiții \Rightarrow redemunirea variabilei

(exprat. comparisō cu f. variabile
redemunire)

! PROPAGAREA CONSTANTELOR



! LIVENESS



ANALISER விடையளவு

- $S \rightarrow Sa \Rightarrow S \rightarrow bSa \mid bba$
 $a \rightarrow ba \Rightarrow S_1 \rightarrow aS_1 \mid cS_1 \mid \epsilon$
 $S \rightarrow S_1 \quad \Downarrow$
 $S \rightarrow bba$
 $A \rightarrow f \quad S \rightarrow bT S_1$
 $T \rightarrow S \mid bT \quad S_1 \rightarrow aS_1 \mid cS_1 \mid \epsilon$

- $S \rightarrow \% aT \mid U!$
 $T \rightarrow aS \mid baT \mid \epsilon$
 $U \rightarrow \# aTU \mid \epsilon$

$$\underline{\underline{First(U!) = \{\#, !\}}}$$

$$\begin{aligned} First(S) &= \{\%, \#, \epsilon\} \\ First(T) &= \{a, b, \epsilon\} \\ First(U) &= \{\#\} \end{aligned}$$

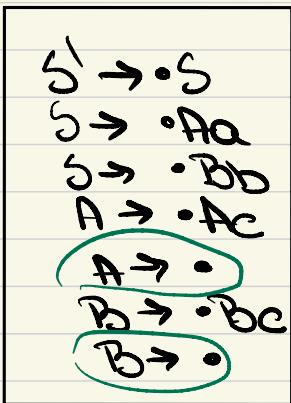
$$\begin{aligned} Follow(S) &\supseteq Follow(T) \cup \{\$\} \\ Follow(T) &\supseteq Follow(S) \\ Follow(T) &\supseteq First(U) \cup \{\$\} \\ Follow(T) &\supseteq Follow(U) \\ Follow(U) &= \{!\} \end{aligned}$$

$$\begin{aligned} Follow(S) &= Follow(T) = \{\$, !, \#\} \\ Follow(U) &= \{!\} \end{aligned}$$

	%	!	#	a	b	\$
S	% aT	U!	U!			
U		ϵ	$\# aT$			
T		ϵ	ϵ	aS	baT	ϵ

$\Rightarrow LL(1) \checkmark$

- $S \rightarrow Aa$
- $S \rightarrow Bb$
- $A \rightarrow Ac$
- $A \rightarrow \epsilon$
- $B \rightarrow BC$
- $B \rightarrow \epsilon$



$$\begin{aligned} \text{First}(S) &= \text{First}(Aa) \cup \text{First}(Bb) \\ &= \{a, b, \epsilon\} \\ \text{First}(A) &= \{\epsilon, c\} \\ \text{First}(B) &= \{\epsilon, c\} \end{aligned}$$

$$\begin{aligned} \text{Follow}(S) &= \{\$\} \\ \text{Follow}(A) &= \{b, c\} \\ \text{Follow}(B) &= \{b, c\} \end{aligned}$$

RR : $\text{Follow}(A) \cap \text{Follow}(B) \neq \emptyset \Rightarrow$ gram conflict

! REGULI DE TIPIRE

Class A {

i : Int;
j : Int;
k : Int;
foo : Bool;
foo() : SELF-TYPE { self };
bar(x : Int) : Int { if x <= k then k ← i + k else k ← j fi },

},

Class B inherits A {

p : SELF-TYPE;
test() : Object { (*Placeholder c]*) };

},

care este continutul de tipare a placeholder-ului in acest caz?

O, M, C

O

M: time const de pt. din program

deoarece sună un param. formal care are același nume ca un obiect → se adaugă în dreapta său

$$O = [i : \text{Int}, j : \text{Int}, b : \text{Bool}, p : \text{SELF-TYPE}_B, \text{self} : \text{SELF-TYPE}_B]$$

$$M(A, \text{foo}) = (\text{SELF-TYPE}) ; M(A, \text{bar}) = (\text{Int}, \text{Int}) ; M(B, \text{test}) = (\text{Object}),$$

deoarece sună un nume de metodă

$$M(B, \text{foo}) = (\text{SELF-TYPE}) ; M(B, \text{bar}) = (\text{Int}, \text{Int})$$

metodă numită din

$$\left\{ \begin{array}{l} O(p) = \text{SELF-TYPE}_B \\ O, M, B \vdash p : \text{SELF-TYPE}_B \end{array} \right. \rightarrow \text{Tiparea apelurilor}$$

$$O(j) = \text{Int}$$

$$5 \text{ pt. să fie}$$

$$O, M, B \vdash 5 : \text{Int}$$

$$\text{Int} \leq \text{Int}$$

$$O, M, B \vdash j \leftarrow 5 : \text{Int}$$

→ tiparea parametrului

$$T_0' = B$$

→ deoarece în ceea ce contează metoda

$$M(B, \text{bar}) = (\text{Int}, \text{Int})$$

$$\text{Int} \leq \text{Int}$$

→ doar tip param. actual vs. param. formal

$$T_{m+1} \leq \text{Int}$$

$$O, M, B \vdash p.\text{bar}(j \leftarrow 5) : \text{Int}$$

regula de tipare pt. dispatch



Dacă funcția apelată are fișătorul SELF-TYPE ⇒

⇒ se consideră tipul fișătorului de apel: tipul obiectului apelant



Când se implementează SELF-TYPE ca un tip concret?

→ se aplică de metode (dispatch)



GENERARE DE COD

Class X {

```
foo(): Foo {  
    bar(): Int {  
        b: Bool;  
        c: String;  
    };
```

Class Y inherits X {

```
bar(): Int {  
    base(): Object {  
        d: Int;  
    };
```

Class Z inherits Y {

```
foo(): Foo {  
    qux(): Int {  
        a: Object;  
    };
```

Reprez. clase în memorie.

X:

```
tag 0  
dim. 5 // util pt. garbage collection  
disp. pointer → Obj. copy  
b          Obj. type - name  
c          Obj. about  
x. foo      x. foo  
x. bar      x. bar
```

Y:

```
tag 1  
dim. 6  
disp. p. → Obj. × 3  
b          x. foo  
c          Y. base (pt. că e supradominată)  
d          Y. base
```

Z:

```
tag 2  
dim. 7  
disp. p. → Obj. × 3  
b          2. foo  
c          Y. bar  
d          Y. base  
a          2. quex
```

convenția de numire a metodelor
în Java

Main.f:

prolog

\$a0 pe

stiva

param. 1

\$t1 = 0

param. 2

\$t2 = 0

param. 3

\$t3 = 0

param. 4

\$t4 = 0

param. 5

\$t5 = 0

param. 6

\$t6 = 0

param. 7

\$t7 = 0

param. 8

\$t8 = 0

param. 9

\$t9 = 0

param. 10

\$t10 = 0

param. 11

\$t11 = 0

param. 12

\$t12 = 0

param. 13

\$t13 = 0

param. 14

\$t14 = 0

param. 15

\$t15 = 0

param. 16

\$t16 = 0

param. 17

\$t17 = 0

param. 18

\$t18 = 0

param. 19

\$t19 = 0

param. 20

\$t20 = 0

param. 21

\$t21 = 0

param. 22

\$t22 = 0

param. 23

\$t23 = 0

param. 24

\$t24 = 0

param. 25

\$t25 = 0

param. 26

\$t26 = 0

param. 27

\$t27 = 0

param. 28

\$t28 = 0

param. 29

\$t29 = 0

param. 30

\$t30 = 0

param. 31

\$t31 = 0

param. 32

\$t32 = 0

param. 33

\$t33 = 0

param. 34

\$t34 = 0

param. 35

\$t35 = 0

param. 36

\$t36 = 0

param. 37

\$t37 = 0

param. 38

\$t38 = 0

param. 39

\$t39 = 0

param. 40

\$t40 = 0

param. 41

\$t41 = 0

param. 42

\$t42 = 0

param. 43

\$t43 = 0

param. 44

\$t44 = 0

param. 45

\$t45 = 0

param. 46

\$t46 = 0

param. 47

\$t47 = 0

param. 48

\$t48 = 0

param. 49

\$t49 = 0

param. 50

\$t50 = 0

param. 51

\$t51 = 0

param. 52

\$t52 = 0

param. 53

\$t53 = 0

param. 54

\$t54 = 0

param. 55

\$t55 = 0

param. 56

\$t56 = 0

param. 57

\$t57 = 0

param. 58

\$t58 = 0

param. 59

\$t59 = 0

param. 60

\$t60 = 0

param. 61

\$t61 = 0

param. 62

\$t62 = 0

param. 63

\$t63 = 0

param. 64

\$t64 = 0

param. 65

\$t65 = 0

param. 66

\$t66 = 0

param. 67

\$t67 = 0

param. 68

\$t68 = 0

param. 69

\$t69 = 0

param. 70

\$t70 = 0

param. 71

\$t71 = 0

param. 72

\$t72 = 0

param. 73

\$t73 = 0

param. 74

\$t74 = 0

param. 75

\$t75 = 0

param. 76

\$t76 = 0

param. 77

\$t77 = 0

param. 78

\$t78 = 0

param. 79

\$t79 = 0

param. 80

\$t80 = 0

param. 81

\$t81 = 0

param. 82

\$t82 = 0

param. 83

\$t83 = 0

param. 84

\$t84 = 0

param. 85

\$t85 = 0

param. 86

\$t86 = 0

param. 87

\$t87 = 0

param. 88

\$t88 = 0

param. 89

\$t89 = 0

param. 90

\$t90 = 0

param. 91

\$t91 = 0

param. 92

\$t92 = 0

param. 93

\$t93 = 0

param. 94

\$t94 = 0

param. 95

\$t95 = 0

param. 96

\$t96 = 0

param. 97

\$t97 = 0

param. 98

\$t98 = 0

param. 99

\$t99 = 0

param. 100

\$t100 = 0

param. 101

\$t101 = 0

param. 102

\$t102 = 0

param. 103

\$t103 = 0

param. 104

\$t104 = 0

param. 105

\$t105 = 0

param. 106

\$t106 = 0

param. 107

\$t107 = 0

param. 108

\$t108 = 0

param. 109

\$t109 = 0

param. 110

\$t110 = 0

param. 111

\$t111 = 0

param. 112

\$t112 = 0

param. 113

\$t113 = 0

param. 114

\$t114 = 0

param. 115

\$t115 = 0

param. 116

\$t116 = 0

param. 117

\$t117 = 0

param. 118

\$t118 = 0

param. 119

\$t119 = 0

param. 120

\$t120 = 0

param. 121

\$t121 = 0

param. 122

\$t122 = 0

param. 123

\$t123 = 0

param. 124

\$t124 = 0

param. 125

\$t125 = 0

param. 126

\$t126 = 0

param. 127

\$t127 = 0

param. 128

\$t128 = 0

param. 129

\$t129 = 0

param. 130

\$t130 = 0

param. 131

\$t131 = 0

param. 132

\$t132 = 0

param. 133

\$t133 = 0

param. 134

\$t134 = 0

param. 135

\$t135 = 0

param. 136

\$t136 = 0

param. 137

\$t137 = 0

param. 138

\$t138 = 0

param. 139

class Main {

a : Int; } pendere că se face suma între ele

f(x : Int, y : Int)

if 0 < x then a + a + y else 0 fi;

self;

} ;

} ;



GARBAGE COLLECTION

- fără semnificație operatională

• class C {

new : C;

set(m : C) : C {

}

next ← m;

self;

}

} ;

class Main {

main() : Int {

let x : C in {

let o₁ : C ← new C, o₂ : C ← new C, o₃ : C ← new C,
o₄ : C ← new C, o₅ : C ← new C, o₆ : C ← new C in

}

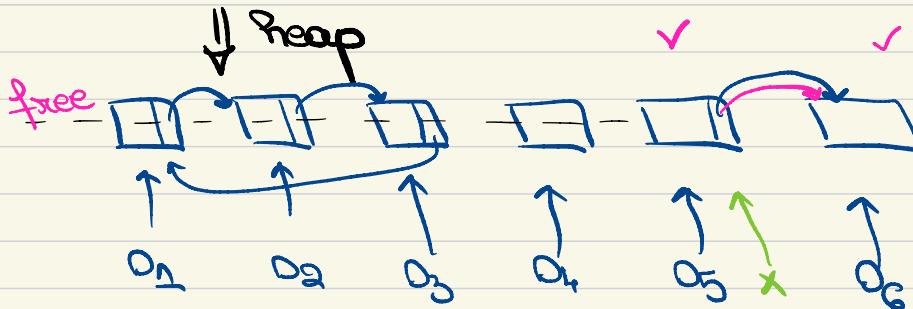
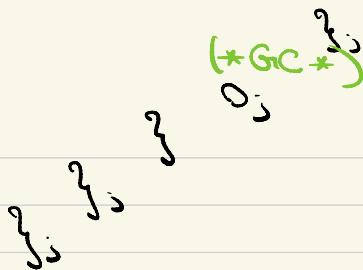
o₁. set(o₂);

o₂. set(o₃);

o₃. set(o₄);

o₄. set(o₅);

x ← o₅; ⇒ ⚡ o₀ : o₅



1. **Mark & Sweep**
- mark \Rightarrow obiecte reachable
 - \rightarrow roots $\xrightarrow{\text{roots}} \$a_0 : O_5$
 $\xrightarrow{\$sp \text{ (stiva)}} \text{în pct GC pe stivă e deasupra } \times$
 - sweep \Rightarrow se înșiruiează prin tot heap-ul și marșalibitul „marked”

Câte obiecte vizibile sunt \Rightarrow toate din heap = 6
 aleocate pînă în acel moment

2. **Stop & Copy** \rightarrow se împarte heap în 2
- old space: $\frac{3}{2}$ obiecte
 - new space: $\frac{3}{2}$ obiecte
- \rightarrow aleacțioane se fac înoldobucura în old space
- \rightarrow formare din roots
- \rightarrow mutarea obiectelor reachable în new space + actualizarea referințelor

\Rightarrow se vizitează și obiecte

3. **Reference Counting** \Rightarrow pt. fiecare obiect se adresează un număr de referințe

\Rightarrow când nr. de ref. = 0 \Rightarrow se eliberează obiectul
 \Rightarrow nu se eliberează obiectele memoriei pînă când se va umple heap-ul

Câte obiecte rămân pe heap în punctul *GC*?

5 (O_1, O_2, O_3, O_5, O_6)
 \rightarrow obiectele referite de ele