Environment ρ

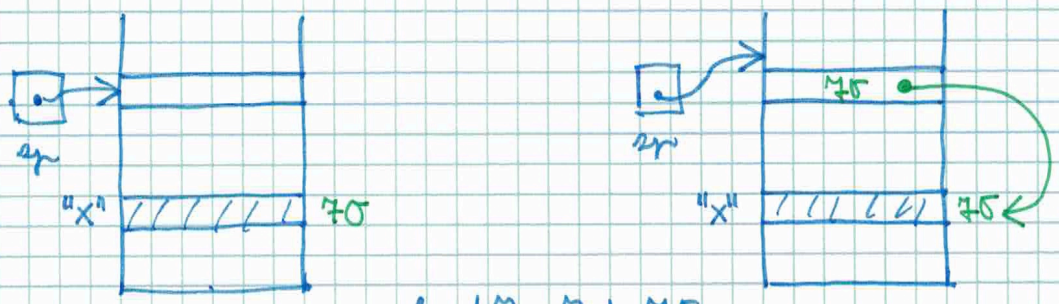
ρ : Ident ⟶ Addr × Scope × Access

Ident: Name des Speicherplatzes
Addr: Adresse (relativ oder absolut) des Speichers
Scope: Global oder Local
Access: Direct oder Indirect

Beispiele
ρ("x") = (70, Global, Direct)
ρ("y") = (−3, Local, Direct)
ρ("z") = (−4, Local, Indirect)
// Global Indirect gibt es nicht in IML

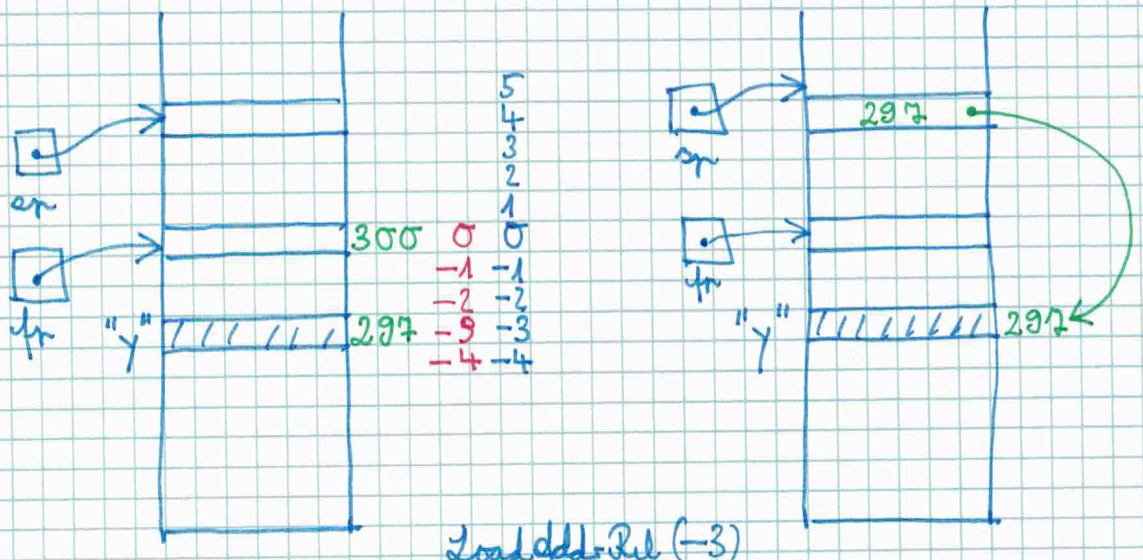
□
// Postcondition: Address of ident is on Top of Stack (TOS)
codeLExp ρ ⟦ ident ⟧ =
let (addr, scope, access) = ρ(ident)
if scope = Global ∧ access = Direct →
LoadImInt addr
if scope = Local ∧ access = Direct →
LoadAddrRel addr
if scope = Local ∧ access = Indirect →
LoadAddrRel addr ; Deref

① $\rho("x") = (75, \text{global}, \text{Direct})$



LoadImmInt 75

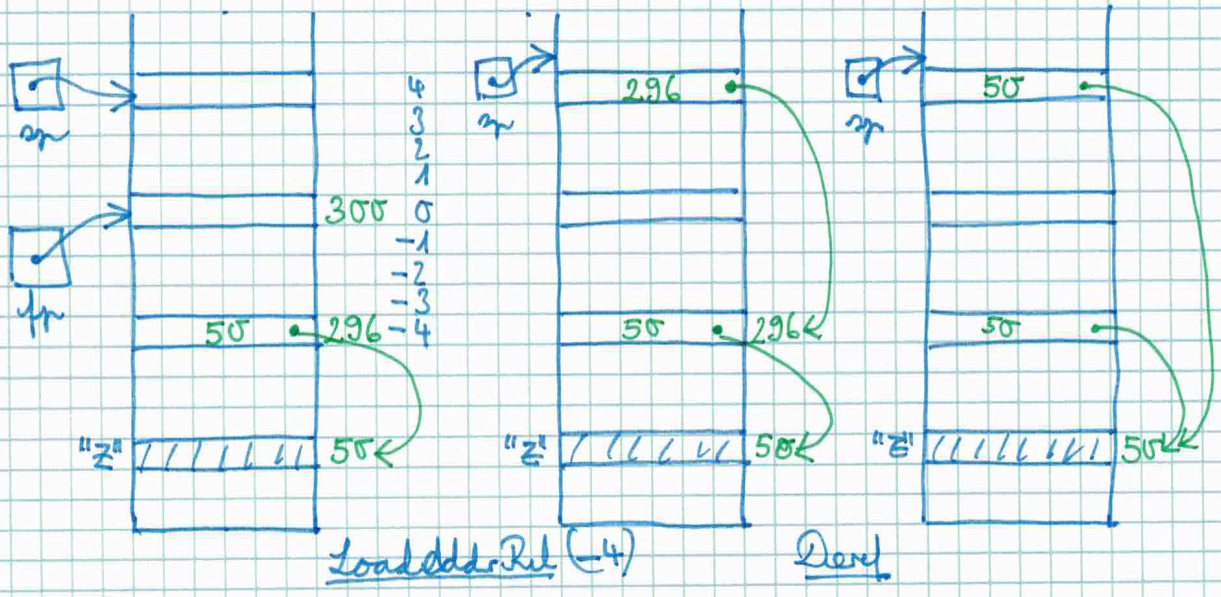
② $\rho("y") = (-3, \text{local}, \text{Direct})$



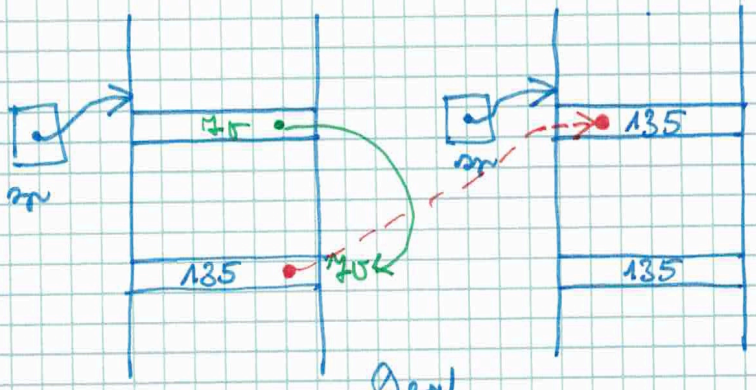
LoadAddrRel (-3)

grün : absolut
blau : relativ

(3) $\rho("z") = (-4, \text{Local}, \text{Indirekt})$



Deref



Deref

codeR Exp ρ ⟦ ident ⟧ =

codeL Exp ρ ⟦ ident ⟧ ; Deref

codeR Exp [ρ] ⟦ literal ⟧ =

LoadImmInt literal if type(literal) = INT

LoadImmInt (bool2int(literal))
if type(literal) = BOOL

codeR Exp ρ ⟦ exp1 op exp2 ⟧ =

codeR Exp ρ ⟦ exp1 ⟧ ;
codeR Exp ρ ⟦ exp2 ⟧ ;
{ MultInt if op = TIMES
⋮
SubInt if op = MINUS
⋮

Achtung: /\? \/?

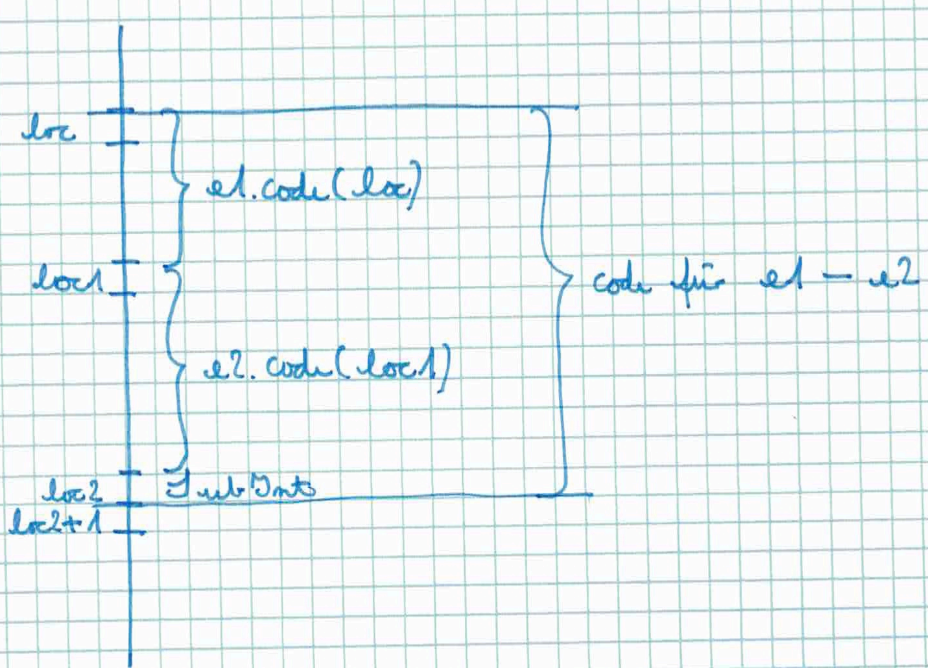
"Kurzschlussauswertung"
bedingte Auswertung
→ mit Jumps

Bsp Sei $\rho(\text{"x"}) = (70, \text{Global}, \text{Direct})$

$$
\begin{aligned}
& \text{codeCmd}\ \rho\ [\![x := x+1]\!] \\
= {} & \text{codeLExp}\ \rho\ [\![x]\!]\ ;\ \text{codeRExp}\ \rho\ [\![x+1]\!]\ ;\ \underline{\text{Store}} \\
= {} & \underline{\text{LoadImmInt}}\ 70\ ;\ \text{codeRExp}\ \rho\ [\![x]\!]\ ;\ \text{codeRExp}\ \rho\ [\![1]\!]\ ;\ \underline{\text{addInt}}\ ;\ \underline{\text{Store}} \\
= {} & \underline{\text{LoadImmInt}}\ 70\ ;\ \text{codeLExp}\ \rho\ [\![x]\!]\ ;\ \underline{\text{Deref}}\ ;\ \underline{\text{LoadImmInt}}\ 1\ ;\ \underline{\text{addInt}}\ ;\ \underline{\text{Store}} \\
= {} & \underline{\text{LoadImmInt}}\ 70\ ;\ \underline{\text{LoadImmInt}}\ 70\ ;\ \underline{\text{Deref}}\ ;\ \underline{\text{LoadImmInt}}\ 1\ ;\ \underline{\text{addInt}}\ ;\ \underline{\text{Store}}
\end{aligned}
$$

□

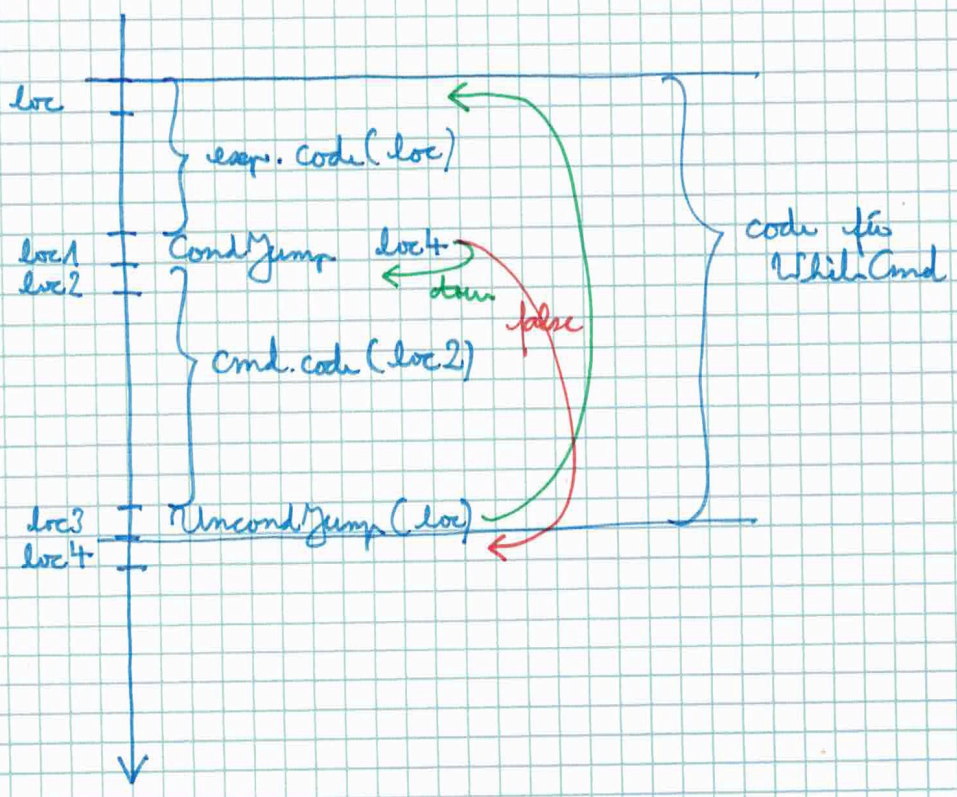
```
class AST.DyadicExpr implements AST.IExpr {
    Opr opr
    AST.IExpr e1
    AST.IExpr e2
          ← nächste freie Position      ← erste freie Position in Code-Array
    int      code(int loc) {
        int loc1 =   e1.code(loc)
        switch (opr) {
            case MINUS:
                int loc2 =   e2.code(loc1)
                codeArray.put(loc2, new SubInt());
                return loc2 + 1
            case TIMES:
                :
        }     →  /\?       \/?
    }
}
```



```
class AS.WhileCmd implements AS.ICmd {

    AS.IExpr expr // guard
    AS.ICmd cmd // body

    int code(int loc) {
        int loc1 = expr.code(loc)
        int loc2 = loc1 + 1
        int loc3 = cmd.code(loc2)
        int loc4 = loc3 + 1
        code.Array.put(loc1, new CondJump(loc4))
        code.Array.put(loc3, new UncondJump(loc))
        return loc4
    }
}
```



cpit HS 2021 14.12.2021

Code-Erzeugung