switch(E)
{case L1:S1 .... case Ln:Sn case default: Sd}

include these keywords

Last example of operational semantics by intermediate code:

One-level function calls - the called function can't call any other functions.

function definition

void f(x_1,...., x_n)
{ S}
start_f: code to execute S
⋮

function call

f(E_1,..., E_n) E_1, ... E_n are expressions

call by value

Code to evaluate E_1
x_1 = result of above evaluation
⋮

Inside "code to execute S",
each occurrence of
"return" statement is
translated into
"goto @ returnLabel"

Indirect goto statement
goto the label contained in
the variable "returnLabel"

Code to evaluate E_n
x_n = result of above evaluation
returnLabel = "ReturnToHere" //save return address
goto start_f
ReturnToHere:

- Structured operational statements (some books categorize this under denotational semantics)
- Step-by-step operation of statements is described by a state-transition function.
  - Need to define the program states

program state α → statement S → program state $\beta = M(S, \alpha)$

Definition of Program States

compile time ↑

static variables ← we'll only treat this - memory for variables are allocated at
function local variables & parameters → inside runtime stack
heap variables - e.g. dynamically created class objects

Represent all static variables by $x_1, \ldots x_n$

These include components of structured data like arrays and class objects.

Array of 10 elements $x_i, x_{i+1}, \ldots, x_{i+9}$ for some i

Class objects of 10 fields $x_j, x_{j+1}, \ldots, x_{j+9}$ for some j

A program state is defined as

$\{\langle x_1, v_1\rangle, \ldots, \langle x_n, v_n\rangle\}$ — a "snapshot" of all vars

Each pair $\langle x_i, v_i\rangle$ represents the fact that the current value of x_i is v_i

Special Values

$\perp$ ("bottom") = the undefined value

$\langle x_i, \perp\rangle$: variable x_i currently has no assigned value.

$\perp_V$ = the runtime error resulting from any kind of runtime errors in expression evaluation.

eg = arithmetic operations on variables with $\perp$ value, arithmetic overflow, division by 0.

$\perp_S$ = the error state resulting from any kind of runtime error

To define the state-transition function $M(S, \alpha)$, we need the auxiliary function

Eval (E, α) for expression evaluation

program state α → expression E → value of E = Eval (E, α)

Example of Eval = arithmetic expressions

$\perp + x = x + \perp = \perp + \perp = \perp_V$ for any value of x

$\perp_V + x = x + \perp_V = \perp_V + \perp_V = \perp_V$ for any value of x

Analogously for $-, *, /$.

Eval (c, α) = the value of c, for any constant literal c

Eval $(x_i, \{\langle x_i, v_i\rangle, \ldots, \langle x_n, v_n\rangle\}) = v_i$ for any variable x_i

$Eval(E_1 + E_2, \alpha) = Eval(E_1, \alpha) + Eval(E_2, \alpha)$

$Eval(E_1 - E_2, \alpha) = Eval(E_1, \alpha) - Eval(E_2, \alpha)$

$Eval(E_1 * E_2, \alpha) = Eval(E_1, \alpha) * Eval(E_2, \alpha)$

$Eval(E_1 / E_2, \alpha) = Eval(E_1, \alpha) / Eval(E_2, \alpha)$

$Eval(-E, \alpha) = -Eval(E, \alpha)$

$\alpha = \{\langle x_1, 1\rangle, \langle x_2, 2\rangle, \langle x_3, 3\rangle, \langle x_4, \perp\rangle\}$

$Eval(2 + x_2 * x_3, \alpha) = Eval(2, \alpha) + Eval(x_2 * x_3, \alpha)$

$= 2 + Eval(x_2 * x_3, \alpha)$

$= 2 + Eval(x_2, \alpha) * Eval(x_3, \alpha)$

$= 2 + 2 * Eval(x_3, \alpha)$

$= 2 + 2 * 3$

$= 8$

$Eval(x_3 + x_4, \alpha) = Eval(x_3, \alpha) + Eval(x_4, \alpha)$

$= 3 + Eval(x_4, \alpha)$

$= 3 + \perp$

$= \perp_V$

$Eval(x_2 / 0, \alpha) = Eval(x_2, \alpha) / Eval(0, \alpha)$

$= 2 / Eval(0, \alpha)$

$= 2 / 0$

$= \perp_V$

Define $\triangleq$ $Eval(B_1 \,\&\&\, B_2, \alpha)$

$Eval(B_1 \,||\, B_2, \alpha)$

$Eval(!\, B, \alpha)$