Semantics of ±C)(i.e. extended C−−)

VILLANI Neven, ENS Paris-Saclay
PROGRAMMATION 1 – COMPILATEUR COCASS

17 décembre 2020

Retranscribed and extended from http://www.lsv.fr/~goubault/CoursProgrammation/minic.html and http://www.lsv.fr/~goubault/CoursProgrammation/prog1_sem1.pdf.

1 Expressions

Notation

$\mathbb{Z}_{64} \triangleq \mathbb{Z}/64\mathbb{Z}$ is the set in which all calculations are done.

We write $(\rho : \mathcal{S} \to \mathbb{Z}_{64}) \in \mathcal{P}$ the environment, where $\mathcal{S}$ is the set of names of variables and functions, $(\mu : \mathbb{Z}_{64} \to \mathbb{Z}_{64}) \in \mathcal{M}$ the memory.

A flag is defined as an element of $\mathcal{E} \triangleq S \sqcup \{\texttt{break}, \texttt{return}, \texttt{continue}, \texttt{nil}\}$.
Intuitively, $\rho, \mu, \chi, v \Vdash_\pi c \Rightarrow \rho', \mu', \chi', v'$ means that when c is executed under the environment ρ with the memory μ, the flag χ, and the previous value v, it updates it to the new environment and memory ρ' and μ', raises χ', and changes the value to v'.

In addition, we write $\mathrm{fun}_\pi^n : \mathcal{S} \to \mathcal{F}^n$ where $\mathcal{F}^n \triangleq (\mathcal{M} \times \mathcal{E} \times \mathbb{Z}_{64})^{\mathcal{P} \times \mathcal{M} \times \mathbb{Z}_{64}^n}$, i.e. functions that take an environment, a memory layout and n 64-bit integer arguments and return one 64-bit integer, the updated memory, and a flag.
$\mathrm{fun}_\mu^n : \mathbb{Z}_{64} \to \mathcal{F}^n$ returns the function (if there is one) defined at the given memory address, and is useful for variables that are function pointers.

For $\mu \in \mathcal{M}, v \in \mathbb{Z}_{64}, x \in \mathbb{Z}_{64}$ we write $\mu[x \mapsto v] : \begin{cases} x \mapsto v & \\ y \mapsto \mu(y) & y \in \mathrm{dom}\,\mu \setminus \{x\} \end{cases}$

1.1 Reading values

For local and global variables :

$$\frac{x \in \mathrm{dom}\,\rho \qquad \rho(x) \in \mathrm{dom}\,\mu}{\rho, \mu, \texttt{nil}, v \vdash_\pi \texttt{VAR}\ x \Rightarrow \rho, \mu, \texttt{nil}, \mu(\rho(x))}(\textsc{Var})$$

i.e. reading a variable returns its contents and changes nothing to the memory.

$$\frac{\chi \neq \texttt{nil}}{\rho, \mu, \chi, v \vdash_\pi \texttt{VAR}\ x \Rightarrow \rho, \mu, \chi, v}(\textsc{Var}^\chi)$$

For constant integers :

$$\frac{}{\rho, \mu, \texttt{nil}, v \vdash_\pi \texttt{CST}\ n \Rightarrow \rho, \mu, \texttt{nil}, n}(\textsc{Cst})$$

$$\frac{\chi \neq \texttt{nil}}{\rho, \mu, \chi, v \vdash_\pi \texttt{CST}\ n \Rightarrow \rho, \mu, \chi, v}(\textsc{Cst}^\chi)$$

For strings :

$$\frac{s \text{ stored at } a \in \mathit{Addr}}{\rho, \mu, \texttt{nil}, v \vdash_\pi \texttt{STRING}\ s \Rightarrow \rho, \mu, \texttt{nil}, a}(\textsc{Str})$$

$$\frac{\chi \neq \texttt{nil}}{\rho, \mu, \chi, v \vdash_\pi \texttt{STRING}\ s \Rightarrow \rho, \mu, \chi, v}(\textsc{Cst}^\chi)$$

For arrays :

$$\frac{\begin{array}{c}\rho, \mu, \chi, v \vdash_\pi i \Rightarrow \rho, \mu_i, \chi_i, v_i \\ \rho, \mu_i, \chi_i, i \vdash_\pi a \Rightarrow \rho, \mu', \texttt{nil}, v_a \\ v_a + v_i \times 8 \in \mathrm{dom}\,\mu'\end{array}}{\rho, \mu, \texttt{nil}, v \vdash_\pi \texttt{OP2}(\texttt{S\_INDEX}, a, i) \Rightarrow \rho, \mu', \texttt{nil}, \mu'(v_a + v_i \times 8)}(\textsc{Idx})$$

None of these are different from the original C−−semantics.

1.2 Unary operators without side-effects

Unary minus (same as C−−) :

$$\frac{\rho, \mu, \chi, v \vdash_\pi e \Rightarrow \rho, \mu', \texttt{nil}, v_e}{\rho, \mu, \chi, v \vdash_\pi \texttt{OP1(M\_MINUS}, e) \Rightarrow \rho, \mu', \texttt{nil}, -v_e}(\textsc{Neg})$$

Unary bitwise negation (same as C−−) :

$$\frac{\rho, \mu, \chi, v \vdash_\pi e \Rightarrow \rho, \mu', \texttt{nil}, v_e}{\rho, \mu, \chi, v \vdash_\pi \texttt{OP1(M\_NOT}, e) \Rightarrow \rho, \mu', tnil, -v_e - 1}(\textsc{Not})$$

Indirection (added in ±C)) :

$$\frac{x \in \operatorname{dom} \rho}{\rho, \mu, \texttt{nil}, v \vdash_\pi \texttt{OP1(M\_ADDR}, x) \Rightarrow \rho, \mu, \texttt{nil}, \rho(x)}(\textsc{Var}^\&)$$

$$\frac{\begin{array}{c}\rho, \mu, \chi, v \vdash_\pi i \Rightarrow \rho, \mu_i, \chi_i, v_i \\ \rho, \mu_i, \chi_i, v_i \vdash_\pi a \Rightarrow \rho, \mu', \texttt{nil}, v_a\end{array}}{\rho, \mu, \chi, v \vdash_\pi \texttt{OP1(M\_ADDR}, \texttt{OP2(S\_INDEX}, a, e)) \Rightarrow \rho, \mu', \texttt{nil}, t + i \times 8}(\textsc{Idx}^\&)$$

$$\frac{\rho, \mu, \chi, v \vdash_\pi a \Rightarrow \rho, \mu', \texttt{nil}, v_a}{\rho, \mu, \chi, v \vdash_\pi \texttt{OP1(M\_ADDR}, \texttt{OP1(M\_DEREF}, a)) \Rightarrow \rho, \mu', \texttt{nil}, v_a}(\textsc{Ptr}^\&)$$

Dereferencing (added in ±C)) :

$$\frac{\rho, \mu, \chi, v \vdash_\pi a \Rightarrow \rho, \mu', \texttt{nil}, v_a \qquad v_a \in \operatorname{dom} \mu'}{\rho, \mu, \chi, v \vdash_\pi \texttt{OP1(M\_DEREF}, a) \Rightarrow \rho, \mu', \texttt{nil}, \mu'(v_a)}(\textsc{Ptr})$$

When the operand raises a non-`nil` flag :

$$\frac{\rho, \mu, \chi, v \vdash_\pi e \Rightarrow \rho, \mu', \chi', v_e \qquad \chi' \neq \texttt{nil}}{\rho, \mu, \chi, v \vdash_\pi \texttt{OP1}(op, e) \Rightarrow \rho, \mu', \chi', v_e}(\textsc{Op1}^\chi)$$

1.3 Binary operators

Multiplication (same as C−−) :

$$\frac{\begin{array}{c}\rho, \mu, \chi, v \vdash_\pi e_2 \Rightarrow \rho, \mu_2, \chi_2, v_2 \\ \rho, \mu_2, \chi_2, v_2 \vdash_\pi e_1 \Rightarrow \rho, \mu', \texttt{nil}, v_1\end{array}}{\rho, \mu, \chi, v \vdash_\pi \texttt{OP2(S\_MUL}, e_1, e_2) \Rightarrow \rho, \mu', \texttt{nil}, v_1 \times v_2}(\textsc{Mul})$$

Addition (same as C−−) :

$$\frac{\begin{array}{c}\rho, \mu, \chi, v \vdash_\pi e_2 \Rightarrow \rho, \mu_2, \chi_2, v_2 \\ \rho, \mu_2, \chi_2, v_2 \vdash_\pi e_1 \Rightarrow \rho, \mu', \texttt{nil}, v_1\end{array}}{\rho, \mu, \chi, v \vdash_\pi \texttt{OP2(S\_ADD}, e_1, e_2) \Rightarrow \rho, \mu', \texttt{nil}, v_1 + v_2}(\textsc{Add})$$

Subtraction (same as C−−) :

$$\frac{\begin{array}{c}\rho, \mu, \chi, v \vdash_\pi e_2 \Rightarrow \rho, \mu_2, \chi_2, v_2 \\ \rho, \mu_2, \chi_2, v_2 \vdash_\pi e_1 \Rightarrow \rho, \mu', \texttt{nil}, v_1\end{array}}{\rho, \mu, \chi, v \vdash_\pi \texttt{OP2(S\_SUB}, e_1, e_2) \Rightarrow \rho, \mu', \texttt{nil}, v_1 - v_2}(\textsc{Sub})$$

Division and remainder (same as C−−) :

$$\frac{\begin{array}{c}\rho, \mu, \chi, v \vdash_\pi e_2 \Rightarrow \rho, \mu_2, \chi_2, v_2 \qquad v_2 \neq 0 \\ \rho, \mu_2, \chi_2, v_2 \vdash_\pi e_1 \Rightarrow \rho, \mu', \texttt{nil}, v_1\end{array}}{\rho, \mu, \chi, v \vdash_\pi \texttt{OP2(S\_DIV}, e_1, e_2) \Rightarrow \rho, \mu', \texttt{nil}, v_1 \text{ div } v_2}(\textsc{Div})$$

$$\frac{\begin{array}{c}\rho, \mu, \chi, v \vdash_\pi e_2 \Rightarrow \rho, \mu_2, \chi_2, v_2 \qquad v_2 \neq 0 \\ \rho, \mu_2, \chi_2, v_2 \vdash_\pi e_1 \Rightarrow \rho, \mu', \texttt{nil}, v_1\end{array}}{\rho, \mu, \chi, v \vdash_\pi \texttt{OP2(S\_MOD}, e_1, e_2) \Rightarrow \rho, \mu', \texttt{nil}, v_1 \text{ mod } v_2}(\textsc{Mod})$$

Shifts (added in ±C)) :

$$\frac{\begin{array}{c}\rho, \mu, \chi, v \vdash_\pi e_2 \Rightarrow \rho, \mu_2, \chi_2, v_2 \\ \rho, \mu_2, \chi_2, v_2 \vdash_\pi e_1 \Rightarrow \rho, \mu', \texttt{nil}, v_1\end{array}}{\rho, \mu, \chi, v \vdash_\pi \texttt{OP2(S\_SHL,}\, e_1, e_2\texttt{)} \Rightarrow \rho, \mu', \texttt{nil}, v_1 \times 2^{v_2}} (\textsc{Shl})$$

$$\frac{\begin{array}{c}\rho, \mu, \chi, v \vdash_\pi e_2 \Rightarrow \rho, \mu_2, \chi_2, v_2 \\ \rho, \mu_2, \chi_2, v_2 \vdash_\pi e_1 \Rightarrow \rho, \mu', \texttt{nil}, v_1\end{array}}{\rho, \mu, \chi, v \vdash_\pi \texttt{OP2(S\_SHR,}\, e_1, e_2\texttt{)} \Rightarrow \rho, \mu', \texttt{nil}, v_1 \text{ div } 2^{v_2}} (\textsc{Shr})$$

Let $\text{dec}_{64} : \{\bot, \top\}^{64} \to \mathbb{Z}_{64}$ the function

$$(b_0, \cdots, b_{63}) \mapsto \sum_{i=0}^{63} (1 \texttt{ if } b_i \texttt{ else } 0) \times 2^i$$

and $\text{bin}_{64} = {\text{dec}_{64}}^{-1}$.
We can now define bitwise operators as follows (added in ±C)).

$$\frac{\begin{array}{cc}\rho, \mu, \chi, v \vdash_\pi e_2 \Rightarrow \rho, \mu_2, \chi_2, v_2 & \rho, \mu_2, \chi_2, v_2 \vdash_\pi e_1 \Rightarrow \rho, \mu', \texttt{nil}, v_1 \\ (b_0^2, \cdots, b_{63}^2) = \text{bin}_{64}(v_2) & (b_0^1, \cdots, b_{63}^1) = \text{bin}_{64}(v_1)\end{array}}{\rho, \mu, \chi, v \vdash_\pi \texttt{OP2(S\_AND,}\, e_1, e_2\texttt{)} \Rightarrow \rho, \mu', \texttt{nil}, \text{dec}_{64}(b_0^1 \wedge b_0^2, \cdots, b_{63}^1 \wedge b_{63}^2), \mu''} (\textsc{And})$$

$$\frac{\begin{array}{cc}\rho, \mu, \chi, v \vdash_\pi e_2 \Rightarrow \rho, \mu_2, \chi_2, v_2 & \rho, \mu_2, \chi_2, v_2 \vdash_\pi e_1 \Rightarrow \rho, \mu', \texttt{nil}, v_1 \\ (b_0^2, \cdots, b_{63}^2) = \text{bin}_{64}(v_2) & (b_0^1, \cdots, b_{63}^1) = \text{bin}_{64}(v_1)\end{array}}{\rho, \mu, \chi, v \vdash_\pi \texttt{OP2(S\_OR,}\, e_1, e_2\texttt{)} \Rightarrow \rho, \mu', \texttt{nil}, \text{dec}_{64}(b_0^1 \vee b_0^2, \cdots, b_{63}^1 \vee b_{63}^2), \mu''} (\textsc{Ior})$$

$$\frac{\begin{array}{cc}\rho, \mu, \chi, v \vdash_\pi e_2 \Rightarrow \rho, \mu_2, \chi_2, v_2 & \rho, \mu_2, \chi_2, v_2 \vdash_\pi e_1 \Rightarrow \rho, \mu', \texttt{nil}, v_1 \\ (b_0^2, \cdots, b_{63}^2) = \text{bin}_{64}(v_2) & (b_0^1, \cdots, b_{63}^1) = \text{bin}_{64}(v_1)\end{array}}{\rho, \mu, \chi, v \vdash_\pi \texttt{OP2(S\_XOR,}\, e_1, e_2\texttt{)} \Rightarrow \rho, \mu', \texttt{nil}, \text{dec}_{64}(b_0^1 \oplus b_0^2, \cdots, b_{63}^1 \oplus b_{63}^2), \mu''} (\textsc{Xor})$$

When one of the operands raises a non-`nil` flag :

$$\frac{\begin{array}{c}\rho, \mu, \chi, v \vdash_\pi e_2 \Rightarrow \rho, \mu_2, \chi_2, v_2 \\ \rho, \mu_2, \chi_2, v_2 \vdash_\pi e_1 \Rightarrow \rho, \mu', \chi', v_1 \\ \chi' \neq \texttt{nil}\end{array}}{\rho, \mu, \chi, v \vdash_\pi \texttt{OP2(}op, e_1, e_2\texttt{)} \Rightarrow \rho, \mu', \chi', v_1} (\textsc{Op2}^\chi)$$

1.4 Comparisons

All are the same as in C−−.

$$\frac{\begin{array}{c}\rho, \mu, \chi, v \vdash_\pi e_2 \Rightarrow \rho, \mu_2, \chi_2, v_2 \\ \rho, \mu_2, \chi_2, v \vdash_\pi e_1 \Rightarrow \rho, \mu', \texttt{nil}, v_1 \\ v_1 = v_2\end{array}}{\rho, \mu, \texttt{nil}, v \vdash_\pi \texttt{CMP(C\_EQ,}\, e_1, e_2\texttt{)} \Rightarrow \rho, \mu', \texttt{nil}, 1} (\textsc{Eq}^\top)$$

$$\frac{\begin{array}{c}\rho, \mu, \chi, v \vdash_\pi e_2 \Rightarrow \rho, \mu_2, \chi_2, v_2 \\ \rho, \mu_2, \chi_2, v \vdash_\pi e_1 \Rightarrow \rho, \mu', \texttt{nil}, v_1 \\ v_1 < v_2\end{array}}{\rho, \mu, \texttt{nil}, v \vdash_\pi \texttt{CMP(C\_LT,}\, e_1, e_2\texttt{)} \Rightarrow \rho, \mu', \texttt{nil}, 1} (\textsc{Lt}^\top)$$

$$\frac{\begin{array}{c}\rho, \mu, \chi, v \vdash_\pi e_2 \Rightarrow \rho, \mu_2, \chi_2, v_2 \\ \rho, \mu_2, \chi_2, v \vdash_\pi e_1 \Rightarrow \rho, \mu', \texttt{nil}, v_1 \\ v_1 \leqslant v_2\end{array}}{\rho, \mu, \texttt{nil}, v \vdash_\pi \texttt{CMP(C\_LE,}\, e_1, e_2\texttt{)} \Rightarrow \rho, \mu', \texttt{nil}, 1} (\textsc{Le}^\top)$$

$$\frac{\begin{array}{c}\rho, \mu, \chi, v \vdash_\pi e_2 \Rightarrow \rho, \mu_2, \chi_2, v_2 \\ \rho, \mu_2, \chi_2, v \vdash_\pi e_1 \Rightarrow \rho, \mu', \texttt{nil}, v_1 \\ v_1 \neq v_2\end{array}}{\rho, \mu, \texttt{nil}, v \vdash_\pi \texttt{CMP(C\_EQ,}\, e_1, e_2\texttt{)} \Rightarrow \rho, \mu', \texttt{nil}, 0} (\textsc{Eq}^\bot)$$

$$\frac{\begin{array}{c}\rho, \mu, \chi, v \vdash_\pi e_2 \Rightarrow \rho, \mu_2, \chi_2, v_2 \\ \rho, \mu_2, \chi_2, v \vdash_\pi e_1 \Rightarrow \rho, \mu', \texttt{nil}, v_1 \\ v_1 \not< v_2\end{array}}{\rho, \mu, \texttt{nil}, v \vdash_\pi \texttt{CMP(C\_LT,}\, e_1, e_2\texttt{)} \Rightarrow \rho, \mu', \texttt{nil}, 0} (\textsc{Lt}^\bot)$$

$$\frac{\begin{array}{c}\rho, \mu, \chi, v \vdash_\pi e_2 \Rightarrow \rho, \mu_2, \chi_2, v_2 \\ \rho, \mu_2, \chi_2, v \vdash_\pi e_1 \Rightarrow \rho, \mu', \texttt{nil}, v_1 \\ v_1 \not\leqslant v_2\end{array}}{\rho, \mu, \texttt{nil}, v \vdash_\pi \texttt{CMP}(\texttt{C\_LE}, e_1, e_2) \Rightarrow \rho, \mu', \texttt{nil}, 0}(\textsc{Le}^\perp)$$

For optimisation purposes mostly, the comparison operators `C_NE`, `C_GT`, `C_GE` may be introduced by the compiler (not by the parser, however).
They are defined as

$$\frac{\begin{array}{c}\rho, \mu, \chi, v \vdash_\pi e_2 \Rightarrow \rho, \mu_2, \chi_2, v_2 \\ \rho, \mu_2, \chi_2, v_2 \vdash_\pi e_1 \Rightarrow \rho, \mu', \texttt{nil}, v_1 \\ v_1 = v_2\end{array}}{\rho, \mu, \texttt{nil}, v \vdash_\pi \texttt{CMP}(\texttt{C\_NE}, e_1, e_2) \Rightarrow \rho, \mu', \texttt{nil}, 0}(\textsc{Ne}^\perp)$$

$$\frac{\begin{array}{c}\rho, \mu, \chi, v \vdash_\pi e_2 \Rightarrow \rho, \mu_2, \chi_2, v_2 \\ \rho, \mu_2, \chi_2, v_2 \vdash_\pi e_1 \Rightarrow \rho, \mu', \texttt{nil}, v_1 \\ v_1 \leqslant v_2\end{array}}{\rho, \mu, \texttt{nil}, v \vdash_\pi \texttt{CMP}(\texttt{C\_GT}, e_1, e_2) \Rightarrow \rho, \mu', \texttt{nil}, 0}(\textsc{Gt}^\perp)$$

$$\frac{\begin{array}{c}\rho, \mu, \chi, v \vdash_\pi e_2 \Rightarrow \rho, \mu_2, \chi_2, v_2 \\ \rho, \mu_2, \chi_2, v_2 \vdash_\pi e_1 \Rightarrow \rho, \mu', \texttt{nil}, v_1 \\ v_1 < v_2\end{array}}{\rho, \mu, \texttt{nil}, v \vdash_\pi \texttt{CMP}(\texttt{C\_GE}, e_1, e_2) \Rightarrow \rho, \mu', \texttt{nil}, 0}(\textsc{Ge}^\perp)$$

$$\frac{\begin{array}{c}\rho, \mu, \chi, v \vdash_\pi e_2 \Rightarrow \rho, \mu_2, \chi_2, v_2 \\ \rho, \mu_2, \chi_2, v_2 \vdash_\pi e_1 \Rightarrow \rho, \mu', \texttt{nil}, v_1 \\ v_1 \neq v_2\end{array}}{\rho, \mu, \texttt{nil}, v \vdash_\pi \texttt{CMP}(\texttt{C\_NE}, e_1, e_2) \Rightarrow \rho, \mu', \texttt{nil}, 1}(\textsc{Ne}^\top)$$

$$\frac{\begin{array}{c}\rho, \mu, \chi, v \vdash_\pi e_2 \Rightarrow \rho, \mu_2, \chi_2, v_2 \\ \rho, \mu_2, \chi_2, v_2 \vdash_\pi e_1 \Rightarrow \rho, \mu', \texttt{nil}, v_1 \\ v_1 \not\leqslant v_2\end{array}}{\rho, \mu, \texttt{nil}, v \vdash_\pi \texttt{CMP}(\texttt{C\_GT}, e_1, e_2) \Rightarrow \rho, \mu', \texttt{nil}, 1}(\textsc{Gt}^\top)$$

$$\frac{\begin{array}{c}\rho, \mu, \chi, v \vdash_\pi e_2 \Rightarrow \rho, \mu_2, \chi_2, v_2 \\ \rho, \mu_2, \chi_2, v_2 \vdash_\pi e_1 \Rightarrow \rho, \mu', \texttt{nil}, v_1 \\ v_1 \not< v_2\end{array}}{\rho, \mu, \texttt{nil}, v \vdash_\pi \texttt{CMP}(\texttt{C\_GE}, e_1, e_2) \Rightarrow \rho, \mu', \texttt{nil}, 1}(\textsc{Ge}^\top)$$

When one of the operands raises a non-`nil` flag :

$$\frac{\begin{array}{c}\rho, \mu, \chi, v \vdash_\pi e_2 \Rightarrow \rho, \mu_2, \chi_2, v_2 \\ \rho, \mu_2, \chi_2, v_2 \vdash_\pi e_1 \Rightarrow \rho, \mu', \chi', v_1 \\ \chi' \neq \texttt{nil}\end{array}}{\rho, \mu, \chi, v \vdash_\pi \texttt{CMP}(op, e_1, e_2) \Rightarrow \rho, \mu', \chi', v_1}(\textsc{Cmp}^\chi)$$

1.5 Assignments

$$\frac{\begin{array}{c}\rho, \mu, \chi, v \vdash_\pi e \Rightarrow \rho, \mu', \texttt{nil}, v_e \\ x \in \operatorname{dom} \rho \qquad \rho(x) \in \operatorname{dom} \mu'\end{array}}{\rho, \mu, \chi, v \vdash_\pi \texttt{SET\_VAR}(x, e) \Rightarrow \rho, \mu'[\rho(x) \mapsto v_e], \texttt{nil}, v_e}(\textsc{Var}^\leftarrow)$$

$$\frac{\rho, \mu, \chi, v \vdash_\pi e \Rightarrow \rho, \mu', \chi', v_e \qquad \chi' \neq \texttt{nil}}{\rho, \mu, \chi, v \vdash_\pi \texttt{SET\_VAR}(x, e) \Rightarrow \rho, \mu', \chi', v_e}(\textsc{Var}^{\leftarrow\chi})$$

$$\frac{\begin{array}{c}\rho, \mu, \chi, v \vdash_\pi e \Rightarrow \rho, \mu_e, \chi_e, v_e \\ \rho, \mu_e, \chi_e, v_e \vdash_\pi i \Rightarrow \rho, \mu', \texttt{nil}, v_i \\ \rho(x) \in \operatorname{dom} \mu'\end{array}}{\rho, \mu, \chi, v \vdash_\pi \texttt{SET\_ARRAY}(x, i, e) \Rightarrow \rho, \mu'[\rho(x) + v_i \times 8 \mapsto v_e], \texttt{nil}, v_e}(\textsc{Idx}^\leftarrow)$$

$$\frac{\begin{array}{c}\rho, \mu, \chi, v \vdash_\pi e \Rightarrow \rho, \mu_e, \chi_e, v_e \\ \rho, \mu_e, \chi_e, v_e \vdash_\pi i \Rightarrow \rho, \mu', \chi', v_i \qquad \chi' \neq \texttt{nil}\end{array}}{\rho, \mu, \chi, v \vdash_\pi \texttt{SET\_ARRAY}(x, i, e) \Rightarrow \rho, \mu', \chi', v_i}(\textsc{Idx}^{\leftarrow\chi})$$

$$\frac{\begin{array}{c}\rho, \mu, \chi, v \vdash_\pi e \Rightarrow \rho, \mu_e, \chi_e, v_e \\ \rho, \mu_e, \chi_e, v_e \vdash_\pi a \Rightarrow \rho, \mu', \texttt{nil}, v_a\end{array}}{\rho, \mu, \chi, v \vdash_\pi \texttt{SET\_DEREF}(a, e) \Rightarrow \rho, \mu'[v_a \mapsto v_e], \texttt{nil}, v_e}(\textsc{Ptr}^\leftarrow)$$

$$\frac{\begin{array}{c}\rho, \mu, \chi, v \vdash_\pi e \Rightarrow \rho, \mu_e, \chi_e, v_e \\ \rho, \mu_e, \chi_e, v_e \vdash_\pi a \Rightarrow \rho, \mu', \chi', v_a \qquad \chi' \neq \texttt{nil}\end{array}}{\rho, \mu, \chi, v \vdash_\pi \texttt{SET\_DEREF}(a, e) \Rightarrow \rho, \mu', \chi', v_a}(\textsc{Ptr}^{\leftarrow\chi})$$

1.6 Increments

On variables :

$$\frac{x \in \operatorname{dom}\rho \qquad \rho(x) = k \in \operatorname{dom}\mu}{\rho, \mu, \mathtt{nil}, v \vdash_\pi \mathtt{OP1}(\mathtt{M\_POST\_INC}, \mathtt{VAR}\ x) \Rightarrow \rho, \mu[k \mapsto \mu(k) + 1], \mathtt{nil}, \mu(k)}(\textsc{Var}^{\bullet\uparrow})$$

$$\frac{x \in \operatorname{dom}\rho \qquad \rho(x) = k \in \operatorname{dom}\mu}{\rho, \mu, \mathtt{nil}, v \vdash_\pi \mathtt{OP1}(\mathtt{M\_POST\_DEC}, \mathtt{VAR}\ x) \Rightarrow \rho, \mu[k \mapsto \mu(k) - 1], \mathtt{nil}, \mu(k)}(\textsc{Var}^{\bullet\downarrow})$$

$$\frac{x \in \operatorname{dom}\rho \qquad \rho(x) = k \in \operatorname{dom}\mu \qquad \mu(k) + 1 = v_k}{\rho, \mu, \mathtt{nil}, v \vdash_\pi \mathtt{OP1}(\mathtt{M\_PRE\_INC}, \mathtt{VAR}\ x) \Rightarrow \rho, \mu[k \mapsto v_k], \mathtt{nil}, v_k}(\textsc{Var}^{\uparrow\bullet})$$

$$\frac{x \in \operatorname{dom}\rho \qquad \rho(x) = k \in \operatorname{dom}\mu \qquad \mu(k) - 1 = v_k}{\rho, \mu, \mathtt{nil}, v \vdash_\pi \mathtt{OP1}(\mathtt{M\_PRE\_DEC}, \mathtt{VAR}\ x) \Rightarrow \rho, \mu[k \mapsto v_k], \mathtt{nil}, v_k}(\textsc{Var}^{\downarrow\bullet})$$

On arrays :

$$\frac{\rho, \mu, \mathtt{nil}, v \vdash_\pi e \Rightarrow i, \mu' \qquad \rho, \mu' \vdash_\pi a \Rightarrow t, \mu'' \qquad t + i \times 8 = k \qquad k \in \operatorname{dom}\mu''}{\rho, \mu, \mathtt{nil}, v \vdash_\pi \mathtt{OP1}(\mathtt{M\_POST\_INC}, \mathtt{OP2}(\mathtt{S\_INDEX}, a, e)) \Rightarrow \mu''(k), \mu''[k \mapsto \mu''(k) + 1]}(\textsc{Idx}^{\bullet\uparrow})$$

$$\frac{\rho, \mu, \mathtt{nil}, v \vdash_\pi e \Rightarrow i, \mu' \qquad \rho, \mu' \vdash_\pi a \Rightarrow t, \mu'' \qquad t + i \times 8 = k \qquad k \in \operatorname{dom}\mu''}{\rho, \mu, \mathtt{nil}, v \vdash_\pi \mathtt{OP1}(\mathtt{M\_POST\_DEC}, \mathtt{OP2}(\mathtt{S\_INDEX}, a, e)) \Rightarrow \mu''(k), \mu''[k \mapsto \mu''(k) - 1]}(\textsc{Idx}^{\bullet\downarrow})$$

$$\frac{\rho, \mu, \mathtt{nil}, v \vdash_\pi e \Rightarrow i, \mu' \qquad \rho, \mu' \vdash_\pi a \Rightarrow t, \mu'' \qquad t + i \times 8 = k \qquad k \in \operatorname{dom}\mu'' \qquad \mu''(k) + 1 = v}{\rho, \mu, \mathtt{nil}, v \vdash_\pi \mathtt{OP1}(\mathtt{M\_PRE\_INC}, \mathtt{OP2}(\mathtt{S\_INDEX}, a, e)) \Rightarrow v, \mu''[k \mapsto v]}(\textsc{Idx}^{\uparrow\bullet})$$

$$\frac{\rho, \mu, \mathtt{nil}, v \vdash_\pi e \Rightarrow i, \mu' \qquad \rho, \mu' \vdash_\pi a \Rightarrow t, \mu'' \qquad t + i \times 8 = k \qquad k \in \operatorname{dom}\mu'' \qquad \mu''(k) - 1 = v}{\rho, \mu, \mathtt{nil}, v \vdash_\pi \mathtt{OP1}(\mathtt{M\_PRE\_DEC}, \mathtt{OP2}(\mathtt{S\_INDEX}, a, e)) \Rightarrow v, \mu''[k \mapsto v]}(\textsc{Idx}^{\downarrow\bullet})$$

On dereferences :

$$\frac{\rho, \mu, \mathtt{nil}, v \vdash_\pi e \Rightarrow k, \mu'}{\rho, \mu, \mathtt{nil}, v \vdash_\pi \mathtt{OP1}(\mathtt{M\_POST\_INC}, \mathtt{OP1}(\mathtt{M\_DEREF}, e) \Rightarrow \mu'(k), \mu'[k \mapsto \mu'(k) + 1]}(\textsc{Ptr}^{\bullet\uparrow})$$

$$\frac{\rho, \mu, \mathtt{nil}, v \vdash_\pi e \Rightarrow k, \mu'}{\rho, \mu, \mathtt{nil}, v \vdash_\pi \mathtt{OP1}(\mathtt{M\_POST\_DEC}, \mathtt{OP1}(\mathtt{M\_DEREF}, e) \Rightarrow \mu'(k), \mu'[k \mapsto \mu'(k) - 1]}(\textsc{Ptr}^{\bullet\downarrow})$$

$$\frac{\rho, \mu, \mathtt{nil}, v \vdash_\pi e \Rightarrow k, \mu' \qquad \mu'(k) + 1 = v}{\rho, \mu, \mathtt{nil}, v \vdash_\pi \mathtt{OP1}(\mathtt{M\_PRE\_INC}, \mathtt{OP1}(\mathtt{M\_DEREF}, e) \Rightarrow v, \mu'[k \mapsto v]}(\textsc{Ptr}^{\uparrow\bullet})$$

$$\frac{\rho, \mu, \mathtt{nil}, v \vdash_\pi e \Rightarrow k, \mu' \qquad \mu'(k) - 1 = v}{\rho, \mu, \mathtt{nil}, v \vdash_\pi \mathtt{OP1}(\mathtt{M\_PRE\_DEC}, \mathtt{OP1}(\mathtt{M\_DEREF}, e) \Rightarrow v, \mu'[k \mapsto v]}(\textsc{Ptr}^{\downarrow\bullet})$$

1.7 Extended assignments

Let $op \in \mathtt{bin\_op} \setminus \{\mathtt{S\_INDEX}\}$.
On variables :

$$\frac{\begin{array}{c}\rho, \mu, \mathtt{nil}, v \vdash_\pi e \Rightarrow v, \mu' \\ x \in \operatorname{dom}\rho \qquad \rho(x) \in \operatorname{dom}(\mu') \qquad \rho(x) = k \qquad \mu'(k) = u \\ \rho, \mu' \vdash_\pi \mathtt{OP2}(op, \mathtt{CST}\ v, \mathtt{CST}\ u) \Rightarrow w, \mu''\end{array}}{\rho, \mu, \mathtt{nil}, v \vdash_\pi \mathtt{OPSET\_VAR}(op, x, e) \Rightarrow w, \mu''[k \mapsto w]}(\textsc{Var}^{\leftarrow op})$$

On arrays :

$$\frac{\begin{array}{c}\rho, \mu, \mathtt{nil}, v \vdash_\pi e_2 \Rightarrow v, \mu' \\ \rho, \mu' \vdash_\pi e_1 \Rightarrow i, \mu'' \\ t \in \operatorname{dom}\rho \qquad \rho(t) + i \times 8 = k \qquad k \in \operatorname{dom}\mu'' \\ \mu''(k) = u \qquad \rho, \mu'' \vdash_\pi \mathtt{OP2}(op, \mathtt{CST}\ v, \mathtt{CST}\ u) \Rightarrow w\mu'''\end{array}}{\rho, \mu, \mathtt{nil}, v \vdash_\pi \mathtt{OPSET\_ARRAY}(op, t, e_1, e_2) \Rightarrow w, \mu'''[k \mapsto w]}(\textsc{Idx}^{\leftarrow op})$$

On dereferences :

$$\frac{\begin{array}{c}\rho, \mu, \mathtt{nil}, v \vdash_\pi e_2 \Rightarrow v, \mu' \\ \rho, \mu' \vdash_\pi e_1 \Rightarrow k, \mu'' \\ k \in \operatorname{dom}\mu'' \qquad \mu''(k) = u \\ \rho, \mu'' \vdash_\pi \mathtt{OP2}(op, \mathtt{CST}\ v, \mathtt{CST}\ u) \Rightarrow w\mu'''\end{array}}{\rho, \mu, \mathtt{nil}, v \vdash_\pi \mathtt{OPSET\_DEREF}(op, e_1, e_2) \Rightarrow w, \mu'''[k \mapsto w]}(\textsc{Ptr}^{\leftarrow op})$$

1.8 Ternary operator

$$\frac{\begin{array}{c}\rho,\mu,\texttt{nil},v\vdash_\pi e\Rightarrow c,\mu' \qquad c=0\\ \rho,\mu'\vdash_\pi e_\bot\Rightarrow v,\mu''\end{array}}{\rho,\mu,\texttt{nil},v\vdash_\pi \texttt{EIF}(e,e_\top,e_\bot)\Rightarrow v,\mu''}(\textsc{Tern}^\bot)$$

$$\frac{\begin{array}{c}\rho,\mu,\texttt{nil},v\vdash_\pi e\Rightarrow c,\mu' \qquad c\neq 0\\ \rho,\mu'\vdash_\pi e_\top\Rightarrow v,\mu''\end{array}}{\rho,\mu,\texttt{nil},v\vdash_\pi \texttt{EIF}(e,e_\top,e_\bot)\Rightarrow v,\mu''}(\textsc{Tern}^\top)$$

1.9 Sequence

$$\frac{\begin{array}{c}\rho,\mu,\texttt{nil},v\vdash_\pi e_1\Rightarrow v_1,\mu_1\\ \rho,\mu_1\vdash_\pi e_2\Rightarrow v_2,\mu_2\\ \cdots\\ \rho,\mu_{n-1}\vdash_\pi e_n\Rightarrow v_n,\mu'\end{array}}{\rho,\mu,\texttt{nil},v\vdash_\pi \texttt{ESEQ}\ [e_1;\cdots;e_n]\Rightarrow v_n,\mu'}(\textsc{Seq}^n)$$

With a special case for the empty sequence :

$$\frac{}{\rho,\mu,\texttt{nil},v\vdash_\pi \texttt{ESEQ}\ []\Rightarrow x,\mu}(\textsc{Seq}^0)$$

where x is an arbitrary value

1.10 Function call

For a toplevel function :

$$\frac{\begin{array}{c}\rho,\mu,\texttt{nil},v\vdash_\pi e_n\Rightarrow v_n,\mu_n\\ \rho,\mu_n\vdash_\pi e_{n-1}\Rightarrow v_{n-1},\mu_{n-1}\\ \cdots\\ \rho,\mu_2\vdash_\pi e_1\Rightarrow v_1,\mu'\\ f\in\operatorname{dom}\mathrm{fun}^n_\pi\setminus\operatorname{dom}\rho \qquad \mathrm{fun}^n_\pi(f)(\rho,\mu',v_1,\cdots,v_n)=(w,\mu'')\end{array}}{\rho,\mu,\texttt{nil},v\vdash_\pi \texttt{CALL}(f,[e_1;\cdots;e_n])\Rightarrow w,\mu''}(\textsc{Call}^n_\pi)$$

For a function pointer :

$$\frac{\begin{array}{c}\rho,\mu,\texttt{nil},v\vdash_\pi e_n\Rightarrow v_n,\mu_n\\ \rho,\mu_n\vdash_\pi e_{n-1}\Rightarrow v_{n-1},\mu_{n-1}\\ \cdots\\ \rho,\mu_2\vdash_\pi e_1\Rightarrow v_1,\mu'\\ f\in\operatorname{dom}\rho \qquad \rho(f)\in\operatorname{dom}\mathrm{fun}^n_\mu \qquad \mathrm{fun}^n_\mu(\rho(f))(\rho,\mu',v_1,\cdots,v_n)=(w,\mu'')\end{array}}{\rho,\mu,\texttt{nil},v\vdash_\pi \texttt{CALL}(f,[e_1;\cdots;e_n])\Rightarrow w,\mu''}(\textsc{Call}^n_\mu)$$

2 Code

2.1 Expressions

When an expression is executed as a statement.
If only the flag `nil` is raised, the expression is executed

$$\frac{\rho,\mu\vdash_\pi e\Rightarrow v',\mu'}{\rho,\mu,\texttt{nil},v\Vdash_\pi \texttt{CEXPR}\ e\Rightarrow \rho,\mu',\texttt{nil},v'}(\textsc{Expr}^{\texttt{nil}})$$

Otherwise it is skipped.

$$\frac{\chi\neq\texttt{nil}}{\rho,\mu,\chi,v\Vdash_\pi \texttt{CEXPR}\ e\Rightarrow \rho,\mu,\chi,v}(\textsc{Expr}^\chi)$$

2.2 Conditional branching

If only `nil` is raised, one of the two branches is executed depending on how the condition evaluates.

$$\frac{\begin{array}{c}\rho,\mu\vdash_\pi e\Rightarrow v,\mu' \qquad v=0\\ \rho,\mu',\texttt{nil},v\Vdash_\pi c_\bot\Rightarrow \rho',\mu',x,v'\end{array}}{\rho,\mu',\texttt{nil},v\Vdash_\pi \texttt{CIF}(e,c_\top,c_\bot)\Rightarrow \rho,\mu',x,v'}(\textsc{If}^\bot)$$

$$\frac{\begin{array}{c}\rho, \mu \vdash_\pi e \Rightarrow v, \mu' \qquad v \neq 0 \\ \rho, \mu', \texttt{nil}, v \Vdash_\pi c_\top \Rightarrow \rho', \mu', x, v'\end{array}}{\rho, \mu, \texttt{nil}, v \Vdash_\pi \texttt{CIF}(e, c_\top, c_\bot) \Rightarrow \rho, \mu', x, v'}(\textsc{If}^\top)$$

Note that the branch is allowed to modify the memory and raise flags, but not change the environment : ρ is preserved.

For all other flags, the condition and both branches are skipped.

$$\frac{\chi \neq \texttt{nil}}{\rho, \mu, \chi, v \Vdash_\pi \texttt{CIF}(e, c_\top, c_\bot) \Rightarrow \rho, \mu, \chi, v}(\textsc{If}^\chi)$$

2.3 Blocks

$$\frac{\begin{array}{c}\rho, \mu, \texttt{nil}, v \Vdash_\pi c \Rightarrow \rho', \mu', \chi', v' \\ \rho', \mu', \chi', v' \Vdash_\pi \texttt{CBLOCK}\ S \Rightarrow \rho'', \mu'', \chi'', v''\end{array}}{\rho, \mu, \texttt{nil}, v \Vdash_\pi \texttt{CBLOCK}(c :: S) \Rightarrow \rho, \mu'', \chi'', v''}(\textsc{Block}^1)$$

$$\frac{}{\rho, \mu, \texttt{nil}, v \Vdash_\pi \texttt{CBLOCK}\ [] \Rightarrow \rho, \mu, \texttt{nil}, v''}(\textsc{Block}^0)$$

Again for blocks, the memory may be changed and flags may be raised, but the environment is preserved.

For all other flags, the whole block is skipped.

$$\frac{\chi \neq \texttt{nil}}{\rho, \mu, \chi, v \Vdash_\pi \texttt{CBLOCK}\ S \Rightarrow \rho, \mu, \chi, v}(\textsc{Block}^\chi)$$

2.4 Loops

A loop with a false condition stops :

$$\frac{\rho, \mu \vdash_\pi e \Rightarrow a, \mu' \qquad a = 0}{\rho, \mu, \texttt{nil}, v \Vdash_\pi \texttt{CWHILE}(e, c, f, \texttt{true}) \Rightarrow \rho, \mu', \texttt{nil}, v}(\textsc{While}^{\bot, f, \texttt{true}})$$

Except in the case of a `do-while` :

$$\frac{\begin{array}{c}\rho, \mu, \texttt{nil}, v \Vdash_\pi c \Rightarrow \rho', \mu', \texttt{nil}, v' \\ \rho, \mu', \texttt{nil}, v' \Vdash_\pi \texttt{CWHILE}(e, c, \texttt{None}, \texttt{true}) \Rightarrow \rho'', \mu'', \chi'', v''\end{array}}{\rho, \mu, \texttt{nil}, v \Vdash_\pi \texttt{CWHILE}(e, c, \texttt{None}, \texttt{false}) \Rightarrow \rho, \mu'', \chi'', v''}(\textsc{While}^{\top, \texttt{None}, \texttt{false}})$$

$$\frac{\begin{array}{c}\rho, \mu, \texttt{nil}, v \Vdash_\pi c \Rightarrow \rho', \mu', \texttt{nil}, v' \\ \rho, \mu' \vdash_\pi f \Rightarrow a, \mu'' \\ \rho, \mu'', \texttt{nil}, v' \Vdash_\pi \texttt{CWHILE}(e, c, \texttt{Some}\ f, \texttt{true}) \Rightarrow \rho''', \mu''', \chi''', v'''\end{array}}{\rho, \mu, \texttt{nil}, v \Vdash_\pi \texttt{CWHILE}(e, c, \texttt{Some}\ f, \texttt{false}) \Rightarrow \rho, \mu''', \chi''', v'''}(\textsc{While}^{\top, \texttt{Some}, \texttt{false}})$$

A loop continues normally if its condition is nonzero and its body does not raise a flag other than `nil` :

$$\frac{\begin{array}{c}\rho, \mu \vdash_\pi e \Rightarrow a, \mu' \qquad a \neq 0 \\ \rho, \mu', \texttt{nil}, v' \Vdash_\pi c \Rightarrow \rho'', \mu'', \texttt{nil}, v'' \\ \rho, \mu'', \texttt{nil}, v'' \Vdash_\pi \texttt{CWHILE}(e, c, \texttt{None}, \texttt{true}) \Rightarrow \rho''', \mu''', \chi''', v'''\end{array}}{\rho, \mu, \texttt{nil}, v \Vdash_\pi \texttt{CWHILE}(e, c, \texttt{None}, \texttt{true}) \Rightarrow \rho, \mu''', \chi''', v'''}(\textsc{While}^{\top, \texttt{None}, \texttt{true}})$$

$$\frac{\begin{array}{c}\rho, \mu \vdash_\pi e \Rightarrow a, \mu' \qquad a \neq 0 \\ \rho, \mu', \texttt{nil}, v' \Vdash_\pi c \Rightarrow \rho'', \mu'', \texttt{nil}, v'' \\ \rho, \mu'' \vdash_\pi f \Rightarrow \rho''', \mu''' \\ \rho, \mu''', \texttt{nil}, v''' \Vdash_\pi \texttt{CWHILE}(e, c, \texttt{Some}\ f, \texttt{true}) \Rightarrow \rho'''', \mu'''', \chi'''', v''''\end{array}}{\rho, \mu, \texttt{nil}, v \Vdash_\pi \texttt{CWHILE}(e, c, \texttt{Some}\ f, \texttt{true}) \Rightarrow \rho, \mu'''', \chi'''', v''''}(\textsc{While}^{\top, \texttt{Some}, \texttt{true}})$$