Semantics of $\pm$C(i.e. extended C−−)

Villani Neven, ENS Paris-Saclay
Programmation 1 – Compilateur COCass

21 décembre 2020

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

Notation

$\mathbb{Z}_{64}$ is the set of 64-bit signed integers, in which all calculations are done when not specified otherwise.

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, $(\mu : \mathbb{Z}_{64} \to \mathbb{Z}_8) \in \mathcal{M}$ the memory.
μ is read by blocks of 8 bytes : $\mu^{64}(i) \triangleq \sum_{k=0}^{7} 2^{8k} \mu(i + k)$.
$\rho_g \in \mathcal{P}$ is the global environment.

A flag is defined as an element of $\mathcal{E} \triangleq S \sqcup \{\texttt{brk}, \texttt{ret}, \texttt{cnt}, \texttt{nil}\}$: either an exception string or a special control flow keyword.
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 $\text{fun}_\pi^n : \mathbb{Z}_{64} \to \texttt{code}$, a wrapper around $\pm$C functions : $\text{fun}_\pi^n(a)(p_1, \cdots, p_n) = c$ updates the environment with $p_1, \cdots, p_n$ and executes the body of the function whose definition was given by the code c and stored at a. This way of considering functions allows in particular for function pointers.

For $\mu \in \mathcal{M}, v \in \mathbb{Z}_8, x \in \mathbb{Z}_{64}$ we write $\mu[x \mapsto v] : \begin{cases} x \mapsto v \\ y \mapsto \mu(y) & y \in \text{dom}\,\mu \setminus \{x\} \end{cases}$
However we will usually use $\mu^{64}[x \mapsto v] \triangleq \mu[x + k \mapsto v_k \mid 0 \leqslant k < 8, \ v = \sum_{k=0}^{8} 2^{8k} v_k]$, i.e. the memory is written 8 bytes at a time.

A similar notation is used for ρ, ρ_g and fun_π^n.

1 Expressions

1.1 Reading values

For local and global variables :

$$\frac{x \in \text{dom}\,\rho \qquad \rho(x) \in \text{dom}\,\mu}{\rho, \mu, \texttt{nil}, v \vdash_\pi \texttt{VAR } x \Rightarrow \rho, \mu, \texttt{nil}, \mu^{64}(\rho(x))}(\textsc{Var})$$

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 :

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

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

For arrays :

$$\frac{\begin{array}{c} \rho, \mu, \chi, v \vdash_\pi i \Rightarrow \rho, \mu_i, \chi_i, v_i \\ \rho, \mu_i, \chi_i, i \vdash_\pi a \Rightarrow \rho, \mu_a, \texttt{nil}, v_a \\ v_a + v_i \times 8 \in \operatorname{dom} \mu_a \end{array}}{\rho, \mu, \chi, v \vdash_\pi \texttt{OP2(S\_INDEX}, a, i) \Rightarrow \rho, \mu_a, \texttt{nil}, \mu_a^{64}(v_a + v_i \times 8)}(\text{Idx})$$

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

1.2 Unary operators without side-effects

Unary minus (same as C−−) :

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

Unary bitwise negation (same as C−−) :

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

Indirection (added in ±C) :

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

$$\frac{\begin{array}{c} \rho, \mu, \chi, v \vdash_\pi i \Rightarrow \rho, \mu_i, \chi_i, v_i \\ \rho, \mu_i, \chi_i, v_i \vdash_\pi a \Rightarrow \rho, \mu_a, \texttt{nil}, v_a \end{array}}{\rho, \mu, \chi, v \vdash_\pi \texttt{OP1(M\_ADDR, OP2(S\_INDEX}, a, i)) \Rightarrow \rho, \mu_a, \texttt{nil}, v_a + v_i \times 8}(\text{Idx}^\&)$$

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

Dereferencing (added in ±C) :

$$\frac{\rho, \mu, \chi, v \vdash_\pi a \Rightarrow \rho, \mu_a, \texttt{nil}, v_a \qquad v_a \in \operatorname{dom} \mu_a}{\rho, \mu, \chi, v \vdash_\pi \texttt{OP1(M\_DEREF}, a) \Rightarrow \rho, \mu_a, \texttt{nil}, \mu_a^{64}(v_a)}(\text{Ptr})$$

When the operand raises a non-nil flag :

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

1.3 Binary operators

Multiplication (same as C−−) :

$$\frac{\begin{array}{c} \rho, \mu, \chi, v \vdash_\pi e_2 \Rightarrow \rho, \mu_2, \chi_2, v_2 \\ \rho, \mu_2, \chi_2, v_2 \vdash_\pi e_1 \Rightarrow \rho, \mu_1, \texttt{nil}, v_1 \end{array}}{\rho, \mu, \chi, v \vdash_\pi \texttt{OP2(S\_MUL}, e_1, e_2) \Rightarrow \rho, \mu_1, \texttt{nil}, v_1 \times v_2}(\text{Mul})$$

Addition (same as C−−) :

$$\frac{\begin{array}{c} \rho, \mu, \chi, v \vdash_\pi e_2 \Rightarrow \rho, \mu_2, \chi_2, v_2 \\ \rho, \mu_2, \chi_2, v_2 \vdash_\pi e_1 \Rightarrow \rho, \mu_1, \texttt{nil}, v_1 \end{array}}{\rho, \mu, \chi, v \vdash_\pi \texttt{OP2(S\_ADD}, e_1, e_2) \Rightarrow \rho, \mu_1, \texttt{nil}, v_1 + v_2}(\text{Add})$$

Subtraction (same as C−−) :

$$\frac{\begin{array}{c} \rho, \mu, \chi, v \vdash_\pi e_2 \Rightarrow \rho, \mu_2, \chi_2, v_2 \\ \rho, \mu_2, \chi_2, v_2 \vdash_\pi e_1 \Rightarrow \rho, \mu_1, \texttt{nil}, v_1 \end{array}}{\rho, \mu, \chi, v \vdash_\pi \texttt{OP2(S\_SUB}, e_1, e_2) \Rightarrow \rho, \mu_1, \texttt{nil}, v_1 - v_2}(\text{Sub})$$

Division and remainder (same as C−−) :

$$\frac{\rho, \mu, \chi, v \vdash_\pi e_2 \Rightarrow \rho, \mu_2, \chi_2, v_2 \qquad v_2 \neq 0 \\ \rho, \mu_2, \chi_2, v_2 \vdash_\pi e_1 \Rightarrow \rho, \mu_1, \texttt{nil}, v_1}{\rho, \mu, \chi, v \vdash_\pi \texttt{OP2(S\_DIV}, e_1, e_2) \Rightarrow \rho, \mu_1, \texttt{nil}, v_1 \operatorname{div} v_2}(\text{Div})$$

$$\frac{\begin{array}{c}\rho,\mu,\chi,v\vdash_\pi e_2\Rightarrow\rho,\mu_2,\chi_2,v_2\qquad v_2\neq 0\\\rho,\mu_2,\chi_2,v_2\vdash_\pi e_1\Rightarrow\rho,\mu_1,\texttt{nil},v_1\end{array}}{\rho,\mu,\chi,v\vdash_\pi\texttt{OP2(S\_MOD},e_1,e_2)\Rightarrow\rho,\mu_1,\texttt{nil},v_1\bmod v_2}(\textsc{Mod})$$

Shifts (added in $\pm$C) :

$$\frac{\begin{array}{c}\rho,\mu,\chi,v\vdash_\pi e_2\Rightarrow\rho,\mu_2,\chi_2,v_2\\\rho,\mu_2,\chi_2,v_2\vdash_\pi e_1\Rightarrow\rho,\mu_1,\texttt{nil},v_1\end{array}}{\rho,\mu,\chi,v\vdash_\pi\texttt{OP2(S\_SHL},e_1,e_2)\Rightarrow\rho,\mu_1,\texttt{nil},v_1\times 2^{v_2}}(\textsc{Shl})$$

$$\frac{\begin{array}{c}\rho,\mu,\chi,v\vdash_\pi e_2\Rightarrow\rho,\mu_2,\chi_2,v_2\\\rho,\mu_2,\chi_2,v_2\vdash_\pi e_1\Rightarrow\rho,\mu_1,\texttt{nil},v_1\end{array}}{\rho,\mu,\chi,v\vdash_\pi\texttt{OP2(S\_SHR},e_1,e_2)\Rightarrow\rho,\mu_1,\texttt{nil},v_1\text{ div }2^{v_2}}(\textsc{Shr})$$

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 $\pm$C).

$$\frac{\begin{array}{cc}\rho,\mu,\chi,v\vdash_\pi e_2\Rightarrow\rho,\mu_2,\chi_2,v_2 & \rho,\mu_2,\chi_2,v_2\vdash_\pi e_1\Rightarrow\rho,\mu_1,\texttt{nil},v_1\\(b_0^2,\cdots,b_{63}^2)=\text{bin}_{64}(v_2) & (b_0^1,\cdots,b_{63}^1)=\text{bin}_{64}(v_1)\end{array}}{\rho,\mu,\chi,v\vdash_\pi\texttt{OP2(S\_AND},e_1,e_2)\Rightarrow\rho,\mu_1,\texttt{nil},\text{dec}_{64}(b_0^1\wedge b_0^2,\cdots,b_{63}^1\wedge b_{63}^2),\mu_1}(\textsc{And})$$

$$\frac{\begin{array}{cc}\rho,\mu,\chi,v\vdash_\pi e_2\Rightarrow\rho,\mu_2,\chi_2,v_2 & \rho,\mu_2,\chi_2,v_2\vdash_\pi e_1\Rightarrow\rho,\mu_1,\texttt{nil},v_1\\(b_0^2,\cdots,b_{63}^2)=\text{bin}_{64}(v_2) & (b_0^1,\cdots,b_{63}^1)=\text{bin}_{64}(v_1)\end{array}}{\rho,\mu,\chi,v\vdash_\pi\texttt{OP2(S\_OR},e_1,e_2)\Rightarrow\rho,\mu_1,\texttt{nil},\text{dec}_{64}(b_0^1\vee b_0^2,\cdots,b_{63}^1\vee b_{63}^2),\mu_1}(\textsc{Ior})$$

$$\frac{\begin{array}{cc}\rho,\mu,\chi,v\vdash_\pi e_2\Rightarrow\rho,\mu_2,\chi_2,v_2 & \rho,\mu_2,\chi_2,v_2\vdash_\pi e_1\Rightarrow\rho,\mu_1,\texttt{nil},v_1\\(b_0^2,\cdots,b_{63}^2)=\text{bin}_{64}(v_2) & (b_0^1,\cdots,b_{63}^1)=\text{bin}_{64}(v_1)\end{array}}{\rho,\mu,\chi,v\vdash_\pi\texttt{OP2(S\_XOR},e_1,e_2)\Rightarrow\rho,\mu_1,\texttt{nil},\text{dec}_{64}(b_0^1\oplus b_0^2,\cdots,b_{63}^1\oplus b_{63}^2),\mu_1}(\textsc{Xor})$$

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

$$\frac{\begin{array}{c}\rho,\mu,\chi,v\vdash_\pi e_2\Rightarrow\rho,\mu_2,\chi_2,v_2\\\rho,\mu_2,\chi_2,v_2\vdash_\pi e_1\Rightarrow\rho,\mu_1,\chi_1,v_1\\\chi_1\neq\texttt{nil}\end{array}}{\rho,\mu,\chi,v\vdash_\pi\texttt{OP2}(op,e_1,e_2)\Rightarrow\rho,\mu_1,\chi_1,v_1}(\textsc{Op2}^\chi)$$

1.4 Comparisons

All are the same as in C−−.

$$\frac{\begin{array}{c}\rho,\mu,\chi,v\vdash_\pi e_2\Rightarrow\rho,\mu_2,\chi_2,v_2\\\rho,\mu_2,\chi_2,v\vdash_\pi e_1\Rightarrow\rho,\mu_1,\texttt{nil},v_1\\v_1=v_2\end{array}}{\rho,\mu,\texttt{nil},v\vdash_\pi\texttt{CMP(C\_EQ},e_1,e_2)\Rightarrow\rho,\mu_1,\texttt{nil},1}(\textsc{Eq}^\top)$$

$$\frac{\begin{array}{c}\rho,\mu,\chi,v\vdash_\pi e_2\Rightarrow\rho,\mu_2,\chi_2,v_2\\\rho,\mu_2,\chi_2,v\vdash_\pi e_1\Rightarrow\rho,\mu_1,\texttt{nil},v_1\\v_1<v_2\end{array}}{\rho,\mu,\texttt{nil},v\vdash_\pi\texttt{CMP(C\_LT},e_1,e_2)\Rightarrow\rho,\mu_1,\texttt{nil},1}(\textsc{Lt}^\top)$$

$$\frac{\begin{array}{c}\rho,\mu,\chi,v\vdash_\pi e_2\Rightarrow\rho,\mu_2,\chi_2,v_2\\\rho,\mu_2,\chi_2,v\vdash_\pi e_1\Rightarrow\rho,\mu_1,\texttt{nil},v_1\\v_1\leqslant v_2\end{array}}{\rho,\mu,\texttt{nil},v\vdash_\pi\texttt{CMP(C\_LE},e_1,e_2)\Rightarrow\rho,\mu_1,\texttt{nil},1}(\textsc{Le}^\top)$$

$$\frac{\begin{array}{c}\rho,\mu,\chi,v\vdash_\pi e_2\Rightarrow\rho,\mu_2,\chi_2,v_2\\\rho,\mu_2,\chi_2,v\vdash_\pi e_1\Rightarrow\rho,\mu_1,\texttt{nil},v_1\\v_1\neq v_2\end{array}}{\rho,\mu,\texttt{nil},v\vdash_\pi\texttt{CMP(C\_EQ},e_1,e_2)\Rightarrow\rho,\mu_1,\texttt{nil},0}(\textsc{Eq}^\bot)$$

$$\frac{\begin{array}{c}\rho,\mu,\chi,v\vdash_\pi e_2\Rightarrow\rho,\mu_2,\chi_2,v_2\\\rho,\mu_2,\chi_2,v\vdash_\pi e_1\Rightarrow\rho,\mu_1,\texttt{nil},v_1\\v_1\not< v_2\end{array}}{\rho,\mu,\texttt{nil},v\vdash_\pi\texttt{CMP(C\_LT},e_1,e_2)\Rightarrow\rho,\mu_1,\texttt{nil},0}(\textsc{Lt}^\bot)$$

$$\frac{\begin{array}{c}\rho,\mu,\chi,v \vdash_\pi e_2 \Rightarrow \rho,\mu_2,\chi_2,v_2 \\ \rho,\mu_2,\chi_2,v \vdash_\pi e_1 \Rightarrow \rho,\mu_1,\mathtt{nil},v_1 \\ v_1 \not\leqslant v_2\end{array}}{\rho,\mu,\mathtt{nil},v \vdash_\pi \mathtt{CMP(C\_LE},e_1,e_2) \Rightarrow \rho,\mu_1,\mathtt{nil},0}(\mathrm{LE}^\perp)$$

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

$$\frac{\begin{array}{c}\rho,\mu,\chi,v \vdash_\pi e_2 \Rightarrow \rho,\mu_2,\chi_2,v_2 \\ \rho,\mu_2,\chi_2,v_2 \vdash_\pi e_1 \Rightarrow \rho,\mu_1,\mathtt{nil},v_1 \\ v_1 = v_2\end{array}}{\rho,\mu,\mathtt{nil},v \vdash_\pi \mathtt{CMP(C\_NE},e_1,e_2) \Rightarrow \rho,\mu_1,\mathtt{nil},0}(\mathrm{NE}^\perp)$$

$$\frac{\begin{array}{c}\rho,\mu,\chi,v \vdash_\pi e_2 \Rightarrow \rho,\mu_2,\chi_2,v_2 \\ \rho,\mu_2,\chi_2,v_2 \vdash_\pi e_1 \Rightarrow \rho,\mu_1,\mathtt{nil},v_1 \\ v_1 \leqslant v_2\end{array}}{\rho,\mu,\mathtt{nil},v \vdash_\pi \mathtt{CMP(C\_GT},e_1,e_2) \Rightarrow \rho,\mu_1,\mathtt{nil},0}(\mathrm{GT}^\perp)$$

$$\frac{\begin{array}{c}\rho,\mu,\chi,v \vdash_\pi e_2 \Rightarrow \rho,\mu_2,\chi_2,v_2 \\ \rho,\mu_2,\chi_2,v_2 \vdash_\pi e_1 \Rightarrow \rho,\mu_1,\mathtt{nil},v_1 \\ v_1 < v_2\end{array}}{\rho,\mu,\mathtt{nil},v \vdash_\pi \mathtt{CMP(C\_GE},e_1,e_2) \Rightarrow \rho,\mu_1,\mathtt{nil},0}(\mathrm{GE}^\perp)$$

$$\frac{\begin{array}{c}\rho,\mu,\chi,v \vdash_\pi e_2 \Rightarrow \rho,\mu_2,\chi_2,v_2 \\ \rho,\mu_2,\chi_2,v_2 \vdash_\pi e_1 \Rightarrow \rho,\mu_1,\mathtt{nil},v_1 \\ v_1 \neq v_2\end{array}}{\rho,\mu,\mathtt{nil},v \vdash_\pi \mathtt{CMP(C\_NE},e_1,e_2) \Rightarrow \rho,\mu_1,\mathtt{nil},1}(\mathrm{NE}^\top)$$

$$\frac{\begin{array}{c}\rho,\mu,\chi,v \vdash_\pi e_2 \Rightarrow \rho,\mu_2,\chi_2,v_2 \\ \rho,\mu_2,\chi_2,v_2 \vdash_\pi e_1 \Rightarrow \rho,\mu_1,\mathtt{nil},v_1 \\ v_1 \not\leqslant v_2\end{array}}{\rho,\mu,\mathtt{nil},v \vdash_\pi \mathtt{CMP(C\_GT},e_1,e_2) \Rightarrow \rho,\mu_1,\mathtt{nil},1}(\mathrm{GT}^\top)$$

$$\frac{\begin{array}{c}\rho,\mu,\chi,v \vdash_\pi e_2 \Rightarrow \rho,\mu_2,\chi_2,v_2 \\ \rho,\mu_2,\chi_2,v_2 \vdash_\pi e_1 \Rightarrow \rho,\mu_1,\mathtt{nil},v_1 \\ v_1 \not< v_2\end{array}}{\rho,\mu,\mathtt{nil},v \vdash_\pi \mathtt{CMP(C\_GE},e_1,e_2) \Rightarrow \rho,\mu_1,\mathtt{nil},1}(\mathrm{GE}^\top)$$

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

$$\frac{\begin{array}{c}\rho,\mu,\chi,v \vdash_\pi e_2 \Rightarrow \rho,\mu_2,\chi_2,v_2 \\ \rho,\mu_2,\chi_2,v_2 \vdash_\pi e_1 \Rightarrow \rho,\mu_1,\chi_1,v_1 \\ \chi_1 \neq \mathtt{nil}\end{array}}{\rho,\mu,\chi,v \vdash_\pi \mathtt{CMP}(op,e_1,e_2) \Rightarrow \rho,\mu_1,\chi_1,v_1}(\mathrm{CMP}^\chi)$$

1.5 Assignments

$$\frac{\begin{array}{c}\rho,\mu,\chi,v \vdash_\pi e \Rightarrow \rho,\mu_e,\mathtt{nil},v_e \\ x \in \mathrm{dom}\,\rho \qquad \rho(x) \in \mathrm{dom}\,\mu_e\end{array}}{\rho,\mu,\chi,v \vdash_\pi \mathtt{SET\_VAR}(x,e) \Rightarrow \rho,\mu_e^{64}[\rho(x) \mapsto v_e],\mathtt{nil},v_e}(\mathrm{VAR}^\leftarrow)$$

$$\frac{\rho,\mu,\chi,v \vdash_\pi e \Rightarrow \rho,\mu_e,\chi_e,v_e \qquad \chi_e \neq \mathtt{nil}}{\rho,\mu,\chi,v \vdash_\pi \mathtt{SET\_VAR}(x,e) \Rightarrow \rho,\mu_e,\chi_e,v_e}(\mathrm{VAR}^{\leftarrow\chi})$$

$$\frac{\begin{array}{c}\rho,\mu,\chi,v \vdash_\pi e \Rightarrow \rho,\mu_e,\chi_e,v_e \\ \rho,\mu_e,\chi_e,v_e \vdash_\pi i \Rightarrow \rho,\mu_i,\mathtt{nil},v_i \\ \rho(x) \in \mathrm{dom}\,\mu_i\end{array}}{\rho,\mu,\chi,v \vdash_\pi \mathtt{SET\_ARRAY}(x,i,e) \Rightarrow \rho,\mu_i^{64}[\rho(x) + v_i \times 8 \mapsto v_e],\mathtt{nil},v_e}(\mathrm{IDX}^\leftarrow)$$

$$\frac{\begin{array}{c}\rho,\mu,\chi,v \vdash_\pi e \Rightarrow \rho,\mu_e,\chi_e,v_e \\ \rho,\mu_e,\chi_e,v_e \vdash_\pi i \Rightarrow \rho,\mu_i,\chi_i,v_i \qquad \chi_i \neq \mathtt{nil}\end{array}}{\rho,\mu,\chi,v \vdash_\pi \mathtt{SET\_ARRAY}(x,i,e) \Rightarrow \rho,\mu_i,\chi_i,v_i}(\mathrm{IDX}^{\leftarrow\chi})$$

$$\frac{\begin{array}{c}\rho,\mu,\chi,v \vdash_\pi e \Rightarrow \rho,\mu_e,\chi_e,v_e \\ \rho,\mu_e,\chi_e,v_e \vdash_\pi a \Rightarrow \rho,\mu_a,\mathtt{nil},v_a\end{array}}{\rho,\mu,\chi,v \vdash_\pi \mathtt{SET\_DEREF}(a,e) \Rightarrow \rho,\mu_a^{64}[v_a \mapsto v_e],\mathtt{nil},v_e}(\mathrm{PTR}^\leftarrow)$$

$$\frac{\begin{array}{c}\rho,\mu,\chi,v \vdash_\pi e \Rightarrow \rho,\mu_e,\chi_e,v_e \\ \rho,\mu_e,\chi_e,v_e \vdash_\pi a \Rightarrow \rho,\mu_a,\chi_a,v_a \qquad \chi_a \neq \mathtt{nil}\end{array}}{\rho,\mu,\chi,v \vdash_\pi \mathtt{SET\_DEREF}(a,e) \Rightarrow \rho,\mu_a,\chi_a,v_a}(\mathrm{PTR}^{\leftarrow\chi})$$

1.6 Increments

On variables :

$$\frac{x \in \mathrm{dom}\,\rho \qquad \rho(x) = k \in \mathrm{dom}\,\mu \qquad \mu^{64}(k) = v_k}{\rho, \mu, \mathtt{nil}, v \vdash_\pi \mathtt{OP1(M\_POST\_INC, VAR}\ x) \Rightarrow \rho, \mu^{64}[k \mapsto v_k + 1], \mathtt{nil}, v_k}(\mathrm{VAR}^{\bullet\uparrow})$$

$$\frac{x \in \mathrm{dom}\,\rho \qquad \rho(x) = k \in \mathrm{dom}\,\mu \qquad \mu^{64}(k) = v_k}{\rho, \mu, \mathtt{nil}, v \vdash_\pi \mathtt{OP1(M\_POST\_DEC, VAR}\ x) \Rightarrow \rho, \mu^{64}[k \mapsto v_k - 1], \mathtt{nil}, v_k}(\mathrm{VAR}^{\bullet\downarrow})$$

$$\frac{x \in \mathrm{dom}\,\rho \qquad \rho(x) = k \in \mathrm{dom}\,\mu \qquad \mu^{64}(k) + 1 = v_k}{\rho, \mu, \mathtt{nil}, v \vdash_\pi \mathtt{OP1(M\_PRE\_INC, VAR}\ x) \Rightarrow \rho, \mu^{64}[k \mapsto v_k], \mathtt{nil}, v_k}(\mathrm{VAR}^{\uparrow\bullet})$$

$$\frac{x \in \mathrm{dom}\,\rho \qquad \rho(x) = k \in \mathrm{dom}\,\mu \qquad \mu^{64}(k) - 1 = v_k}{\rho, \mu, \mathtt{nil}, v \vdash_\pi \mathtt{OP1(M\_PRE\_DEC, VAR}\ x) \Rightarrow \rho, \mu^{64}[k \mapsto v_k], \mathtt{nil}, v_k}(\mathrm{VAR}^{\downarrow\bullet})$$

On arrays :

$$\frac{\begin{array}{c} \rho, \mu, \chi, v \vdash_\pi i \Rightarrow \rho, \mu_i, \chi_i, v_i \\ \rho, \mu_i, \chi_i, v_i \vdash_\pi a \Rightarrow \rho, \mu_a, \mathtt{nil}, v_a \\ v_a + v_i \times 8 = k \qquad k \in \mathrm{dom}\,\mu_a \end{array}}{\rho, \mu, \chi, v \vdash_\pi \mathtt{OP1(M\_POST\_INC, OP2(S\_INDEX},a,e)) \Rightarrow \rho, \mu_a^{64}[k \mapsto \mu_a(k) + 1], \mathtt{nil}, \mu_a^{64}(k)}(\mathrm{IDX}^{\bullet\uparrow})$$

$$\frac{\begin{array}{c} \rho, \mu, \chi, v \vdash_\pi i \Rightarrow \rho, \mu_i, \chi_i, v_i \\ \rho, \mu_i, \chi_i, v_i \vdash_\pi a \Rightarrow \rho, \mu_a, \mathtt{nil}, v_a \\ v_a + v_i \times 8 = k \qquad k \in \mathrm{dom}\,\mu_a \end{array}}{\rho, \mu, \chi, v \vdash_\pi \mathtt{OP1(M\_POST\_DEC, OP2(S\_INDEX},a,e)) \Rightarrow \rho, \mu_a^{64}[k \mapsto \mu_a(k) - 1], \mathtt{nil}, \mu_a^{64}(k)}(\mathrm{IDX}^{\bullet\downarrow})$$

$$\frac{\begin{array}{c} \rho, \mu, \chi, v \vdash_\pi i \Rightarrow \rho, \mu_i, \chi_i, v_i \\ \rho, \mu_i, \chi_i, v_i \vdash_\pi a \Rightarrow \rho, \mu_a, \mathtt{nil}, v_a \\ v_a + v_i \times 8 = k \qquad k \in \mathrm{dom}\,\mu_a \qquad \mu_a^{64}(k) + 1 = v_k \end{array}}{\rho, \mu, \chi, v \vdash_\pi \mathtt{OP1(M\_PRE\_INC, OP2(S\_INDEX},a,e)) \Rightarrow \rho, \mu_a[k \mapsto v_k], \mathtt{nil}, v_k}(\mathrm{IDX}^{\uparrow\bullet})$$

$$\frac{\begin{array}{c} \rho, \mu, \chi, v \vdash_\pi i \Rightarrow \rho, \mu_i, \chi_i, v_i \\ \rho, \mu_i, \chi_i, v_i \vdash_\pi a \Rightarrow \rho, \mu_a, \mathtt{nil}, v_a \\ v_a + v_i \times 8 = k \qquad k \in \mathrm{dom}\,\mu_a \qquad \mu_a^{64}(k) - 1 = v_k \end{array}}{\rho, \mu, \chi, v \vdash_\pi \mathtt{OP1(M\_PRE\_DEC, OP2(S\_INDEX},a,e)) \Rightarrow \rho, \mu_a[k \mapsto v_k], \mathtt{nil}, v_k}(\mathrm{IDX}^{\downarrow\bullet})$$

On dereferences :

$$\frac{\rho, \mu, \chi, v \vdash_\pi e \Rightarrow \rho, \mu_e, \mathtt{nil}, v_e}{\rho, \mu, \chi, v \vdash_\pi \mathtt{OP1(M\_POST\_INC, OP1(M\_DEREF},e)) \Rightarrow \rho, \mu_e^{64}[v_e \mapsto \mu_e(v_e) + 1], \mathtt{nil}, \mu_e(v_e)}(\mathrm{PTR}^{\bullet\uparrow})$$

$$\frac{\rho, \mu, \chi, v \vdash_\pi e \Rightarrow \rho, \mu_e, \mathtt{nil}, v_e}{\rho, \mu, \chi, v \vdash_\pi \mathtt{OP1(M\_POST\_DEC, OP1(M\_DEREF},e)) \Rightarrow \rho, \mu_e^{64}[v_e \mapsto \mu_e(v_e) - 1], \mathtt{nil}, \mu_e^{64}(v_e)}(\mathrm{PTR}^{\bullet\downarrow})$$

$$\frac{\rho, \mu, \chi, v \vdash_\pi e \Rightarrow \rho, \mu_e, \mathtt{nil}, v_e \qquad \mu_e^{64}(v_e) + 1 = k}{\rho, \mu, \chi, v \vdash_\pi \mathtt{OP1(M\_PRE\_INC, OP1(M\_DEREF},e)) \Rightarrow \rho, \mu_e[v_e \mapsto k], \mathtt{nil}, k}(\mathrm{PTR}^{\uparrow\bullet})$$

$$\frac{\rho, \mu, \chi, v \vdash_\pi e \Rightarrow \rho, \mu_e, \mathtt{nil}, v_e \qquad \mu_e^{64}(v_e) - 1 = k}{\rho, \mu, \chi, v \vdash_\pi \mathtt{OP1(M\_PRE\_DEC, OP1(M\_DEREF},e)) \Rightarrow \rho, \mu_e[v_e \mapsto k], \mathtt{nil}, k}(\mathrm{PTR}^{\downarrow\bullet})$$

1.7 Extended assignments

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

$$\frac{\begin{array}{c} \rho, \mu, \chi, v \vdash_\pi e \Rightarrow \rho, \mu_e, \chi_e, v_e \\ x \in \mathrm{dom}\,\rho \qquad \rho(x) \in \mathrm{dom}(\mu_e) \qquad \rho(x) = k \qquad \mu_e^{64}(k) = u \\ \rho, \mu_e, \chi_e, v_e \vdash_\pi \mathtt{OP2}(op, \mathtt{CST}\ v, \mathtt{CST}\ u) \Rightarrow \rho, \mu', \mathtt{nil}, w \end{array}}{\rho, \mu, \chi, v \vdash_\pi \mathtt{OPSET\_VAR}(op, x, e) \Rightarrow \rho, \mu'[k \mapsto w], \mathtt{nil}, w}(\mathrm{VAR}^{\leftarrow op})$$

$$\frac{\rho, \mu, \chi, v \vdash_\pi e \Rightarrow \rho, \mu_e, \chi_e, v_e \qquad \chi_e \neq \mathtt{nil}}{\rho, \mu, \chi, v \vdash_\pi \mathtt{OPSET\_VAR}(op, x, e) \Rightarrow \rho, \mu_e, \chi_e, v_e}(\mathrm{VAR}^{\leftarrow op\chi})$$

On arrays :

$$\frac{\begin{array}{c} \rho, \mu, \chi, v \vdash_\pi e \Rightarrow \rho, \mu_e, \chi_e, v_e \\ \rho, \mu_e, \chi_e, v_e \vdash_\pi i \Rightarrow \rho, \mu_i, \chi_i, v_i \\ t \in \mathrm{dom}\,\rho \qquad \rho(t) + v_i \times 8 = k \qquad k \in \mathrm{dom}\,\mu_i \\ \mu_i^{64}(k) = u \qquad \rho, \mu_i, \chi_i, v_i \vdash_\pi \mathtt{OP2}(op, \mathtt{CST}\ v, \mathtt{CST}\ u) \Rightarrow \rho, \mu', , \mathtt{nil}, w \end{array}}{\rho, \mu, \chi, v \vdash_\pi \mathtt{OPSET\_ARRAY}(op, t, e_1, e_2) \Rightarrow \rho, \mu'[k \mapsto w], \mathtt{nil}, w}(\mathrm{IDX}^{\leftarrow op})$$

$$\frac{\begin{array}{c}\rho,\mu,\chi,v\vdash_\pi e\Rightarrow\rho,\mu_e,\chi_e,v_e\\\rho,\mu_e,\chi_e,v_e\vdash_\pi i\Rightarrow\rho,\mu_i,\chi_i,v_i\qquad\chi_i\neq\mathtt{nil}\end{array}}{\rho,\mu,\chi,v\vdash_\pi\mathtt{OPSET\_ARRAY}(op,t,e_1,e_2)\Rightarrow\rho,\mu_i,\chi_i,v_i}(\text{IDX}^{\leftarrow op\chi})$$

On dereferences :

$$\frac{\begin{array}{c}\rho,\mu,\chi,v\vdash_\pi a\Rightarrow\rho,\mu_a,\chi_a,v_a\\\rho,\mu_a,\chi_a,v_a\vdash_\pi e\Rightarrow\rho,\mu_e,\chi_e,v_e\\v_e\in\operatorname{dom}\mu_e\qquad\mu_e^{64}(v_e)=u\\\rho,\mu_e,\chi_e,v_e\vdash_\pi\mathtt{OP2}(op,\mathtt{CST}\ v,\mathtt{CST}\ u)\Rightarrow\rho,\mu',\mathtt{nil},w\end{array}}{\rho,\mu,\chi,v\vdash_\pi\mathtt{OPSET\_DEREF}(op,e_1,e_2)\Rightarrow\rho,\mu'[k\mapsto w],\mathtt{nil},w}(\text{PTR}^{\leftarrow op})$$

$$\frac{\begin{array}{c}\rho,\mu,\chi,v\vdash_\pi a\Rightarrow\rho,\mu_a,\chi_a,v_a\\\rho,\mu_a,\chi_a,v_a\vdash_\pi e\Rightarrow\rho,\mu_e,\chi_e,v_e\qquad\chi_e\neq\mathtt{nil}\end{array}}{\rho,\mu,\chi,v\vdash_\pi\mathtt{OPSET\_DEREF}(op,e_1,e_2)\Rightarrow\rho,\mu_e,\chi_e,v_e}(\text{PTR}^{\leftarrow op\chi})$$

1.8   Ternary operator

$$\frac{\begin{array}{c}\rho,\mu,\chi,v\vdash_\pi e\Rightarrow\rho,\mu_e,\mathtt{nil},v_e\qquad v_e=0\\\rho,\mu_e,\mathtt{nil},v_e\vdash_\pi e_\perp\Rightarrow\rho,\mu_\perp,\chi_\perp,v_\perp\end{array}}{\rho,\mu,\chi,v\vdash_\pi\mathtt{EIF}(e,e_\top,e_\perp)\Rightarrow\rho,\mu_\perp,\chi_\perp,v_\perp}(\text{TERN}^\perp)$$

$$\frac{\begin{array}{c}\rho,\mu,\chi,v\vdash_\pi e\Rightarrow\rho,\mu_e,\mathtt{nil},v_e\qquad v_e\neq0\\\rho,\mu_e,\mathtt{nil},v_e\vdash_\pi e_\top\Rightarrow\rho,\mu_\top,\chi_\top,v_\top\end{array}}{\rho,\mu,\chi,v\vdash_\pi\mathtt{EIF}(e,e_\top,e_\perp)\Rightarrow\rho,\mu_\top,\chi_\top,v_\top}(\text{TERN}^\top)$$

$$\frac{\rho,\mu,\chi,v\vdash_\pi e\Rightarrow\rho,\mu_e,\chi_e,v_e\qquad\chi_e\neq\mathtt{nil}}{\rho,\mu,\chi,v\vdash_\pi\mathtt{EIF}(e,e_\top,e_\perp)\Rightarrow\rho,\mu_e,\chi_e,v_e}(\text{TERN}^\chi)$$

1.9   Sequence

$$\frac{\begin{array}{c}\rho,\mu_0,\chi_0,v_0\vdash_\pi e_1\Rightarrow\rho,\mu_1,\chi_1,v_1\\\cdots\\\rho,\mu_{n-1},\chi_{n-1},v_{n-1}\vdash_\pi e_n\Rightarrow\rho,\mu_n,\chi_n,v_n\end{array}}{\rho,\mu_0,\chi_0,v_0\vdash_\pi\mathtt{ESEQ}\ [e_1;\cdots;e_n]\Rightarrow\rho,\mu_n,\chi_n,v_n}(\text{SEQ}^n)$$

1.10   Function call

Works for both a toplevel function and a function pointer :

$$\frac{\begin{array}{c}\rho,\mu_{n+1},\chi_{n+1},v_{n+1}\vdash_\pi e_n\Rightarrow\rho,\mu_n,\chi_n,v_n\\\cdots\\\rho,\mu_2,\chi_2,v_2\vdash_\pi e_1\Rightarrow\rho,\mu_1,\mathtt{nil},v_1\\f\in\operatorname{dom}\operatorname{fun}_\pi^n\\\rho_g,\mu_1,\mathtt{nil},0\vdash_\pi\operatorname{fun}_\pi^n(f)(v_1,\cdots,v_n)\Rightarrow\rho_f,\mu_f,\chi_f,v_f\end{array}}{\rho,\mu_n,\chi_n,v_n\vdash_\pi\mathtt{CALL}(f,[e_1;\cdots;e_n])\Rightarrow\rho,\mu_f,\chi_f,v_f}(\text{CALL}^n)$$

$$\frac{\begin{array}{c}\rho,\mu_{n+1},\chi_{n+1},v_{n+1}\vdash_\pi e_n\Rightarrow\rho,\mu_n,\chi_n,v_n\\\cdots\\\rho,\mu_2,\chi_2,v_2\vdash_\pi e_1\Rightarrow\rho,\mu_1,\chi_1,v_1\\\chi_1\neq\mathtt{nil}\end{array}}{\rho,\mu_{n+1},\chi_{n+1},v_{n+1}\vdash_\pi\mathtt{CALL}(f,[e_1;\cdots;e_n])\Rightarrow\rho,\mu_1,\chi_1,v_1}(\text{CALL}^\chi)$$

2   Code

2.1   Expressions

An expression as statement is simply executed. If a non-$\mathtt{nil}$ flag is raised, it will be skipped anyway.

$$\frac{\rho,\mu,\chi,v\vdash_\pi e\Rightarrow\rho,\mu_e,\chi_e,v_e}{\rho,\mu,\chi,v\Vdash_\pi\mathtt{CEXPR}\ e\Rightarrow\rho,\mu_e,\chi_e,v_e}(\text{EXPR})$$

2.2 Conditional branching

If only `nil` is raised after the evaluation of the condition, one of the two branches is executed.

$$\frac{\rho, \mu, \chi, v \vdash_\pi e \Rightarrow \rho, \mu_e, \texttt{nil}, v_e \qquad v_e = 0 \qquad \rho, \mu_e, \texttt{nil}, v_e \vdash_\pi c_\bot \Rightarrow \rho_\bot, \mu_\bot, \chi_\bot, v_\bot}{\rho, \mu, \chi, v \vdash_\pi \texttt{CIF}(e, c_\top, c_\bot) \Rightarrow \rho, \mu_\bot, \chi_\bot, v_\bot} (\text{IF}^\bot)$$

$$\frac{\rho, \mu, \chi, v \vdash_\pi e \Rightarrow \rho, \mu_e, \texttt{nil}, v_e \qquad v_e \neq 0 \qquad \rho, \mu_e, \texttt{nil}, v_e \vdash_\pi c_\top \Rightarrow \rho_\top, \mu_\top, \chi_\top, v_\top}{\rho, \mu, \chi, v \vdash_\pi \texttt{CIF}(e, c_\top, c_\bot) \Rightarrow \rho, \mu_\top, \chi_\top, v_\top} (\text{IF}^\top)$$

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, neither of the branches is executed.

$$\frac{\rho, \mu, \chi, v \vdash_\pi e \Rightarrow \rho, \mu_e, \chi_e, v_e \qquad \chi_e \neq \texttt{nil}}{\rho, \mu, \chi, v \vdash_\pi \texttt{CIF}(e, c_\top, c_\bot) \Rightarrow \rho, \mu_e, \chi_e, v_e} (\text{IF}^\chi)$$

2.3 Blocks

$$\frac{\rho, \mu, \chi, v \vdash_\pi c \Rightarrow \rho', \mu', \chi', v' \qquad \rho', \mu', \chi', v' \vdash_\pi \texttt{CBLOCK } S \Rightarrow \rho'', \mu'', \chi'', v''}{\rho, \mu, \chi, v \vdash_\pi \texttt{CBLOCK}(c :: S) \Rightarrow \rho, \mu'', \chi'', v''} (\text{BLOCK}^1)$$

$$\frac{}{\rho, \mu, \chi, v \vdash_\pi \texttt{CBLOCK } [] \Rightarrow \rho, \mu, \chi, v} (\text{BLOCK}^0)$$

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

2.4 Loops

$$\frac{}{\rho, \mu, \chi, v \vdash_\pi \texttt{None} \Rightarrow \rho, \mu, \chi, v} (\text{NONE})$$

$$\frac{\rho, \mu, \chi, v \vdash_\pi f \Rightarrow \rho_f, \mu_f, \chi_f, v_f}{\rho, \mu, \chi, v \vdash_\pi \texttt{Some } f \Rightarrow \rho_f, \mu_f, \chi_f, v_f} (\text{SOME})$$

A loop with a false condition stops :

$$\frac{\rho, \mu, \chi, v \vdash_\pi e \Rightarrow \rho, \mu_e, \texttt{nil}, v_e \qquad v_e = 0}{\rho, \mu, \chi, v \vdash_\pi \texttt{CWHILE}(e, c, f, \texttt{true}) \Rightarrow \rho, \mu', \texttt{nil}, v} (\text{WHILE}^{\bot, \texttt{true}})$$

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

$$\frac{\rho, \mu, \chi, v \vdash_\pi c \Rightarrow \rho_c, \mu_c, \chi_c, v_c \qquad \rho, \mu_c, \chi_c, v_c \vdash_\pi f \Rightarrow \rho, \mu_f, \texttt{nil}, v_f \qquad \rho, \mu_f, \texttt{nil}, v_f \vdash_\pi \texttt{CWHILE}(e, c, f, \texttt{true}) \Rightarrow \rho, \mu_w, \chi_w, v_w}{\rho, \mu, \chi, v \vdash_\pi \texttt{CWHILE}(e, c, f, \texttt{false}) \Rightarrow \rho, \mu_w, \chi_w, v_w} (\text{WHILE}^{\texttt{false}})$$

A loop continues normally if its condition is nonzero :

$$\frac{\rho, \mu, \chi, v \vdash_\pi e \Rightarrow \rho, \mu_e, \texttt{nil}, v_e \quad v_e \neq 0 \qquad \rho, \mu_e, \texttt{nil}, v_e \vdash_\pi c \Rightarrow \rho_c, \mu_c, \chi_c, v_c \quad \chi_c \notin \{\texttt{brk}, \texttt{cnt}\} \qquad \rho, \mu_c, \chi_c, v_c \vdash_\pi f \Rightarrow \rho, \mu_f, \chi_f, v_f \qquad \rho, \mu_f, \chi_f, v_f \vdash_\pi \texttt{CWHILE}(e, c, f, \texttt{true}) \Rightarrow \rho, \mu_w, \chi_w, v_w}{\rho, \mu, \chi, v \vdash_\pi \texttt{CWHILE}(e, c, f, \texttt{true}) \Rightarrow \rho, \mu_w, \chi_w, v_w} (\text{WHILE}^{\top, \texttt{true}})$$

A flag skips the loop :

$$\frac{\rho, \mu, \chi, v \vdash_\pi e \Rightarrow \rho, \mu_e, \chi_e, v_e \qquad \chi_e \neq \texttt{nil}}{\rho, \mu, \chi, v \vdash_\pi \texttt{CWHILE}(e, c, f, \texttt{true}) \Rightarrow \rho, \mu_e, \chi_e, v_e} (\text{WHILE}^{\chi, \texttt{true}})$$

$$\frac{\chi \neq \texttt{nil}}{\rho, \mu, \chi, v \vdash_\pi \texttt{CWHILE}(e, c, f, \texttt{false}) \Rightarrow \rho, \mu, \chi, v} (\text{WHILE}^{\chi, \texttt{false}})$$

`cnt` executes the finally clause before continuing as normal :

$$\frac{\begin{array}{c}\rho, \mu, \chi, v \vdash_\pi e \Rightarrow \rho, \mu_e, \texttt{nil}, v_e \qquad v_e \neq 0 \\ \rho, \mu_e, \texttt{nil}, v_e \vdash_\pi c \Rightarrow \rho_c, \mu_c, \texttt{cnt}, v_c \\ \rho, \mu_c, \texttt{nil}, v_c \vdash_\pi f \Rightarrow \rho, \mu_f, \chi_f, v_f \\ \rho, \mu_f, \chi_f, v_f \vdash_\pi \texttt{CWHILE}(e, c, f, \texttt{true}) \Rightarrow \rho, \mu_w, \chi_w, v_w\end{array}}{\rho, \mu, \chi, v \vdash_\pi \texttt{CWHILE}(e, c, f, \texttt{true}) \Rightarrow \rho, \mu_w, \chi_w, v_w}(\textsc{While}^{\texttt{cnt,true}})$$

$$\frac{\begin{array}{c}\rho, \mu, \chi, v \vdash_\pi c \Rightarrow \rho_c, \mu_c, \texttt{cnt}, v_c \\ \rho, \mu_c, \texttt{nil}, v_c \vdash_\pi f \Rightarrow \rho, \mu_f, \chi_f, v_f \\ \rho, \mu_f, \chi_f, v_f \vdash_\pi \texttt{CWHILE}(e, c, f, \texttt{true}) \Rightarrow \rho, \mu_w, \chi_w, v_w\end{array}}{\rho, \mu, \chi, v \vdash_\pi \texttt{CWHILE}(e, c, f, \texttt{false}) \Rightarrow \rho, \mu_w, \chi_w, v_w}(\textsc{While}^{\texttt{cnt,false}})$$

`brk` interrupts the loop but is not retransmitted :

$$\frac{\begin{array}{c}\rho, \mu, \chi, v \vdash_\pi e \Rightarrow \rho, \mu_e, \texttt{nil}, v_e \qquad v_e \neq 0 \\ \rho, \mu_e, \texttt{nil}, v_e \vdash_\pi c \Rightarrow \rho_c, \mu_c, \texttt{brk}, v_c\end{array}}{\rho, \mu, \chi, v \vdash_\pi \texttt{CWHILE}(e, c, f, \texttt{true}) \Rightarrow \rho, \mu_c, \texttt{nil}, v_c}(\textsc{While}^{\texttt{brk,true}})$$

$$\frac{\rho, \mu, \chi, v \vdash_\pi c \Rightarrow \rho_c, \mu_c, \texttt{brk}, v_c}{\rho, \mu, \chi, v \vdash_\pi \texttt{CWHILE}(e, c, f, \texttt{false}) \Rightarrow \rho, \mu_c, \texttt{nil}, v_c}(\textsc{While}^{\texttt{brk,false}})$$

2.5   Control flow

$$\frac{}{\rho, \mu, \texttt{nil}, v \vdash_\pi \texttt{CBREAK} \Rightarrow \rho, \mu, \texttt{brk}, 0}(\textsc{Break})$$

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

$$\frac{}{\rho, \mu, \texttt{nil}, v \vdash_\pi \texttt{CCONTINUE} \Rightarrow \rho, \mu, \texttt{cnt}, 0}(\textsc{Continue})$$

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

$$\frac{}{\rho, \mu, \texttt{nil}, v \vdash_\pi \texttt{CRETURN None} \Rightarrow \rho, \mu, \texttt{ret}, 0}(\textsc{Return}^{\texttt{None}})$$

$$\frac{\rho, \mu, \chi, v \vdash_\pi e \Rightarrow \rho, \mu_e, \texttt{nil}, v_e}{\rho, \mu, \texttt{nil}, v \vdash_\pi \texttt{CRETURN}(\texttt{Some } e) \Rightarrow \rho, \mu_e, \texttt{ret}, v_e}(\textsc{Return}^{\texttt{Some}})$$

$$\frac{\chi \neq \texttt{nil}}{\rho, \mu, \chi, v \vdash_\pi \texttt{CRETURN None} \Rightarrow \rho, \mu, \chi, v}(\textsc{Return}^{\texttt{None}\chi})$$

$$\frac{\rho, \mu, \chi, v \vdash_\pi e \Rightarrow \rho, \mu_e, \chi_e, v_e \qquad \chi_e \neq \texttt{nil}}{\rho, \mu, \texttt{nil}, v \vdash_\pi \texttt{CRETURN}(\texttt{Some } e) \Rightarrow \rho, \mu_e, \chi_e, v_e}(\textsc{Return}^{\texttt{Some}\chi})$$

2.6   Local variable declarations

First, the obvious :

$$\frac{}{\rho, \mu, \chi, v \vdash_\pi \texttt{CLOCAL } [] \Rightarrow \rho, \mu, \chi, v}(\textsc{Local}^0)$$

$$\frac{\chi \neq \texttt{nil}}{\rho, \mu, \chi, v \vdash_\pi \texttt{CLOCAL } d \Rightarrow \rho, \mu, \chi, v}(\textsc{Local}^\chi)$$

There are never functions defined in a `CLOCAL`, only `CDECL` :

$$\frac{\begin{array}{c}\rho, \mu, \chi, v \vdash_\pi e \Rightarrow \rho, \mu_e, \texttt{nil}, v_e \\ k \in alloc^8(\mu_e) \qquad \rho' = \rho[w \mapsto k] \qquad \mu' = \mu_e[k \mapsto v_e] \\ \rho', \mu', \texttt{nil}, v_e \vdash_\pi \texttt{CLOCAL } S \Rightarrow \rho_s, \mu_s, \chi_s, v_s\end{array}}{\rho, \mu, \chi, v \vdash_\pi \texttt{CLOCAL}(\texttt{CDECL}(w, e) :: S) \Rightarrow \rho_s, \mu_s, \chi_s, v_s}(\textsc{Local}^1)$$

2.7 Throw

If a flag is already raised, skip the `CTHROW` :

$$\frac{\rho, \mu, \chi, v \vdash_\pi e \Rightarrow \rho, \mu_e, \chi_e, v_e \qquad \chi_e \neq \texttt{nil}}{\rho, \mu, \chi, v \vdash_\pi \texttt{CTHROW}(s, e) \Rightarrow \rho, \mu_e, \chi_e, v_e} (\text{Throw}^\chi)$$

Otherwise raise the new exception $s \in S$:

$$\frac{\rho, \mu, \chi, v \vdash_\pi e \Rightarrow \rho, \mu_e, \texttt{nil}, v_e}{\rho, \mu, \chi, v \vdash_\pi \texttt{CTHROW}(s, e) \Rightarrow \rho, \mu_e, s, v_e} (\text{Throw})$$