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

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

22 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 :

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

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

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

For constant integers :

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

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

For strings :

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

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

For arrays :

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

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

1.2 Unary operators without side-effects

Unary minus (same as C−−) :

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

Unary bitwise negation (same as C−−) :

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

Indirection (added in ±C) :

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

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

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

Dereferencing (added in ±C) :

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

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

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

1.3 Binary operators

Multiplication (same as C−−) :

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

Addition (same as C−−) :

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

Subtraction (same as C−−) :

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

Division and remainder (same as C−−) :

$$\frac{\begin{array}{c}\rho, \mu, \chi, v \vdash^e_\pi e_2 \Rightarrow \rho, \mu_2, \chi_2, v_2 \qquad v_2 \neq 0 \\ \rho, \mu_2, \chi_2, v_2 \vdash^e_\pi e_1 \Rightarrow \rho, \mu_1, \texttt{nil}, v_1\end{array}}{\rho, \mu, \chi, v \vdash^e_\pi \texttt{OP2}(\texttt{S\_DIV}, e_1, e_2) \Rightarrow \rho, \mu_1, \texttt{nil}, v_1\ \mathrm{div}\ v_2}(\textsc{Div})$$

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

Shifts (added in ±C) :

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

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

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 ±C).

$$\frac{\begin{array}{c}\rho, \mu, \chi, v \vdash^e_\pi e_2 \Rightarrow \rho, \mu_2, \chi_2, v_2 \qquad \rho, \mu_2, \chi_2, v_2 \vdash^e_\pi e_1 \Rightarrow \rho, \mu_1, \texttt{nil}, v_1 \\ (b^2_0, \cdots, b^2_{63}) = \text{bin}_{64}(v_2) \qquad (b^1_0, \cdots, b^1_{63}) = \text{bin}_{64}(v_1)\end{array}}{\rho, \mu, \chi, v \vdash^e_\pi \texttt{OP2(S\_AND}, e_1, e_2\texttt{)} \Rightarrow \rho, \mu_1, \texttt{nil}, \text{dec}_{64}(b^1_0 \wedge b^2_0, \cdots, b^1_{63} \wedge b^2_{63}), \mu_1}(\textsc{And})$$

$$\frac{\begin{array}{c}\rho, \mu, \chi, v \vdash^e_\pi e_2 \Rightarrow \rho, \mu_2, \chi_2, v_2 \qquad \rho, \mu_2, \chi_2, v_2 \vdash^e_\pi e_1 \Rightarrow \rho, \mu_1, \texttt{nil}, v_1 \\ (b^2_0, \cdots, b^2_{63}) = \text{bin}_{64}(v_2) \qquad (b^1_0, \cdots, b^1_{63}) = \text{bin}_{64}(v_1)\end{array}}{\rho, \mu, \chi, v \vdash^e_\pi \texttt{OP2(S\_OR}, e_1, e_2\texttt{)} \Rightarrow \rho, \mu_1, \texttt{nil}, \text{dec}_{64}(b^1_0 \vee b^2_0, \cdots, b^1_{63} \vee b^2_{63}), \mu_1}(\textsc{Ior})$$

$$\frac{\begin{array}{c}\rho, \mu, \chi, v \vdash^e_\pi e_2 \Rightarrow \rho, \mu_2, \chi_2, v_2 \qquad \rho, \mu_2, \chi_2, v_2 \vdash^e_\pi e_1 \Rightarrow \rho, \mu_1, \texttt{nil}, v_1 \\ (b^2_0, \cdots, b^2_{63}) = \text{bin}_{64}(v_2) \qquad (b^1_0, \cdots, b^1_{63}) = \text{bin}_{64}(v_1)\end{array}}{\rho, \mu, \chi, v \vdash^e_\pi \texttt{OP2(S\_XOR}, e_1, e_2\texttt{)} \Rightarrow \rho, \mu_1, \texttt{nil}, \text{dec}_{64}(b^1_0 \oplus b^2_0, \cdots, b^1_{63} \oplus b^2_{63}), \mu_1}(\textsc{Xor})$$

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

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

1.4 Comparisons

All are the same as in C−−.

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

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

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

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

$$\frac{\begin{array}{c}\rho, \mu, \chi, v \vdash^e_\pi e_2 \Rightarrow \rho, \mu_2, \chi_2, v_2 \\ \rho, \mu_2, \chi_2, v \vdash^e_\pi e_1 \Rightarrow \rho, \mu_1, \texttt{nil}, v_1 \\ v_1 \nless v_2\end{array}}{\rho, \mu, \texttt{nil}, v \vdash^e_\pi \texttt{CMP(C\_LT}, e_1, e_2\texttt{)} \Rightarrow \rho, \mu_1, \texttt{nil}, 0}(\textsc{Lt}^\bot)$$

$$\frac{\begin{array}{c}\rho, \mu, \chi, v \vdash^e_\pi e_2 \Rightarrow \rho, \mu_2, \chi_2, v_2 \\ \rho, \mu_2, \chi_2, v \vdash^e_\pi e_1 \Rightarrow \rho, \mu_1, \texttt{nil}, v_1 \\ v_1 \not\leqslant v_2\end{array}}{\rho, \mu, \texttt{nil}, v \vdash^e_\pi \texttt{CMP}(\texttt{C\_LE}, e_1, e_2) \Rightarrow \rho, \mu_1, \texttt{nil}, 0}(\textsc{Le}^\perp)$$

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

$$\frac{\begin{array}{c}\rho, \mu, \chi, v \vdash^e_\pi e_2 \Rightarrow \rho, \mu_2, \chi_2, v_2 \\ \rho, \mu_2, \chi_2, v_2 \vdash^e_\pi e_1 \Rightarrow \rho, \mu_1, \texttt{nil}, v_1 \\ v_1 = v_2\end{array}}{\rho, \mu, \texttt{nil}, v \vdash^e_\pi \texttt{CMP}(\texttt{C\_NE}, e_1, e_2) \Rightarrow \rho, \mu_1, \texttt{nil}, 0}(\textsc{Ne}^\perp)$$

$$\frac{\begin{array}{c}\rho, \mu, \chi, v \vdash^e_\pi e_2 \Rightarrow \rho, \mu_2, \chi_2, v_2 \\ \rho, \mu_2, \chi_2, v_2 \vdash^e_\pi e_1 \Rightarrow \rho, \mu_1, \texttt{nil}, v_1 \\ v_1 \leqslant v_2\end{array}}{\rho, \mu, \texttt{nil}, v \vdash^e_\pi \texttt{CMP}(\texttt{C\_GT}, e_1, e_2) \Rightarrow \rho, \mu_1, \texttt{nil}, 0}(\textsc{Gt}^\perp)$$

$$\frac{\begin{array}{c}\rho, \mu, \chi, v \vdash^e_\pi e_2 \Rightarrow \rho, \mu_2, \chi_2, v_2 \\ \rho, \mu_2, \chi_2, v_2 \vdash^e_\pi e_1 \Rightarrow \rho, \mu_1, \texttt{nil}, v_1 \\ v_1 < v_2\end{array}}{\rho, \mu, \texttt{nil}, v \vdash^e_\pi \texttt{CMP}(\texttt{C\_GE}, e_1, e_2) \Rightarrow \rho, \mu_1, \texttt{nil}, 0}(\textsc{Ge}^\perp)$$

$$\frac{\begin{array}{c}\rho, \mu, \chi, v \vdash^e_\pi e_2 \Rightarrow \rho, \mu_2, \chi_2, v_2 \\ \rho, \mu_2, \chi_2, v_2 \vdash^e_\pi e_1 \Rightarrow \rho, \mu_1, \texttt{nil}, v_1 \\ v_1 \neq v_2\end{array}}{\rho, \mu, \texttt{nil}, v \vdash^e_\pi \texttt{CMP}(\texttt{C\_NE}, e_1, e_2) \Rightarrow \rho, \mu_1, \texttt{nil}, 1}(\textsc{Ne}^\top)$$

$$\frac{\begin{array}{c}\rho, \mu, \chi, v \vdash^e_\pi e_2 \Rightarrow \rho, \mu_2, \chi_2, v_2 \\ \rho, \mu_2, \chi_2, v_2 \vdash^e_\pi e_1 \Rightarrow \rho, \mu_1, \texttt{nil}, v_1 \\ v_1 \not\leqslant v_2\end{array}}{\rho, \mu, \texttt{nil}, v \vdash^e_\pi \texttt{CMP}(\texttt{C\_GT}, e_1, e_2) \Rightarrow \rho, \mu_1, \texttt{nil}, 1}(\textsc{Gt}^\top)$$

$$\frac{\begin{array}{c}\rho, \mu, \chi, v \vdash^e_\pi e_2 \Rightarrow \rho, \mu_2, \chi_2, v_2 \\ \rho, \mu_2, \chi_2, v_2 \vdash^e_\pi e_1 \Rightarrow \rho, \mu_1, \texttt{nil}, v_1 \\ v_1 \not< v_2\end{array}}{\rho, \mu, \texttt{nil}, v \vdash^e_\pi \texttt{CMP}(\texttt{C\_GE}, e_1, e_2) \Rightarrow \rho, \mu_1, \texttt{nil}, 1}(\textsc{Ge}^\top)$$

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

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

1.5 Assignments

$$\frac{\begin{array}{c}\rho, \mu, \chi, v \vdash^e_\pi e \Rightarrow \rho, \mu_e, \texttt{nil}, v_e \\ x \in \operatorname{dom}\rho \qquad \rho(x) \in \operatorname{dom}\mu_e\end{array}}{\rho, \mu, \chi, v \vdash^e_\pi \texttt{SET\_VAR}(x, e) \Rightarrow \rho, \mu_e^{64}[\rho(x) \mapsto v_e], \texttt{nil}, v_e}(\textsc{Var}^\leftarrow)$$

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

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

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

$$\frac{\begin{array}{c}\rho, \mu, \chi, v \vdash^e_\pi e \Rightarrow \rho, \mu_e, \chi_e, v_e \\ \rho, \mu_e, \chi_e, v_e \vdash^e_\pi a \Rightarrow \rho, \mu_a, \texttt{nil}, v_a\end{array}}{\rho, \mu, \chi, v \vdash^e_\pi \texttt{SET\_DEREF}(a, e) \Rightarrow \rho, \mu_a^{64}[v_a \mapsto v_e], \texttt{nil}, v_e}(\textsc{Ptr}^\leftarrow)$$

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

1.6 Increments

On variables :

$$\frac{x \in \operatorname{dom}\rho \qquad \rho(x) = k \in \operatorname{dom}\mu \qquad \mu^{64}(k) = v_k}{\rho, \mu, \texttt{nil}, v \vdash^e_\pi \texttt{OP1(M\_POST\_INC, VAR } x\texttt{)} \Rightarrow \rho, \mu^{64}[k \mapsto v_k + 1], \texttt{nil}, v_k}\,(\textsc{Var}^{\bullet\uparrow})$$

$$\frac{x \in \operatorname{dom}\rho \qquad \rho(x) = k \in \operatorname{dom}\mu \qquad \mu^{64}(k) = v_k}{\rho, \mu, \texttt{nil}, v \vdash^e_\pi \texttt{OP1(M\_POST\_DEC, VAR } x\texttt{)} \Rightarrow \rho, \mu^{64}[k \mapsto v_k - 1], \texttt{nil}, v_k}\,(\textsc{Var}^{\bullet\downarrow})$$

$$\frac{x \in \operatorname{dom}\rho \qquad \rho(x) = k \in \operatorname{dom}\mu \qquad \mu^{64}(k) + 1 = v_k}{\rho, \mu, \texttt{nil}, v \vdash^e_\pi \texttt{OP1(M\_PRE\_INC, VAR } x\texttt{)} \Rightarrow \rho, \mu^{64}[k \mapsto v_k], \texttt{nil}, v_k}\,(\textsc{Var}^{\uparrow\bullet})$$

$$\frac{x \in \operatorname{dom}\rho \qquad \rho(x) = k \in \operatorname{dom}\mu \qquad \mu^{64}(k) - 1 = v_k}{\rho, \mu, \texttt{nil}, v \vdash^e_\pi \texttt{OP1(M\_PRE\_DEC, VAR } x\texttt{)} \Rightarrow \rho, \mu^{64}[k \mapsto v_k], \texttt{nil}, v_k}\,(\textsc{Var}^{\downarrow\bullet})$$

On arrays :

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

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

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

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

On dereferences :

$$\frac{\rho, \mu, \chi, v \vdash^e_\pi e \Rightarrow \rho, \mu_e, \texttt{nil}, v_e}{\rho, \mu, \chi, v \vdash^e_\pi \texttt{OP1(M\_POST\_INC, OP1(M\_DEREF, } e\texttt{))} \Rightarrow \rho, \mu_e^{64}[v_e \mapsto \mu_e(v_e) + 1], \texttt{nil}, \mu_e(v_e)}\,(\textsc{Ptr}^{\bullet\uparrow})$$

$$\frac{\rho, \mu, \chi, v \vdash^e_\pi e \Rightarrow \rho, \mu_e, \texttt{nil}, v_e}{\rho, \mu, \chi, v \vdash^e_\pi \texttt{OP1(M\_POST\_DEC, OP1(M\_DEREF, } e\texttt{))} \Rightarrow \rho, \mu_e^{64}[v_e \mapsto \mu_e(v_e) - 1], \texttt{nil}, \mu_e^{64}(v_e)}\,(\textsc{Ptr}^{\bullet\downarrow})$$

$$\frac{\rho, \mu, \chi, v \vdash^e_\pi e \Rightarrow \rho, \mu_e, \texttt{nil}, v_e \qquad \mu_e^{64}(v_e) + 1 = k}{\rho, \mu, \chi, v \vdash^e_\pi \texttt{OP1(M\_PRE\_INC, OP1(M\_DEREF, } e\texttt{))} \Rightarrow \rho, \mu_e[v_e \mapsto k], \texttt{nil}, k}\,(\textsc{Ptr}^{\uparrow\bullet})$$

$$\frac{\rho, \mu, \chi, v \vdash^e_\pi e \Rightarrow \rho, \mu_e, \texttt{nil}, v_e \qquad \mu_e^{64}(v_e) - 1 = k}{\rho, \mu, \chi, v \vdash^e_\pi \texttt{OP1(M\_PRE\_DEC, OP1(M\_DEREF, } e\texttt{))} \Rightarrow \rho, \mu_e[v_e \mapsto k], \texttt{nil}, k}\,(\textsc{Ptr}^{\downarrow\bullet})$$

1.7 Extended assignments

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

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

$$\frac{\rho, \mu, \chi, v \vdash^e_\pi e \Rightarrow \rho, \mu_e, \chi_e, v_e \qquad \chi_e \neq \texttt{nil}}{\rho, \mu, \chi, v \vdash^e_\pi \texttt{OPSET\_VAR(}op, x, e\texttt{)} \Rightarrow \rho, \mu_e, \chi_e, v_e}\,(\textsc{Var}^{\leftarrow op\chi})$$

On arrays :

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

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

On dereferences :

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

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

1.8 Ternary operator

$$\frac{\begin{array}{c}\rho, \mu, \chi, v \vdash^e_\pi e \Rightarrow \rho, \mu_e, \texttt{nil}, v_e \qquad v_e = 0 \\ \rho, \mu_e, \texttt{nil}, v_e \vdash^e_\pi e_\bot \Rightarrow \rho, \mu_\bot, \chi_\bot, v_\bot\end{array}}{\rho, \mu, \chi, v \vdash^e_\pi \texttt{EIF}(e, e_\top, e_\bot) \Rightarrow \rho, \mu_\bot, \chi_\bot, v_\bot}(\textsc{Tern}^\bot)$$

$$\frac{\begin{array}{c}\rho, \mu, \chi, v \vdash^e_\pi e \Rightarrow \rho, \mu_e, \texttt{nil}, v_e \qquad v_e \neq 0 \\ \rho, \mu_e, \texttt{nil}, v_e \vdash^e_\pi e_\top \Rightarrow \rho, \mu_\top, \chi_\top, v_\top\end{array}}{\rho, \mu, \chi, v \vdash^e_\pi \texttt{EIF}(e, e_\top, e_\bot) \Rightarrow \rho, \mu_\top, \chi_\top, v_\top}(\textsc{Tern}^\top)$$

$$\frac{\rho, \mu, \chi, v \vdash^e_\pi e \Rightarrow \rho, \mu_e, \chi_e, v_e \qquad \chi_e \neq \texttt{nil}}{\rho, \mu, \chi, v \vdash^e_\pi \texttt{EIF}(e, e_\top, e_\bot) \Rightarrow \rho, \mu_e, \chi_e, v_e}(\textsc{Tern}^\chi)$$

1.9 Sequence

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

1.10 Function call

Works for both a toplevel function and a function pointer :

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

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

2 Code

2.1 Expressions

An expression as statement is simply executed. If a non-`nil` flag is raised, it will be skipped anyway.

$$\frac{\rho, \mu, \chi, v \vdash^e_\pi e \Rightarrow \rho, \mu_e, \chi_e, v_e}{\rho, \mu, \chi, v \vdash^c_\pi \texttt{CEXPR}\ e \Rightarrow \rho, \mu_e, \chi_e, v_e}(\textsc{Expr})$$

2.2 Conditional branching

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

$$\frac{\rho,\mu,\chi,\upsilon \vdash^e_\pi e \Rightarrow \rho,\mu_e,\texttt{nil},\upsilon_e \qquad \upsilon_e = 0 \quad \rho,\mu_e,\texttt{nil},\upsilon_e \vdash^c_\pi c_\bot \Rightarrow \rho_\bot,\mu_\bot,\chi_\bot,\upsilon_\bot}{\rho,\mu,\chi,\upsilon \vdash^c_\pi \texttt{CIF}(e,c_\top,c_\bot) \Rightarrow \rho,\mu_\bot,\chi_\bot,\upsilon_\bot}(\textsc{If}^\bot)$$

$$\frac{\rho,\mu,\chi,\upsilon \vdash^e_\pi e \Rightarrow \rho,\mu_e,\texttt{nil},\upsilon_e \qquad \upsilon_e \neq 0 \quad \rho,\mu_e,\texttt{nil},\upsilon_e \vdash^c_\pi c_\top \Rightarrow \rho_\top,\mu_\top,\chi_\top,\upsilon_\top}{\rho,\mu,\chi,\upsilon \vdash^c_\pi \texttt{CIF}(e,c_\top,c_\bot) \Rightarrow \rho,\mu_\top,\chi_\top,\upsilon_\top}(\textsc{If}^\top)$$

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, neither of the branches is executed.

$$\frac{\rho,\mu,\chi,\upsilon \vdash^e_\pi e \Rightarrow \rho,\mu_e,\chi_e,\upsilon_e \qquad \chi_e \neq \texttt{nil}}{\rho,\mu,\chi,\upsilon \vdash^c_\pi \texttt{CIF}(e,c_\top,c_\bot) \Rightarrow \rho,\mu_e,\chi_e,\upsilon_e}(\textsc{If}^\chi)$$

2.3 Blocks

$$\frac{\rho,\mu,\chi,\upsilon \vdash^c_\pi c \Rightarrow \rho',\mu',\chi',\upsilon' \quad \rho',\mu',\chi',\upsilon' \vdash^c_\pi \texttt{CBLOCK}\ S \Rightarrow \rho'',\mu'',\chi'',\upsilon''}{\rho,\mu,\chi,\upsilon \vdash^c_\pi \texttt{CBLOCK}(c :: S) \Rightarrow \rho,\mu'',\chi'',\upsilon''}(\textsc{Block}^1)$$

$$\frac{}{\rho,\mu,\chi,\upsilon \vdash^c_\pi \texttt{CBLOCK}\ [] \Rightarrow \rho,\mu,\chi,\upsilon}(\textsc{Block}^0)$$

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

2.4 Loops

$$\frac{}{\rho,\mu,\chi,\upsilon \vdash^{e?}_\pi \texttt{None} \Rightarrow \rho,\mu,\chi,\upsilon}(\textsc{None})$$

$$\frac{\rho,\mu,\chi,\upsilon \vdash^e_\pi f \Rightarrow \rho_f,\mu_f,\chi_f,\upsilon_f}{\rho,\mu,\chi,\upsilon \vdash^{e?}_\pi \texttt{Some}\ f \Rightarrow \rho_f,\mu_f,\chi_f,\upsilon_f}(\textsc{Some})$$

A loop with a false condition stops :

$$\frac{\rho,\mu,\chi,\upsilon \vdash^e_\pi e \Rightarrow \rho,\mu_e,\texttt{nil},\upsilon_e \qquad \upsilon_e = 0}{\rho,\mu,\chi,\upsilon \vdash^c_\pi \texttt{CWHILE}(e,c,f,\texttt{true}) \Rightarrow \rho,\mu',\texttt{nil},\upsilon}(\textsc{While}^{\bot,\texttt{true}})$$

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

$$\frac{\rho,\mu,\chi,\upsilon \vdash^c_\pi c \Rightarrow \rho_c,\mu_c,\chi_c,\upsilon_c \quad \rho,\mu_c,\chi_c,\upsilon_c \vdash^{e?}_\pi f \Rightarrow \rho,\mu_f,\texttt{nil},\upsilon_f \quad \rho,\mu_f,\texttt{nil},\upsilon_f \vdash^c_\pi \texttt{CWHILE}(e,c,f,\texttt{true}) \Rightarrow \rho,\mu_w,\chi_w,\upsilon_w}{\rho,\mu,\chi,\upsilon \vdash^c_\pi \texttt{CWHILE}(e,c,f,\texttt{false}) \Rightarrow \rho,\mu_w,\chi_w,\upsilon_w}(\textsc{While}^{\texttt{false}})$$

A loop continues normally if its condition is nonzero :

$$\frac{\rho,\mu,\chi,\upsilon \vdash^e_\pi e \Rightarrow \rho,\mu_e,\texttt{nil},\upsilon_e \qquad \upsilon_e \neq 0 \quad \rho,\mu_e,\texttt{nil},\upsilon_e \vdash^c_\pi c \Rightarrow \rho_c,\mu_c,\chi_c,\upsilon_c \qquad \chi_c \notin \{\texttt{brk},\texttt{cnt}\} \quad \rho,\mu_c,\chi_c,\upsilon_c \vdash^{e?}_\pi f \Rightarrow \rho,\mu_f,\chi_f,\upsilon_f \quad \rho,\mu_f,\chi_f,\upsilon_f \vdash^c_\pi \texttt{CWHILE}(e,c,f,\texttt{true}) \Rightarrow \rho,\mu_w,\chi_w,\upsilon_w}{\rho,\mu,\chi,\upsilon \vdash^c_\pi \texttt{CWHILE}(e,c,f,\texttt{true}) \Rightarrow \rho,\mu_w,\chi_w,\upsilon_w}(\textsc{While}^{\top,\texttt{true}})$$

A flag skips the loop :

$$\frac{\rho,\mu,\chi,\upsilon \vdash^e_\pi e \Rightarrow \rho,\mu_e,\chi_e,\upsilon_e \qquad \chi_e \neq \texttt{nil}}{\rho,\mu,\chi,\upsilon \vdash^c_\pi \texttt{CWHILE}(e,c,f,\texttt{true}) \Rightarrow \rho,\mu_e,\chi_e,\upsilon_e}(\textsc{While}^{\chi,\texttt{true}})$$

$$\frac{\chi \neq \texttt{nil}}{\rho,\mu,\chi,\upsilon \vdash^c_\pi \texttt{CWHILE}(e,c,f,\texttt{false}) \Rightarrow \rho,\mu,\chi,\upsilon}(\textsc{While}^{\chi,\texttt{false}})$$

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

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

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

`brk` interrupts the loop but is not retransmitted :

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

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

2.5 Control flow

$$\frac{}{\rho, \mu, \mathtt{nil}, v \vdash^c_\pi \mathtt{CBREAK} \Rightarrow \rho, \mu, \mathtt{brk}, 0}(\textsc{Break})$$

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

$$\frac{}{\rho, \mu, \mathtt{nil}, v \vdash^c_\pi \mathtt{CCONTINUE} \Rightarrow \rho, \mu, \mathtt{cnt}, 0}(\textsc{Continue})$$

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

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

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

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

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

2.6 Local variable declarations

First, the obvious :

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

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

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

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

2.7 Throw

If a flag is already raised, skip the CTHROW :

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

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

$$\frac{\rho,\mu,\chi,v \vdash^{e}_{\pi} e \Rightarrow \rho,\mu_e,\texttt{nil},v_e}{\rho,\mu,\chi,v \vdash^{c}_{\pi} \texttt{CTHROW}(s,e) \Rightarrow \rho,\mu_e,s,v_e}(\textsc{Throw})$$

2.8 Switch

$$\frac{\begin{array}{c}\rho,\mu,\chi,v \vdash^{e}_{\pi} e \Rightarrow \rho,\mu_e,\chi_e,v_e \\ \rho,\mu_e,\chi_e,v_e \vdash^{c}_{\pi} \texttt{CBLOCK}(L(v_e)) \Rightarrow \rho,\mu_l,\chi_l,v_l\end{array}}{\rho,\mu,\chi,v \vdash^{c}_{\pi} \texttt{CSWITCH}(e,L,c) \Rightarrow \rho,\mu_l,\chi_l,v_l}(\textsc{Switch})$$

Where for $L = [(j_1,l_1);\cdots;(j_n,l_n)]$, $L(v_e)$ is defined as follows :
Let $I_i = \{j_1,\cdots,j_i\}$ for $1 \leqslant i \leqslant n$, $I_{n+1} = \mathbb{Z}_{64}$.
$\tilde{j} \triangleq \min_{1\leqslant i\leqslant n+1}\{i \mid v_e \in I_i\}$, finally $L(v_e) \triangleq [l_{\tilde{j}};\cdots;l_n;c]$.