1 Definitions

1.1 Identifiers

$$
\begin{array}{rcl}
f & \in & \text{FieldName} \\
x & \in & \text{VarName}
\end{array}
$$

1.2 Values

$$
\begin{array}{rcl}
u & \in & \text{UnrestrictedValue} \\
v & \in & \text{Value} \\
r & \in & \text{Reference}
\end{array}
$$

1.3 Memory

$$
\begin{array}{rcrcl}
M & \in & \text{Memory} & = & \text{LocalMemory} \times \text{GlobalMemory} \\
 & & \text{LocalMemory} & = & \text{Var} \rightharpoonup \text{RuntimeVal} \\
 & & \text{GlobalMemory} & = & \text{AccountAddr} \rightharpoonup \text{Account} \\
 & & \text{Account} & = & \text{ModuleName} \rightharpoonup \text{Module} \\
 & & \text{Module} & = & \text{StructName} \rightharpoonup \text{StructSig} \\
 & & \text{StructSig} & = & \text{Kind} \times (\text{FieldName} \times \text{NonRefType})^*
\end{array}
$$

We define $M(l)$ to be the value stored at l in memory M, where l could be a local variable or a reference. We define $M[l \mapsto v]$ to be the memory with l updated to have value v, and otherwise identical with M. We use $M \setminus x$ to denote the memory with x removed, and otherwise identical with M.

2 Judgements

Judgement	**Meaning**
rq	reference r has mutability q
$M \rhd \kappa\tau\{f_1 \colon \tau_1, \ldots, f_n \colon \tau_n\}$	In memory M struct type τ has kind κ, field name f_i and field types τ_i
$\langle M, S\rangle \xrightarrow{i} \langle M', S'\rangle$	state $\langle M, S\rangle$ steps to $\langle M, S\rangle$ after executing instruction i

Table 1.

3 Operational Semantics

3.1 Local Instructions

$$\frac{M(x) = v \vee M(x) = r}{\langle M, S\rangle \xrightarrow{\text{MvLoc}\langle x\rangle} \langle M \setminus x, M(x) :: S\rangle}\ \textbf{MvLoc}$$

$$\frac{M(x) = u \vee M(x) = r}{\langle M, S\rangle \xrightarrow{\text{CpLoc}\langle x\rangle} \langle M, M(x) :: S\rangle}\ \textbf{CpLoc}$$

$$\frac{s = u \vee s = r}{\langle M, s :: S\rangle \xrightarrow{\text{StLoc}\langle x\rangle} \langle M[x \mapsto s], S\rangle}\ \textbf{StLoc}$$

$$\frac{M(x) = v}{\langle M, S\rangle \xrightarrow{\text{BorrowLoc}\langle x\rangle} \langle M, \mathbf{ref}\,\langle x, [], \mathbf{mut}\rangle\rangle}\ \textbf{BorrowLoc}$$

$$\frac{r = \mathbf{ref}\,\langle l, p, q\rangle}{\langle M, r :: S\rangle \xrightarrow{\text{BorrowField}\langle f\rangle} \langle M, \mathbf{ref}\,\langle l, p :: f, q\rangle :: S\rangle}\ \textbf{BorrowField}$$

$$\frac{r = \mathbf{ref}\,\langle l, p, q\rangle}{\langle M, r :: S\rangle \xrightarrow{\text{FreezeRef}} \langle M, \mathbf{ref}\,\langle M, \mathbf{ref}\,\langle l, p, \mathbf{immut}\rangle\rangle\rangle}\ \textbf{FreezeRef}$$

$$\frac{M(r) = u}{\langle M, r :: S\rangle \xrightarrow{\text{ReadRef}} \langle M, u :: S\rangle}\ \textbf{ReadRef}$$

$$\frac{r\,\mathbf{mut} \quad M(r) = u}{\langle M, v :: r :: S\rangle \xrightarrow{\text{WriteRef}} \langle M[r \mapsto v], S\rangle}\ \textbf{WriteRef}$$

$$\frac{s = u \vee s = r}{\langle M, s :: S\rangle \xrightarrow{\text{Pop}} \langle M, S\rangle}\ \textbf{Pop}$$

$$\frac{M \rhd \mathbf{resource}\,\tau\,\{f_1 \colon \tau_1, \ldots, f_n \colon \tau_n\}}{\langle M, [v_i]_{i=1}^n :: S\rangle \xrightarrow{\text{Pack}\langle\tau\rangle} \langle M, \mathbf{resource}\,\tau\,\{f_1 \colon v_1, \ldots, f_n \colon v_n\} :: S\rangle}\ \textbf{PackR}$$

$$\frac{M \rhd \mathbf{unrestricted}\,\tau\,\{f_1 \colon \tau_1, \ldots, f_n \colon \tau_n\}}{\langle M, [u_i]_{i=1}^n :: S\rangle \xrightarrow{\text{Pack}\langle\tau\rangle} \langle M, \mathbf{resource}\,\tau\,\{f_1 \colon u_1, \ldots, f_n \colon u_n\} :: S\rangle}\ \textbf{PackU}$$

$$\frac{}{\langle M, \kappa\tau\,\{f_1 \colon v_1, \ldots, f_n \colon v_n\} :: S\rangle \xrightarrow{\text{Unpack}} \langle M, v_1 :: \cdots :: v_n :: S\rangle}\ \textbf{Unpack}$$

$$\frac{}{\langle M, S\rangle \xrightarrow{\text{LoadConst}\langle a\rangle} \langle M, a :: S\rangle}\ \textbf{LoadConst}$$

$$\frac{|\mathrm{op}| = n}{\langle M, u_1 :: \cdots :: u_n :: S\rangle \xrightarrow{\text{Op}} \langle M, \mathrm{op}(u_1, \ldots, u_n) :: S\rangle}\ \textbf{StackOp}$$