

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

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

18 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 $(\rho : \mathcal{S} \rightarrow \mathbb{Z}_{64}) \in \mathcal{P}$ the environment, where \mathcal{S} is the set of names of variables and functions, $(\mu : \mathbb{Z}_{64} \rightarrow \mathbb{Z}_8) \in \mathcal{M}$ 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 $\mathcal{E} \triangleq \mathcal{S} \sqcup \{\mathbf{brk}, \mathbf{ret}, \mathbf{cnt}, \mathbf{nil}\}$: 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 $\text{fun}_{\pi}^n : \mathbb{Z}_{64} \rightarrow \mathbf{code}$, a wrapper around $\pm C$ functions : $\text{fun}_{\pi}^n(a)(p_1, \dots, p_n) = c$ updates the environment with p_1, \dots, 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 $\mu^{64}[x \mapsto v] \triangleq \mu[x+k \mapsto v_k \mid 0 \leq k < 8, v = \sum_{k=0}^7 2^{8k} v_k]$, 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 :

$$\frac{x \in \text{dom } \rho \quad \rho(x) \in \text{dom } \mu}{\rho, \mu, \mathbf{nil}, v \vdash_{\pi} \mathbf{VAR } x \Rightarrow \rho, \mu, \mathbf{nil}, \mu^{64}(\rho(x))} (\mathbf{VAR})$$

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

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

For constant integers :

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

For strings :

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

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

For arrays :

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

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

1.2 Unary operators without side-effects

Unary minus (same as C—) :

$$\frac{\rho, \mu, \chi, v \vdash_{\pi} e \Rightarrow \rho, \mu_e, \mathbf{nil}, v_e}{\rho, \mu, \chi, v \vdash_{\pi} \mathbf{OP1}(\mathbf{M_MINUS}, e) \Rightarrow \rho, \mu_e, \mathbf{nil}, -v_e} (\text{NEG})$$

Unary bitwise negation (same as C—) :

$$\frac{\rho, \mu, \chi, v \vdash_{\pi} e \Rightarrow \rho, \mu_e, \mathbf{nil}, v_e}{\rho, \mu, \chi, v \vdash_{\pi} \mathbf{OP1}(\mathbf{M_NOT}, e) \Rightarrow \rho, \mu_e, \mathbf{nil}, -v_e - 1} (\text{NOT})$$

Indirection (added in $\pm C$) :

$$\frac{x \in \text{dom } \rho}{\rho, \mu, \mathbf{nil}, v \vdash_{\pi} \mathbf{OP1}(\mathbf{M_ADDR}, \mathbf{VAR} \ x) \Rightarrow \rho, \mu, \mathbf{nil}, \rho(x)} (\text{VAR}^{\&})$$

$$\frac{\begin{array}{c} \rho, \mu, \chi, v \vdash_{\pi} i \Rightarrow \rho, \mu_i, \chi_i, v_i \\ \rho, \mu_i, \chi_i, v_i \vdash_{\pi} a \Rightarrow \rho, \mu_a, \mathbf{nil}, v_a \end{array}}{\rho, \mu, \chi, v \vdash_{\pi} \mathbf{OP1}(\mathbf{M_ADDR}, \mathbf{OP2}(\mathbf{S_INDEX}, a, i)) \Rightarrow \rho, \mu_a, \mathbf{nil}, v_a + v_i \times 8} (\text{IDX}^{\&})$$

$$\frac{\rho, \mu, \chi, v \vdash_{\pi} a \Rightarrow \rho, \mu_a, \mathbf{nil}, v_a}{\rho, \mu, \chi, v \vdash_{\pi} \mathbf{OP1}(\mathbf{M_ADDR}, \mathbf{OP1}(\mathbf{M_DEREF}, a)) \Rightarrow \rho, \mu_a, \mathbf{nil}, v_a} (\text{PTR}^{\&})$$

Dereferencing (added in $\pm C$) :

$$\frac{\rho, \mu, \chi, v \vdash_{\pi} a \Rightarrow \rho, \mu_a, \mathbf{nil}, v_a \quad v_a \in \text{dom } \mu_a}{\rho, \mu, \chi, v \vdash_{\pi} \mathbf{OP1}(\mathbf{M_DEREF}, a) \Rightarrow \rho, \mu_a, \mathbf{nil}, \mu_a^{64}(v_a)} (\text{PTR})$$

When the operand raises a non- \mathbf{nil} flag :

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

1.3 Binary operators

Multiplication (same as C—) :

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

Addition (same as C—) :

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

Subtraction (same as C—) :

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

Division and remainder (same as C—) :

$$\frac{\begin{array}{c} \rho, \mu, \chi, v \vdash_{\pi} e_2 \Rightarrow \rho, \mu_2, \chi_2, v_2 \quad v_2 \neq 0 \\ \rho, \mu_2, \chi_2, v_2 \vdash_{\pi} e_1 \Rightarrow \rho, \mu_1, \mathbf{nil}, v_1 \end{array}}{\rho, \mu, \chi, v \vdash_{\pi} \mathbf{OP2}(\mathbf{S_DIV}, e_1, e_2) \Rightarrow \rho, \mu_1, \mathbf{nil}, v_1 \text{ div } v_2} (\text{DIV})$$

$$\frac{\begin{array}{c} \rho, \mu, \chi, v \vdash_{\pi} e_2 \Rightarrow \rho, \mu_2, \chi_2, v_2 \quad v_2 \neq 0 \\ \rho, \mu_2, \chi_2, v_2 \vdash_{\pi} e_1 \Rightarrow \rho, \mu_1, \mathbf{nil}, v_1 \end{array}}{\rho, \mu, \chi, v \vdash_{\pi} \text{OP2}(\mathbf{S_MOD}, e_1, e_2) \Rightarrow \rho, \mu_1, \mathbf{nil}, v_1 \bmod v_2} (\text{MOD})$$

Shifts (added in $\pm C$) :

$$\frac{\begin{array}{c} \rho, \mu, \chi, v \vdash_{\pi} e_2 \Rightarrow \rho, \mu_2, \chi_2, v_2 \\ \rho, \mu_2, \chi_2, v_2 \vdash_{\pi} e_1 \Rightarrow \rho, \mu_1, \mathbf{nil}, v_1 \end{array}}{\rho, \mu, \chi, v \vdash_{\pi} \text{OP2}(\mathbf{S_SHL}, e_1, e_2) \Rightarrow \rho, \mu_1, \mathbf{nil}, v_1 \times 2^{v_2}} (\text{SHL})$$

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

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

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

and $\text{bin}_{64} = \text{dec}_{64}^{-1}$.

We can now define bitwise operators as follows (added in $\pm C$).

$$\frac{\begin{array}{c} \rho, \mu, \chi, v \vdash_{\pi} e_2 \Rightarrow \rho, \mu_2, \chi_2, v_2 \quad \rho, \mu_2, \chi_2, v_2 \vdash_{\pi} e_1 \Rightarrow \rho, \mu_1, \mathbf{nil}, v_1 \\ (b_0^2, \dots, b_{63}^2) = \text{bin}_{64}(v_2) \quad (b_0^1, \dots, b_{63}^1) = \text{bin}_{64}(v_1) \end{array}}{\rho, \mu, \chi, v \vdash_{\pi} \text{OP2}(\mathbf{S_AND}, e_1, e_2) \Rightarrow \rho, \mu_1, \mathbf{nil}, \text{dec}_{64}(b_0^1 \wedge b_0^2, \dots, b_{63}^1 \wedge b_{63}^2), \mu_1} (\text{AND})$$

$$\frac{\begin{array}{c} \rho, \mu, \chi, v \vdash_{\pi} e_2 \Rightarrow \rho, \mu_2, \chi_2, v_2 \quad \rho, \mu_2, \chi_2, v_2 \vdash_{\pi} e_1 \Rightarrow \rho, \mu_1, \mathbf{nil}, v_1 \\ (b_0^2, \dots, b_{63}^2) = \text{bin}_{64}(v_2) \quad (b_0^1, \dots, b_{63}^1) = \text{bin}_{64}(v_1) \end{array}}{\rho, \mu, \chi, v \vdash_{\pi} \text{OP2}(\mathbf{S_OR}, e_1, e_2) \Rightarrow \rho, \mu_1, \mathbf{nil}, \text{dec}_{64}(b_0^1 \vee b_0^2, \dots, b_{63}^1 \vee b_{63}^2), \mu_1} (\text{IOR})$$

$$\frac{\begin{array}{c} \rho, \mu, \chi, v \vdash_{\pi} e_2 \Rightarrow \rho, \mu_2, \chi_2, v_2 \quad \rho, \mu_2, \chi_2, v_2 \vdash_{\pi} e_1 \Rightarrow \rho, \mu_1, \mathbf{nil}, v_1 \\ (b_0^2, \dots, b_{63}^2) = \text{bin}_{64}(v_2) \quad (b_0^1, \dots, b_{63}^1) = \text{bin}_{64}(v_1) \end{array}}{\rho, \mu, \chi, v \vdash_{\pi} \text{OP2}(\mathbf{S_XOR}, e_1, e_2) \Rightarrow \rho, \mu_1, \mathbf{nil}, \text{dec}_{64}(b_0^1 \oplus b_0^2, \dots, b_{63}^1 \oplus b_{63}^2), \mu_1} (\text{XOR})$$

When one of the operands raises a non- \mathbf{nil} flag :

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

1.4 Comparisons

All are the same as in C—.

$$\frac{\begin{array}{c} \rho, \mu, \chi, v \vdash_{\pi} e_2 \Rightarrow \rho, \mu_2, \chi_2, v_2 \\ \rho, \mu_2, \chi_2, v \vdash_{\pi} e_1 \Rightarrow \rho, \mu_1, \mathbf{nil}, v_1 \\ v_1 = v_2 \end{array}}{\rho, \mu, \mathbf{nil}, v \vdash_{\pi} \text{CMP}(\mathbf{C_EQ}, e_1, e_2) \Rightarrow \rho, \mu_1, \mathbf{nil}, 1} (\text{EQ}^{\top})$$

$$\frac{\begin{array}{c} \rho, \mu, \chi, v \vdash_{\pi} e_2 \Rightarrow \rho, \mu_2, \chi_2, v_2 \\ \rho, \mu_2, \chi_2, v \vdash_{\pi} e_1 \Rightarrow \rho, \mu_1, \mathbf{nil}, v_1 \\ v_1 < v_2 \end{array}}{\rho, \mu, \mathbf{nil}, v \vdash_{\pi} \text{CMP}(\mathbf{C_LT}, e_1, e_2) \Rightarrow \rho, \mu_1, \mathbf{nil}, 1} (\text{LT}^{\top})$$

$$\frac{\begin{array}{c} \rho, \mu, \chi, v \vdash_{\pi} e_2 \Rightarrow \rho, \mu_2, \chi_2, v_2 \\ \rho, \mu_2, \chi_2, v \vdash_{\pi} e_1 \Rightarrow \rho, \mu_1, \mathbf{nil}, v_1 \\ v_1 \leq v_2 \end{array}}{\rho, \mu, \mathbf{nil}, v \vdash_{\pi} \text{CMP}(\mathbf{C_LE}, e_1, e_2) \Rightarrow \rho, \mu_1, \mathbf{nil}, 1} (\text{LE}^{\top})$$

$$\frac{\begin{array}{c} \rho, \mu, \chi, v \vdash_{\pi} e_2 \Rightarrow \rho, \mu_2, \chi_2, v_2 \\ \rho, \mu_2, \chi_2, v \vdash_{\pi} e_1 \Rightarrow \rho, \mu_1, \mathbf{nil}, v_1 \\ v_1 \neq v_2 \end{array}}{\rho, \mu, \mathbf{nil}, v \vdash_{\pi} \text{CMP}(\mathbf{C_EQ}, e_1, e_2) \Rightarrow \rho, \mu_1, \mathbf{nil}, 0} (\text{EQ}^{\perp})$$

$$\frac{\begin{array}{c} \rho, \mu, \chi, v \vdash_{\pi} e_2 \Rightarrow \rho, \mu_2, \chi_2, v_2 \\ \rho, \mu_2, \chi_2, v \vdash_{\pi} e_1 \Rightarrow \rho, \mu_1, \mathbf{nil}, v_1 \\ v_1 \not\leq v_2 \end{array}}{\rho, \mu, \mathbf{nil}, v \vdash_{\pi} \text{CMP}(\mathbf{C_LT}, e_1, e_2) \Rightarrow \rho, \mu_1, \mathbf{nil}, 0} (\text{LT}^{\perp})$$

$$\begin{array}{c}
\rho, \mu, \chi, v \vdash_{\pi} e_2 \Rightarrow \rho, \mu_2, \chi_2, v_2 \\
\rho, \mu_2, \chi_2, v \vdash_{\pi} e_1 \Rightarrow \rho, \mu_1, \mathbf{nil}, v_1 \\
v_1 \not\leq v_2 \\
\hline
\rho, \mu, \mathbf{nil}, v \vdash_{\pi} \text{CMP}(\text{C_LE}, e_1, e_2) \Rightarrow \rho, \mu_1, \mathbf{nil}, 0 \quad (\text{LE}^{\perp})
\end{array}$$

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

$$\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, \mathbf{nil}, v_1 \\
v_1 = v_2 \\
\hline
\rho, \mu, \mathbf{nil}, v \vdash_{\pi} \text{CMP}(\text{C_NE}, e_1, e_2) \Rightarrow \rho, \mu_1, \mathbf{nil}, 0 \quad (\text{NE}^{\perp}) \\
\\
\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, \mathbf{nil}, v_1 \\
v_1 \leq v_2 \\
\hline
\rho, \mu, \mathbf{nil}, v \vdash_{\pi} \text{CMP}(\text{C_GT}, e_1, e_2) \Rightarrow \rho, \mu_1, \mathbf{nil}, 0 \quad (\text{GT}^{\perp}) \\
\\
\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, \mathbf{nil}, v_1 \\
v_1 < v_2 \\
\hline
\rho, \mu, \mathbf{nil}, v \vdash_{\pi} \text{CMP}(\text{C_GE}, e_1, e_2) \Rightarrow \rho, \mu_1, \mathbf{nil}, 0 \quad (\text{GE}^{\perp}) \\
\\
\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, \mathbf{nil}, v_1 \\
v_1 \neq v_2 \\
\hline
\rho, \mu, \mathbf{nil}, v \vdash_{\pi} \text{CMP}(\text{C_NE}, e_1, e_2) \Rightarrow \rho, \mu_1, \mathbf{nil}, 1 \quad (\text{NE}^{\top}) \\
\\
\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, \mathbf{nil}, v_1 \\
v_1 \not\leq v_2 \\
\hline
\rho, \mu, \mathbf{nil}, v \vdash_{\pi} \text{CMP}(\text{C_GT}, e_1, e_2) \Rightarrow \rho, \mu_1, \mathbf{nil}, 1 \quad (\text{GT}^{\top}) \\
\\
\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, \mathbf{nil}, v_1 \\
v_1 \not< v_2 \\
\hline
\rho, \mu, \mathbf{nil}, v \vdash_{\pi} \text{CMP}(\text{C_GE}, e_1, e_2) \Rightarrow \rho, \mu_1, \mathbf{nil}, 1 \quad (\text{GE}^{\top})
\end{array}$$

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

$$\begin{array}{c}
\rho, \mu, \chi, v \vdash_{\pi} e_2 \Rightarrow \rho, \mu_2, \chi_2, v_2 \\
\rho, \mu_2, \chi_2, v_2 \vdash_{\pi} e_1 \Rightarrow \rho, \mu_1, \chi_1, v_1 \\
\chi_1 \neq \mathbf{nil} \\
\hline
\rho, \mu, \chi, v \vdash_{\pi} \text{CMP}(op, e_1, e_2) \Rightarrow \rho, \mu_1, \chi_1, v_1 \quad (\text{CMP}^x)
\end{array}$$

1.5 Assignments

$$\begin{array}{c}
\rho, \mu, \chi, v \vdash_{\pi} e \Rightarrow \rho, \mu_e, \mathbf{nil}, v_e \\
x \in \text{dom } \rho \quad \rho(x) \in \text{dom } \mu_e \\
\hline
\rho, \mu, \chi, v \vdash_{\pi} \text{SET_VAR}(x, e) \Rightarrow \rho, \mu_e^{64}[\rho(x) \mapsto v_e], \mathbf{nil}, v_e \quad (\text{VAR}^{\leftarrow}) \\
\\
\rho, \mu, \chi, v \vdash_{\pi} e \Rightarrow \rho, \mu_e, \chi_e, v_e \quad \chi_e \neq \mathbf{nil} \\
\rho, \mu, \chi, v \vdash_{\pi} \text{SET_VAR}(x, e) \Rightarrow \rho, \mu_e, \chi_e, v_e \quad (\text{VAR}^{\leftarrow x}) \\
\\
\rho, \mu, \chi, v \vdash_{\pi} e \Rightarrow \rho, \mu_e, \chi_e, v_e \\
\rho, \mu_e, \chi_e, v_e \vdash_{\pi} i \Rightarrow \rho, \mu_i, \mathbf{nil}, v_i \\
\rho(x) \in \text{dom } \mu_i \\
\hline
\rho, \mu, \chi, v \vdash_{\pi} \text{SET_ARRAY}(x, i, e) \Rightarrow \rho, \mu_i^{64}[\rho(x) + v_i \times 8 \mapsto v_e], \mathbf{nil}, v_e \quad (\text{IDX}^{\leftarrow}) \\
\\
\rho, \mu, \chi, v \vdash_{\pi} e \Rightarrow \rho, \mu_e, \chi_e, v_e \\
\rho, \mu_e, \chi_e, v_e \vdash_{\pi} i \Rightarrow \rho, \mu_i, \chi_i, v_i \quad \chi_i \neq \mathbf{nil} \\
\rho, \mu, \chi, v \vdash_{\pi} \text{SET_ARRAY}(x, i, e) \Rightarrow \rho, \mu_i, \chi_i, v_i \quad (\text{IDX}^{\leftarrow x}) \\
\\
\rho, \mu, \chi, v \vdash_{\pi} e \Rightarrow \rho, \mu_e, \chi_e, v_e \\
\rho, \mu_e, \chi_e, v_e \vdash_{\pi} a \Rightarrow \rho, \mu_a, \mathbf{nil}, v_a \\
\hline
\rho, \mu, \chi, v \vdash_{\pi} \text{SET_DEREF}(a, e) \Rightarrow \rho, \mu_a^{64}[v_a \mapsto v_e], \mathbf{nil}, v_e \quad (\text{PTR}^{\leftarrow}) \\
\\
\rho, \mu, \chi, v \vdash_{\pi} e \Rightarrow \rho, \mu_e, \chi_e, v_e \\
\rho, \mu_e, \chi_e, v_e \vdash_{\pi} a \Rightarrow \rho, \mu_a, \chi_a, v_a \quad \chi_a \neq \mathbf{nil} \\
\rho, \mu, \chi, v \vdash_{\pi} \text{SET_DEREF}(a, e) \Rightarrow \rho, \mu_a, \chi_a, v_a \quad (\text{PTR}^{\leftarrow x})
\end{array}$$

1.6 Increments

On variables :

$$\begin{array}{c}
\frac{x \in \text{dom } \rho \quad \rho(x) = k \in \text{dom } \mu \quad \mu^{64}(k) = v_k}{\rho, \mu, \text{nil}, v \vdash_{\pi} \text{OP1}(\text{M_POST_INC}, \text{VAR } x) \Rightarrow \rho, \mu^{64}[k \mapsto v_k + 1], \text{nil}, v_k} (\text{VAR}^{\bullet\uparrow}) \\
\frac{x \in \text{dom } \rho \quad \rho(x) = k \in \text{dom } \mu \quad \mu^{64}(k) = v_k}{\rho, \mu, \text{nil}, v \vdash_{\pi} \text{OP1}(\text{M_POST_DEC}, \text{VAR } x) \Rightarrow \rho, \mu^{64}[k \mapsto v_k - 1], \text{nil}, v_k} (\text{VAR}^{\bullet\downarrow}) \\
\frac{x \in \text{dom } \rho \quad \rho(x) = k \in \text{dom } \mu \quad \mu^{64}(k) + 1 = v_k}{\rho, \mu, \text{nil}, v \vdash_{\pi} \text{OP1}(\text{M_PRE_INC}, \text{VAR } x) \Rightarrow \rho, \mu^{64}[k \mapsto v_k], \text{nil}, v_k} (\text{VAR}^{\uparrow\bullet}) \\
\frac{x \in \text{dom } \rho \quad \rho(x) = k \in \text{dom } \mu \quad \mu^{64}(k) - 1 = v_k}{\rho, \mu, \text{nil}, v \vdash_{\pi} \text{OP1}(\text{M_PRE_DEC}, \text{VAR } x) \Rightarrow \rho, \mu^{64}[k \mapsto v_k], \text{nil}, v_k} (\text{VAR}^{\downarrow\bullet})
\end{array}$$

On arrays :

$$\begin{array}{c}
\frac{\rho, \mu, \chi, v \vdash_{\pi} i \Rightarrow \rho, \mu_i, \chi_i, v_i \quad \rho, \mu_i, \chi_i, v_i \vdash_{\pi} a \Rightarrow \rho, \mu_a, \text{nil}, v_a \quad v_a + v_i \times 8 = k \quad k \in \text{dom } \mu_a}{\rho, \mu, \chi, v \vdash_{\pi} \text{OP1}(\text{M_POST_INC}, \text{OP2}(\text{S_INDEX}, a, e)) \Rightarrow \rho, \mu_a^{64}[k \mapsto \mu_a(k) + 1], \text{nil}, \mu_a^{64}(k)} (\text{IDX}^{\bullet\uparrow}) \\
\frac{\rho, \mu, \chi, v \vdash_{\pi} i \Rightarrow \rho, \mu_i, \chi_i, v_i \quad \rho, \mu_i, \chi_i, v_i \vdash_{\pi} a \Rightarrow \rho, \mu_a, \text{nil}, v_a \quad v_a + v_i \times 8 = k \quad k \in \text{dom } \mu_a}{\rho, \mu, \chi, v \vdash_{\pi} \text{OP1}(\text{M_POST_DEC}, \text{OP2}(\text{S_INDEX}, a, e)) \Rightarrow \rho, \mu_a^{64}[k \mapsto \mu_a(k) - 1], \text{nil}, \mu_a^{64}(k)} (\text{IDX}^{\bullet\downarrow}) \\
\frac{\rho, \mu, \chi, v \vdash_{\pi} i \Rightarrow \rho, \mu_i, \chi_i, v_i \quad \rho, \mu_i, \chi_i, v_i \vdash_{\pi} a \Rightarrow \rho, \mu_a, \text{nil}, v_a \quad v_a + v_i \times 8 = k \quad k \in \text{dom } \mu_a \quad \mu_a^{64}(k) + 1 = v_k}{\rho, \mu, \chi, v \vdash_{\pi} \text{OP1}(\text{M_PRE_INC}, \text{OP2}(\text{S_INDEX}, a, e)) \Rightarrow \rho, \mu_a[k \mapsto v_k], \text{nil}, v_k} (\text{IDX}^{\uparrow\bullet}) \\
\frac{\rho, \mu, \chi, v \vdash_{\pi} i \Rightarrow \rho, \mu_i, \chi_i, v_i \quad \rho, \mu_i, \chi_i, v_i \vdash_{\pi} a \Rightarrow \rho, \mu_a, \text{nil}, v_a \quad v_a + v_i \times 8 = k \quad k \in \text{dom } \mu_a \quad \mu_a^{64}(k) - 1 = v_k}{\rho, \mu, \chi, v \vdash_{\pi} \text{OP1}(\text{M_PRE_DEC}, \text{OP2}(\text{S_INDEX}, a, e)) \Rightarrow \rho, \mu_a[k \mapsto v_k], \text{nil}, v_k} (\text{IDX}^{\downarrow\bullet})
\end{array}$$

On dereferences :

$$\begin{array}{c}
\frac{\rho, \mu, \chi, v \vdash_{\pi} e \Rightarrow \rho, \mu_e, \text{nil}, v_e}{\rho, \mu, \chi, v \vdash_{\pi} \text{OP1}(\text{M_POST_INC}, \text{OP1}(\text{M_DEREF}, e)) \Rightarrow \rho, \mu_e^{64}[v_e \mapsto \mu_e(v_e) + 1], \text{nil}, \mu_e(v_e)} (\text{PTR}^{\bullet\uparrow}) \\
\frac{\rho, \mu, \chi, v \vdash_{\pi} e \Rightarrow \rho, \mu_e, \text{nil}, v_e}{\rho, \mu, \chi, v \vdash_{\pi} \text{OP1}(\text{M_POST_DEC}, \text{OP1}(\text{M_DEREF}, e)) \Rightarrow \rho, \mu_e^{64}[v_e \mapsto \mu_e(v_e) - 1], \text{nil}, \mu_e^{64}(v_e)} (\text{PTR}^{\bullet\downarrow}) \\
\frac{\rho, \mu, \chi, v \vdash_{\pi} e \Rightarrow \rho, \mu_e, \text{nil}, v_e \quad \mu_e^{64}(v_e) + 1 = k}{\rho, \mu, \chi, v \vdash_{\pi} \text{OP1}(\text{M_PRE_INC}, \text{OP1}(\text{M_DEREF}, e)) \Rightarrow \rho, \mu_e[v_e \mapsto k], \text{nil}, k} (\text{PTR}^{\uparrow\bullet}) \\
\frac{\rho, \mu, \chi, v \vdash_{\pi} e \Rightarrow \rho, \mu_e, \text{nil}, v_e \quad \mu_e^{64}(v_e) - 1 = k}{\rho, \mu, \chi, v \vdash_{\pi} \text{OP1}(\text{M_PRE_DEC}, \text{OP1}(\text{M_DEREF}, e)) \Rightarrow \rho, \mu_e[v_e \mapsto k], \text{nil}, k} (\text{PTR}^{\downarrow\bullet})
\end{array}$$

1.7 Extended assignments

Let $op \in \text{bin_op} \setminus \{\text{S_INDEX}\}$.

On variables :

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

On arrays :

$$\frac{\rho, \mu, \chi, v \vdash_{\pi} e \Rightarrow \rho, \mu_e, \chi_e, v_e \quad \rho, \mu_e, \chi_e, v_e \vdash_{\pi} i \Rightarrow \rho, \mu_i, \chi_i, v_i \quad t \in \text{dom } \rho \quad \rho(t) + v_i \times 8 = k \quad k \in \text{dom } \mu_i \quad \mu_i^{64}(k) = u \quad \rho, \mu_i, \chi_i, v_i \vdash_{\pi} \text{OP2}(op, \text{CST } v, \text{CST } u) \Rightarrow \rho, \mu', \text{nil}, w}{\rho, \mu, \chi, v \vdash_{\pi} \text{OPSET_ARRAY}(op, t, e_1, e_2) \Rightarrow \rho, \mu'[k \mapsto w], \text{nil}, w} (\text{IDX}^{\leftarrow op})$$

$$\frac{\rho, \mu, \chi, v \vdash_{\pi} e \Rightarrow \rho, \mu_e, \chi_e, v_e \quad \rho, \mu_e, \chi_e, v_e \vdash_{\pi} i \Rightarrow \rho, \mu_i, \chi_i, v_i \quad \chi_i \neq \mathbf{nil}}{\rho, \mu, \chi, v \vdash_{\pi} \mathbf{OPSET_ARRAY}(op, t, e_1, e_2) \Rightarrow \rho, \mu_i, \chi_i, v_i} (\mathbf{IDX}^{\leftarrow op\chi})$$

On dereferences :

$$\frac{\rho, \mu, \chi, v \vdash_{\pi} a \Rightarrow \rho, \mu_a, \chi_a, v_a \quad \rho, \mu_a, \chi_a, v_a \vdash_{\pi} e \Rightarrow \rho, \mu_e, \chi_e, v_e \quad v_e \in \text{dom } \mu_e \quad \mu_e^{64}(v_e) = u}{\rho, \mu_e, \chi_e, v_e \vdash_{\pi} \mathbf{OP2}(op, \mathbf{CST } v, \mathbf{CST } u) \Rightarrow \rho, \mu', \mathbf{nil}, w} (\mathbf{PTR}^{\leftarrow op})$$

$$\frac{\rho, \mu, \chi, v \vdash_{\pi} a \Rightarrow \rho, \mu_a, \chi_a, v_a \quad \rho, \mu_a, \chi_a, v_a \vdash_{\pi} e \Rightarrow \rho, \mu_e, \chi_e, v_e \quad \chi_e \neq \mathbf{nil}}{\rho, \mu, \chi, v \vdash_{\pi} \mathbf{OPSET_DEREF}(op, e_1, e_2) \Rightarrow \rho, \mu'[k \mapsto w], \mathbf{nil}, w} (\mathbf{PTR}^{\leftarrow op\chi})$$

1.8 Ternary operator

$$\frac{\rho, \mu, \chi, v \vdash_{\pi} e \Rightarrow \rho, \mu_e, \mathbf{nil}, v_e \quad v_e = 0 \quad \rho, \mu_e, \mathbf{nil}, v_e \vdash_{\pi} e_{\perp} \Rightarrow \rho, \mu_{\perp}, \chi_{\perp}, v_{\perp}}{\rho, \mu, \chi, v \vdash_{\pi} \mathbf{EIF}(e, e_{\top}, e_{\perp}) \Rightarrow \rho, \mu_{\perp}, \chi_{\perp}, v_{\perp}} (\mathbf{TERN}^{\perp})$$

$$\frac{\rho, \mu, \chi, v \vdash_{\pi} e \Rightarrow \rho, \mu_e, \mathbf{nil}, v_e \quad v_e \neq 0 \quad \rho, \mu_e, \mathbf{nil}, v_e \vdash_{\pi} e_{\top} \Rightarrow \rho, \mu_{\top}, \chi_{\top}, v_{\top}}{\rho, \mu, \chi, v \vdash_{\pi} \mathbf{EIF}(e, e_{\top}, e_{\perp}) \Rightarrow \rho, \mu_{\top}, \chi_{\top}, v_{\top}} (\mathbf{TERN}^{\top})$$

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

1.9 Sequence

$$\frac{\rho, \mu_0, \chi_0, v_0 \vdash_{\pi} e_1 \Rightarrow \rho, \mu_1, \chi_1, v_1 \quad \dots \quad \rho, \mu_{n-1}, \chi_{n-1}, v_{n-1} \vdash_{\pi} e_n \Rightarrow \rho, \mu_n, \chi_n, v_n}{\rho, \mu_0, \chi_0, v_0 \vdash_{\pi} \mathbf{ESEQ}[e_1; \dots; e_n] \Rightarrow \rho, \mu_n, \chi_n, v_n} (\mathbf{SEQ}^n)$$

1.10 Function call

Works for both a toplevel function and a function pointer :

$$\frac{\rho, \mu_{n+1}, \chi_{n+1}, v_{n+1} \vdash_{\pi} e_n \Rightarrow \rho, \mu_n, \chi_n, v_n \quad \dots \quad \rho, \mu_2, \chi_2, v_2 \vdash_{\pi} e_1 \Rightarrow \rho, \mu_1, \mathbf{nil}, v_1 \quad f \in \text{dom fun}_{\pi}^n}{\rho_g, \mu_1, \mathbf{nil}, 0 \vdash_{\pi} \text{fun}_{\pi}^n(f)(v_1, \dots, v_n) \Rightarrow \rho_f, \mu_f, \chi_f, v_f} (\mathbf{CALL}^n)$$

$$\frac{\rho, \mu_{n+1}, \chi_{n+1}, v_{n+1} \vdash_{\pi} e_n \Rightarrow \rho, \mu_n, \chi_n, v_n \quad \dots \quad \rho, \mu_2, \chi_2, v_2 \vdash_{\pi} e_1 \Rightarrow \rho, \mu_1, \chi_1, v_1 \quad \chi_1 \neq \mathbf{nil}}{\rho, \mu_{n+1}, \chi_{n+1}, v_{n+1} \vdash_{\pi} \mathbf{CALL}(f, [e_1; \dots; e_n]) \Rightarrow \rho, \mu_1, \chi_1, v_1} (\mathbf{CALL}^{\chi})$$

2 Code

2.1 Expressions

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

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

2.2 Conditional branching

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

$$\frac{\begin{array}{l} \rho, \mu, \chi, v \vdash_{\pi} e \Rightarrow \rho, \mu_e, \mathbf{nil}, v_e \quad v_e = 0 \\ \rho, \mu_e, \mathbf{nil}, v_e \vdash_{\pi} c_{\perp} \Rightarrow \rho_{\perp}, \mu_{\perp}, \chi_{\perp}, v_{\perp} \end{array}}{\rho, \mu, \chi, v \vdash_{\pi} \mathbf{CIF}(e, c_{\top}, c_{\perp}) \Rightarrow \rho, \mu_{\perp}, \chi_{\perp}, v_{\perp}} (\text{IF}^{\perp})$$

$$\frac{\begin{array}{l} \rho, \mu, \chi, v \vdash_{\pi} e \Rightarrow \rho, \mu_e, \mathbf{nil}, v_e \quad v_e \neq 0 \\ \rho, \mu_e, \mathbf{nil}, v_e \vdash_{\pi} c_{\top} \Rightarrow \rho_{\top}, \mu_{\top}, \chi_{\top}, v_{\top} \end{array}}{\rho, \mu, \chi, v \vdash_{\pi} \mathbf{CIF}(e, c_{\top}, c_{\perp}) \Rightarrow \rho, \mu_{\top}, \chi_{\top}, v_{\top}} (\text{IF}^{\top})$$

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.

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

2.3 Blocks

$$\frac{\begin{array}{l} \rho, \mu, \chi, v \vdash_{\pi} c \Rightarrow \rho', \mu', \chi', v' \\ \rho', \mu', \chi', v' \vdash_{\pi} \mathbf{CBLOCK} S \Rightarrow \rho'', \mu'', \chi'', v'' \end{array}}{\rho, \mu, \chi, v \vdash_{\pi} \mathbf{CBLOCK}(c :: S) \Rightarrow \rho, \mu'', \chi'', v''} (\text{BLOCK}^1)$$

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

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

2.4 Loops

A loop with a false condition stops :

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

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

$$\frac{\begin{array}{l} \rho, \mu, \chi, v \vdash_{\pi} c \Rightarrow \rho_c, \mu_c, \mathbf{nil}, v_c \\ \rho, \mu_c, \mathbf{nil}, v_c \vdash_{\pi} \mathbf{CWHILE}(e, c, \mathbf{None}, \mathbf{true}) \Rightarrow \rho, \mu_w, \chi_w, v_w \end{array}}{\rho, \mu, \chi, v \vdash_{\pi} \mathbf{CWHILE}(e, c, \mathbf{None}, \mathbf{false}) \Rightarrow \rho, \mu_w, \chi_w, v_w} (\text{WHILE}^{\top, \mathbf{None}, \mathbf{false}})$$

$$\frac{\begin{array}{l} \rho, \mu, \chi, v \vdash_{\pi} c \Rightarrow \rho_c, \mu_c, \chi_c, v_c \\ \rho, \mu_c, \chi_c, v_c \vdash_{\pi} f \Rightarrow \rho, \mu_f, \mathbf{nil}, v_f \\ \rho, \mu_f, \mathbf{nil}, v_f \vdash_{\pi} \mathbf{CWHILE}(e, c, \mathbf{Some} f, \mathbf{true}) \Rightarrow \rho, \mu_w, \chi_w, v_w \end{array}}{\rho, \mu, \chi, v \vdash_{\pi} \mathbf{CWHILE}(e, c, \mathbf{Some} f, \mathbf{false}) \Rightarrow \rho, \mu_w, \chi_w, v_w} (\text{WHILE}^{\top, \mathbf{Some}, \mathbf{false}})$$

A loop continues normally if its condition is nonzero :

$$\frac{\begin{array}{l} \rho, \mu, \chi, v \vdash_{\pi} e \Rightarrow \rho, \mu_e, \mathbf{nil}, v_e \quad v_e \neq 0 \\ \rho, \mu_e, \mathbf{nil}, v_e \vdash_{\pi} c \Rightarrow \rho_c, \mu_c, \chi_c, v_c \quad \chi_c \notin \{\mathbf{brk}, \mathbf{cnt}\} \\ \rho, \mu_c, \chi_c, v_c \vdash_{\pi} \mathbf{CWHILE}(e, c, \mathbf{None}, \mathbf{true}) \Rightarrow \rho, \mu_w, \chi_w, v_w \end{array}}{\rho, \mu, \chi, v \vdash_{\pi} \mathbf{CWHILE}(e, c, \mathbf{None}, \mathbf{true}) \Rightarrow \rho, \mu_w, \chi_w, v_w} (\text{WHILE}^{\top, \mathbf{None}, \mathbf{true}})$$

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

A flag skips the loop :

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

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

cnt executes the finally clause before continuing as normal :

$$\begin{array}{c}
\frac{\rho, \mu, \chi, v \vdash_{\pi} e \Rightarrow \rho, \mu_e, \mathbf{nil}, v_e \quad v_e \neq 0}{\rho, \mu_e, \mathbf{nil}, v_e \vdash_{\pi} c \Rightarrow \rho_c, \mu_c, \mathbf{cnt}, v_c} \\
\frac{\rho, \mu_c, \mathbf{nil}, v_c \vdash_{\pi} \mathbf{CWHILE}(e, c, \mathbf{None}, \mathbf{true}) \Rightarrow \rho, \mu_w, \chi_w, v_w}{\rho, \mu, \chi, v \vdash_{\pi} \mathbf{CWHILE}(e, c, \mathbf{None}, \mathbf{true}) \Rightarrow \rho, \mu_w, \chi_w, v_w} (\mathbf{WHILE}^{\mathbf{cnt}, \mathbf{None}, \mathbf{true}}) \\
\\
\frac{\rho, \mu, \chi, v \vdash_{\pi} e \Rightarrow \rho, \mu_e, \mathbf{nil}, v_e \quad v_e \neq 0}{\rho, \mu_e, \mathbf{nil}, v_e \vdash_{\pi} c \Rightarrow \rho_c, \mu_c, \mathbf{cnt}, v_c} \\
\frac{\rho, \mu_c, \mathbf{nil}, v_c \vdash_{\pi} f \Rightarrow \rho, \mu_f, \chi_f, v_f}{\rho, \mu_f, \chi_f, v_f \vdash_{\pi} \mathbf{CWHILE}(e, c, \mathbf{Some} \ f, \mathbf{true}) \Rightarrow \rho, \mu_w, \chi_w, v_w} \\
\frac{\rho, \mu, \chi, v \vdash_{\pi} \mathbf{CWHILE}(e, c, \mathbf{Some} \ f, \mathbf{true}) \Rightarrow \rho, \mu_w, \chi_w, v_w}{\rho, \mu, \chi, v \vdash_{\pi} \mathbf{CWHILE}(e, c, \mathbf{Some} \ f, \mathbf{true}) \Rightarrow \rho, \mu_w, \chi_w, v_w} (\mathbf{WHILE}^{\mathbf{cnt}, \mathbf{Some}, \mathbf{true}}) \\
\\
\frac{\rho, \mu, \chi, v \vdash_{\pi} c \Rightarrow \rho_c, \mu_c, \mathbf{cnt}, v_c}{\rho, \mu_c, \mathbf{nil}, v_c \vdash_{\pi} \mathbf{CWHILE}(e, c, \mathbf{None}, \mathbf{true}) \Rightarrow \rho, \mu_w, \chi_w, v_w} \\
\frac{\rho, \mu_c, \mathbf{nil}, v_c \vdash_{\pi} \mathbf{CWHILE}(e, c, \mathbf{None}, \mathbf{true}) \Rightarrow \rho, \mu_w, \chi_w, v_w}{\rho, \mu, \chi, v \vdash_{\pi} \mathbf{CWHILE}(e, c, \mathbf{None}, \mathbf{false}) \Rightarrow \rho, \mu_w, \chi_w, v_w} (\mathbf{WHILE}^{\mathbf{cnt}, \mathbf{None}, \mathbf{false}}) \\
\\
\frac{\rho, \mu, \chi, v \vdash_{\pi} c \Rightarrow \rho_c, \mu_c, \mathbf{cnt}, v_c}{\rho, \mu_c, \mathbf{nil}, v_c \vdash_{\pi} f \Rightarrow \rho, \mu_f, \chi_f, v_f} \\
\frac{\rho, \mu_f, \chi_f, v_f \vdash_{\pi} \mathbf{CWHILE}(e, c, \mathbf{Some} \ f, \mathbf{true}) \Rightarrow \rho, \mu_w, \chi_w, v_w}{\rho, \mu, \chi, v \vdash_{\pi} \mathbf{CWHILE}(e, c, \mathbf{Some} \ f, \mathbf{false}) \Rightarrow \rho, \mu_w, \chi_w, v_w} (\mathbf{WHILE}^{\mathbf{cnt}, \mathbf{Some}, \mathbf{false}})
\end{array}$$

brk interrupts the loop but is not retransmitted :

$$\begin{array}{c}
\frac{\rho, \mu, \chi, v \vdash_{\pi} e \Rightarrow \rho, \mu_e, \mathbf{nil}, v_e \quad v_e \neq 0}{\rho, \mu_e, \mathbf{nil}, v_e \vdash_{\pi} c \Rightarrow \rho_c, \mu_c, \mathbf{brk}, v_c} \\
\frac{\rho, \mu_e, \mathbf{nil}, v_e \vdash_{\pi} c \Rightarrow \rho_c, \mu_c, \mathbf{brk}, v_c}{\rho, \mu, \chi, v \vdash_{\pi} \mathbf{CWHILE}(e, c, \mathbf{None}, \mathbf{true}) \Rightarrow \rho, \mu_c, \mathbf{nil}, v_c} (\mathbf{WHILE}^{\mathbf{brk}, \mathbf{true}}) \\
\\
\frac{\rho, \mu, \chi, v \vdash_{\pi} c \Rightarrow \rho_c, \mu_c, \mathbf{brk}, v_c}{\rho, \mu, \chi, v \vdash_{\pi} \mathbf{CWHILE}(e, c, \mathbf{None}, \mathbf{false}) \Rightarrow \rho, \mu_c, \mathbf{nil}, v_c} (\mathbf{WHILE}^{\mathbf{brk}, \mathbf{false}})
\end{array}$$

2.5 Control flow

$$\begin{array}{c}
\frac{}{\rho, \mu, \mathbf{nil}, v \vdash_{\pi} \mathbf{CBREAK} \Rightarrow \rho, \mu, \mathbf{brk}, 0} (\mathbf{BREAK}) \\
\\
\frac{\chi \neq \mathbf{nil}}{\rho, \mu, \chi, v \vdash_{\pi} \mathbf{CBREAK} \Rightarrow \rho, \mu, \chi, v} (\mathbf{BREAK}^{\chi}) \\
\\
\frac{}{\rho, \mu, \mathbf{nil}, v \vdash_{\pi} \mathbf{CCONTINUE} \Rightarrow \rho, \mu, \mathbf{cnt}, 0} (\mathbf{CONTINUE}) \\
\\
\frac{\chi \neq \mathbf{nil}}{\rho, \mu, \chi, v \vdash_{\pi} \mathbf{CCONTINUE} \Rightarrow \rho, \mu, \chi, v} (\mathbf{CONTINUE}^{\chi}) \\
\\
\frac{}{\rho, \mu, \mathbf{nil}, v \vdash_{\pi} \mathbf{CRETURN} \ \mathbf{None} \Rightarrow \rho, \mu, \mathbf{ret}, 0} (\mathbf{RETURN}^{\mathbf{None}}) \\
\\
\frac{\rho, \mu, \chi, v \vdash_{\pi} e \Rightarrow \rho, \mu_e, \mathbf{nil}, v_e}{\rho, \mu, \mathbf{nil}, v \vdash_{\pi} \mathbf{CRETURN}(\mathbf{Some} \ e) \Rightarrow \rho, \mu_e, \mathbf{ret}, v_e} (\mathbf{RETURN}^{\mathbf{Some}}) \\
\\
\frac{\chi \neq \mathbf{nil}}{\rho, \mu, \chi, v \vdash_{\pi} \mathbf{CRETURN} \ \mathbf{None} \Rightarrow \rho, \mu, \chi, v} (\mathbf{RETURN}^{\mathbf{None}\chi}) \\
\\
\frac{\rho, \mu, \chi, v \vdash_{\pi} e \Rightarrow \rho, \mu_e, \chi_e, v_e \quad \chi_e \neq \mathbf{nil}}{\rho, \mu, \mathbf{nil}, v \vdash_{\pi} \mathbf{CRETURN}(\mathbf{Some} \ e) \Rightarrow \rho, \mu_e, \chi_e, v_e} (\mathbf{RETURN}^{\mathbf{Some}\chi})
\end{array}$$