The MOO Language: Syntax and Semantics

Doug Orleans

March 16, 1999

1 Syntax

2 Semantics

We define the sets of entities used in the operational semantics:

$$
\begin{aligned}
v \in \mathit{Value} &= \mathit{Int} + \mathit{Real} + \mathit{String} + \mathit{Objref} + \mathit{Err} + \mathit{List} \\
\mathit{Int} &= \{-2^{31}, \ldots, -1, 0, 1, \ldots, 2^{31}-1\} \\
\mathit{Real} &= \langle \text{IEEE 754 double precision floating point} \rangle - \{\text{Inf, NaN}\} \\
s \in \mathit{String} &= \mathit{ASCII}^* \\
o \in \mathit{Objref} &= \{\texttt{\#}\} \cdot \mathit{Int} \\
\mathit{Err} &= \left\{ \begin{array}{c} \text{E\_NONE, E\_TYPE, E\_DIV, E\_PERM, E\_PROPNF,} \\ \text{E\_VERBNF, E\_VARNF, E\_INVIND, E\_RECMOVE, E\_MAXREC,} \\ \text{E\_RANGE, E\_ARGS, E\_NACC, E\_INVARG, E\_QUOTA} \end{array} \right\} \\
\mathit{List} &= \mathit{Value}^* \\
\mathit{False} &= \{0, 0.0, \texttt{""}, \langle\rangle\} + \mathit{Objref} + \mathit{Err} \\
\mathit{True} &= \mathit{Value} - \mathit{False} \\
x &\in \mathit{Ident} \\
\sigma \in \mathit{Store} &= \mathit{Objref} \hookrightarrow \mathit{Obj} \\
\rho \in \mathit{Env} &= \mathit{Permission} \times \mathit{Objref} \times \mathit{String} \times \mathit{Bindings} \\
\beta \in \mathit{Bindings} &= \mathit{Ident} \hookrightarrow \mathit{Value}
\end{aligned}
$$

$$
\begin{aligned}
Obj &= Permission \times ObjectFlags \times Parent \times Fields \times Methods \\
\pi \in Permission &= Objref \\
\phi \in ObjectFlags &= P(\{\mathbf{r}, \mathbf{w}, \mathbf{f}, \mathbf{player}, \mathbf{programmer}, \mathbf{wizard}\}) \\
p \in Parent &= Objref \\
F \in Fields &= String \hookrightarrow Field \\
Field &= Permission \times FieldFlags \times Value \\
FieldFlags &= P(\{\mathbf{r}, \mathbf{w}, \mathbf{c}\}) \\
M \in Methods &= String \hookrightarrow Method \\
Method &= Permission \times MethodFlags \times prog \\
MethodFlags &= P(\{\mathbf{r}, \mathbf{w}, \mathbf{x}\}) \\
Q \in TaskQueue &= Task^* \\
\tau \in Task &= Taskid \times Time \times Cont \times Env \\
i \in Taskid &= Int \\
t \in Time &= Int + \{\infty\}
\end{aligned}
$$

The set of continuations $\kappa \in Cont$ is inductively defined below:

$$
\begin{aligned}
\kappa ::= {} & halt \\
| \quad & \langle \diamond, \kappa \rangle \\
| \quad & \langle [\,]\ S \ldots, \kappa \rangle \\
| \quad & \langle \text{if } ([\,])\ S_{then} \ldots \text{ else } S_{else} \ldots \text{ endif}, \kappa \rangle \\
| \quad & \langle x = [\,], \kappa \rangle \\
| \quad & \langle [\,].(M), \kappa \rangle \\
| \quad & \langle o.([\,]), \kappa \rangle \\
| \quad & \langle [\,]{:}(M)(arg_1, \ldots, arg_n), \kappa \rangle \\
| \quad & \langle o{:}([\,])(arg_1, \ldots, arg_n), \kappa \rangle \\
| \quad & \langle o{:}(s)(v_1, \ldots, v_{i-1}, [\,], arg_{i+1}, \ldots, arg_n), \kappa \rangle \\
| \quad & \langle [\,]\rho, \kappa \rangle \\
| \quad & \langle \text{pass}(v_1, \ldots, v_{i-1}, [\,], arg_{i+1}, \ldots, arg_n), \kappa \rangle \\
| \quad & \langle \text{create}([\,]), \kappa \rangle \\
| \quad & \langle \text{suspend}([\,]), \kappa \rangle \\
| \quad & \langle \text{resume}([\,], M), \kappa \rangle \\
| \quad & \langle \text{resume}(i, [\,]), \kappa \rangle \\
| \quad & \langle [\,]v, \kappa \rangle \\
| \quad & \langle \text{fork } x([\,])S \ldots \text{ endfork}, \kappa \rangle \\
| \quad & \langle \text{kill\_task}([\,]), \kappa \rangle \\
| \quad & \langle \text{set\_task\_perms}([\,]), \kappa \rangle
\end{aligned}
$$

We define some auxiliary functions for use in the semantics:

$$
\begin{gathered}
\text{quote} : \mathit{Ident} \to \mathit{String} \\
\text{quote}(x) = \langle \text{string representation of } x \rangle \\
\text{flags} : \mathit{Obj} \to \mathit{ObjectFlags} \\
\text{flags}(\langle \pi, \phi, p, F, M \rangle) = \phi \\
\text{parent} : \mathit{Obj} \to \mathit{Parent} \\
\text{parent}(\langle \pi, \phi, p, F, M \rangle) = p \\
\text{valid} : \mathit{Objref} \times \mathit{Store} \to \mathit{boolean} \\
\text{valid}(o, \sigma) \iff o \in \text{dom}(\sigma) \\
\text{perm} : \mathit{Env} \to \mathit{Permission} \\
\text{perm}(\langle \pi, \theta, \mu, \beta \rangle) = \pi \\
\text{this} : \mathit{Env} \to \mathit{Objref} \\
\text{this}(\langle \pi, \theta, \mu, \beta \rangle) = \theta \\
\text{cur\_meth} : \mathit{Env} \to \mathit{String} \\
\text{cur\_meth}(\langle \pi, \theta, \mu, \beta \rangle) = \mu \\
\text{bindings} : \mathit{Env} \to \mathit{Bindings} \\
\text{bindings}(\langle \pi, \theta, \mu, \beta \rangle) = \beta \\
\text{flags} : \mathit{Method} \to \mathit{MethodFlags} \\
\text{flags}(\langle \pi, \phi, \mathit{prog} \rangle) = \phi
\end{gathered}
$$

$$
\begin{gathered}
\text{lookup\_var} : \mathit{Ident} \times \mathit{Env} \to \mathit{Value} + \{\text{E\_VARNF}\} \\
\text{lookup\_var}(x, \langle \pi, \theta, \mu, \beta \rangle) = \begin{cases} \beta(x) & \text{if } x \in \text{dom}(\beta); \\ \text{E\_VARNF} & \text{otherwise.} \end{cases} \\
\text{lookup\_field} : \mathit{Obj} \times \mathit{String} \times \mathit{Store} \to \mathit{Field} + \{\text{E\_PROPNF}\} \\
\text{lookup\_field}(\langle \pi, \phi, p, F, M \rangle, f, \sigma) = \begin{cases} F(f) & \text{if } f \in \text{dom}(F); \\ \langle \pi', \phi', v \rangle & \begin{array}{l} \text{if } f \notin \text{dom}(F) \wedge \text{valid}(p, \sigma) \\ \quad \wedge\, \text{lookup\_field}(\sigma(p), f, \sigma) = \langle \pi', \phi', v \rangle \\ \quad \wedge\, \mathbf{c} \notin \phi'; \end{array} \\ \langle \pi, \phi', v \rangle & \begin{array}{l} \text{if } f \notin \text{dom}(F) \wedge \text{valid}(p, \sigma) \\ \quad \wedge\, \text{lookup\_field}(\sigma(p), f, \sigma) = \langle \pi', \phi', v \rangle \\ \quad \wedge\, \mathbf{c} \in \phi'; \end{array} \\ \text{E\_PROPNF} & \text{otherwise.} \end{cases} \\
\text{lookup\_method} : \mathit{Obj} \times \mathit{String} \times \mathit{Store} \to \mathit{Method} + \{\text{E\_VERBNF}\} \\
\text{lookup\_method}(\langle \pi, \phi, p, F, M \rangle, \mu, \sigma) = \begin{cases} M(\mu) & \text{if } \mu \in \text{dom}(M) \wedge \mathbf{x} \in \text{flags}(M(\mu)); \\ \text{lookup\_method}(\sigma(p), \mu, \sigma) & \begin{array}{l} \text{if } (\mu \notin \text{dom}(M) \vee \mathbf{x} \notin \text{flags}(M(\mu))) \\ \quad \wedge\, \text{valid}(p, \sigma); \end{array} \\ \text{E\_VERBNF} & \text{otherwise.} \end{cases}
\end{gathered}
$$

$$\text{wizard} : \textit{Permission} \times \textit{Store} \rightarrow \textit{boolean}$$
$$\text{wizard}(\pi, \sigma) \iff \text{valid}(\pi, \sigma) \wedge \textbf{wizard} \in \text{flags}(\sigma(\pi))$$
$$\text{controls} : \textit{Permission} \times \textit{Permission} \times \textit{Store} \rightarrow \textit{boolean}$$
$$\text{controls}(\pi_1, \pi_2, \sigma) \iff \pi_1 = \pi_2 \vee \text{wizard}(\pi_1, \sigma)$$
$$\text{can\_read} : \textit{Permission} \times \textit{Permission} \times \textit{FieldFlags} \times \textit{Store} \rightarrow \textit{boolean}$$
$$\text{can\_read}(\pi_1, \pi_2, \phi, \sigma) \iff \text{controls}(\pi_1, \pi_2, \sigma) \vee \mathbf{r} \in \phi$$
$$\text{can\_write} : \textit{Permission} \times \textit{Permission} \times \textit{FieldFlags} \times \textit{Store} \rightarrow \textit{boolean}$$
$$\text{can\_write}(\pi_1, \pi_2, \phi, \sigma) \iff \text{controls}(\pi_1, \pi_2, \sigma) \vee \mathbf{w} \in \phi$$
$$\text{can\_create} : \textit{Permission} \times \textit{Permission} \times \textit{ObjectFlags} \times \textit{Store} \rightarrow \textit{boolean}$$
$$\text{can\_create}(\pi_1, \pi_2, \phi, \sigma) \iff \text{controls}(\pi_1, \pi_2, \sigma) \vee \mathbf{f} \in \phi$$
$$\text{fresh\_objid} : \textit{Store} \rightarrow \textit{Objref}$$
$$\text{fresh\_objid}(\sigma) = \text{the smallest } o \in \textit{Objref} \text{ such that } \forall o' \in \text{dom}(\sigma), o > o'.$$
$$\text{fresh\_taskid} : \cdot \rightarrow \textit{Taskid}$$
$$\text{fresh\_taskid}() = \text{a task id that has not been used before}$$
$$\text{future} : \textit{Int} \rightarrow \textit{Time}$$
$$\text{future}(v) = \text{a time representing } v \text{ seconds from now}$$
$$\text{taskid} : \textit{Task} \rightarrow \textit{Taskid}$$
$$\text{taskid}(\langle i, t, \kappa, \rho \rangle) = i$$
$$\text{time} : \textit{Task} \rightarrow \textit{Time}$$
$$\text{time}(\langle i, t, \kappa, \rho \rangle) = t$$
$$\text{enqueue\_task} : \textit{TaskQueue} \times \textit{Task} \rightarrow \textit{TaskQueue}$$
$$\text{enqueue\_task}(Q, \tau') = \text{a reordering } Q' \text{ of } Q + \{\tau'\} \text{ such that}$$
$$\text{for any two tasks } \tau_1, \tau_2 \in Q', \text{time}(\tau_1) < \text{time}(\tau_2) \implies \tau_1 \text{ comes before } \tau_2 \text{ in } Q'.$$

We define three big-step evaluation functions: $\twoheadrightarrow_s$, for evaluating sequences of statements, $\twoheadrightarrow_e$, for evaluating expressions, and $\cdot\twoheadrightarrow_c$, for evaluating continuation applications.

$$\twoheadrightarrow_s: \textit{stmt}^* \times \textit{Taskid} \times \textit{TaskQueue} \times \textit{Store} \times \textit{Env} \times \textit{Cont} \rightarrow \textit{TaskQueue} \times \textit{Store}$$
$$\twoheadrightarrow_e: \textit{expr} \times \textit{Taskid} \times \textit{TaskQueue} \times \textit{Store} \times \textit{Env} \times \textit{Cont} \rightarrow \textit{TaskQueue} \times \textit{Store}$$
$$\cdot \twoheadrightarrow_c: \textit{Cont} \times \textit{Value} \times \textit{Taskid} \times \textit{TaskQueue} \times \textit{Store} \times \textit{Env} \rightarrow \textit{TaskQueue} \times \textit{Store}$$

Above these, we have a small-step evaluation function that runs tasks:

$$\rightarrow_t: \textit{TaskQueue} \times \textit{Store} \rightarrow \textit{TaskQueue} \times \textit{Store}$$

They are defined by the following rules:

[Apply Halt]
$$\langle halt \cdot v, i, Q, \sigma, \rho\rangle \twoheadrightarrow_c \langle Q, \sigma\rangle$$

[Eval Null]
$$\frac{\langle \kappa \cdot \diamond, i, Q, \sigma, \rho\rangle \twoheadrightarrow_c \langle Q', \sigma'\rangle}{\langle ;, i, Q, \sigma, \rho, \kappa\rangle \twoheadrightarrow_s \langle Q', \sigma'\rangle}$$

[Eval Stmts]
$$\frac{\langle S_1, i, Q, \sigma, \rho, \langle [\ ]\ S_2 \ldots, \kappa\rangle\rangle \twoheadrightarrow_s \langle Q', \sigma'\rangle}{\langle S_1 S_2 \ldots, i, Q, \sigma, \rho, \kappa\rangle \twoheadrightarrow_s \langle Q', \sigma'\rangle}$$

[Apply Stmts]
$$\frac{v \neq \diamond \qquad \langle \kappa \cdot v, i, Q, \sigma, \rho\rangle \twoheadrightarrow_c \langle Q', \sigma'\rangle}{\langle\langle [\ ]\ S \ldots, \kappa\rangle \cdot v, i, Q, \sigma, \rho\rangle \twoheadrightarrow_c \langle Q', \sigma'\rangle}$$
$$\frac{\langle S \ldots, i, Q, \sigma, \rho, \kappa\rangle \twoheadrightarrow_s \langle Q', \sigma'\rangle}{\langle\langle [\ ]\ S \ldots, \kappa\rangle \cdot \diamond, i, Q, \sigma, \rho\rangle \twoheadrightarrow_c \langle Q', \sigma'\rangle}$$

[Eval Expr]
$$\frac{\langle M, i, Q, \sigma, \rho, \langle \diamond, \kappa\rangle\rangle \twoheadrightarrow_e \langle Q', \sigma'\rangle}{\langle M;, i, Q, \sigma, \rho, \kappa\rangle \twoheadrightarrow_e \langle Q', \sigma'\rangle}$$

[Apply Expr]
$$\frac{\langle \kappa \cdot \diamond, i, Q, \sigma, \rho\rangle \twoheadrightarrow_c \langle Q', \sigma'\rangle}{\langle\langle \diamond, \kappa\rangle \cdot v, i, Q, \sigma, \rho\rangle \twoheadrightarrow_c \langle Q', \sigma'\rangle}$$

[Eval Return]
$$\frac{\langle M, i, Q, \sigma, \rho, \kappa\rangle \twoheadrightarrow_e \langle Q', \sigma'\rangle}{\langle \text{return } M;, i, Q, \sigma, \rho, \kappa\rangle \twoheadrightarrow_s \langle Q', \sigma'\rangle}$$

[Eval If]
$$\frac{\langle M, i, Q, \sigma, \rho, \langle \text{if } ([\ ])\ S_{then} \ldots \text{ else } S_{else} \ldots \text{ endif}, \kappa\rangle\rangle \twoheadrightarrow_e \langle Q', \sigma'\rangle}{\langle \text{if } (M)\ S_{then} \ldots \text{ else } S_{else} \ldots \text{ endif}, i, Q, \sigma, \rho, \kappa\rangle \twoheadrightarrow_s \langle Q', \sigma'\rangle}$$

[Apply If]
$$\frac{v \in \mathit{True} \qquad \langle S_{then} \ldots, i, Q, \sigma, \rho, \kappa\rangle \twoheadrightarrow_e \langle Q', \sigma'\rangle}{\langle\langle \text{if } ([\ ])\ S_{then} \ldots \text{ else } S_{else} \ldots \text{ endif}, \kappa\rangle \cdot v, i, Q, \sigma, \rho\rangle \twoheadrightarrow_c \langle Q', \sigma'\rangle}$$
$$\frac{v \in \mathit{False} \qquad \langle S_{else} \ldots, i, Q, \sigma, \rho, \kappa\rangle \twoheadrightarrow_e \langle Q', \sigma'\rangle}{\langle\langle \text{if } ([\ ])\ S_{then} \ldots \text{ else } S_{else} \ldots \text{ endif}, \kappa\rangle \cdot v, i, Q, \sigma, \rho\rangle \twoheadrightarrow_c \langle Q', \sigma'\rangle}$$

[Eval While]
$$\frac{\langle M, i, Q, \sigma, \rho, \langle \text{if } ([\ ])\ S \ldots \text{while } (M)\ S \ldots \text{ endwhile else endif}, \kappa\rangle\rangle \twoheadrightarrow_e \langle Q', \sigma'\rangle}{\langle \text{while } (M)\ S \ldots \text{ endwhile}, i, Q, \sigma, \rho, \kappa\rangle \twoheadrightarrow_s \langle Q', \sigma'\rangle}$$

[Eval Const]
$$\frac{\langle \kappa \cdot c, i, Q, \sigma, \rho \rangle \twoheadrightarrow_c \langle Q', \sigma' \rangle}{\langle c, i, Q, \sigma, \rho, \kappa \rangle \twoheadrightarrow_e \langle Q', \sigma' \rangle}$$

[Eval Var]
$$\frac{\langle \kappa \cdot \text{lookup\_var}(x, \rho), i, Q, \sigma, \rho \rangle \twoheadrightarrow_c \langle Q', \sigma' \rangle}{\langle x, i, Q, \sigma, \rho, \kappa \rangle \twoheadrightarrow_e \langle Q', \sigma' \rangle}$$

[Eval Var Assign]
$$\frac{\langle M, i, Q, \sigma, \rho, \langle x = [\ ], \kappa \rangle \rangle \twoheadrightarrow_e \langle Q', \sigma' \rangle}{\langle x = M, i, Q, \sigma, \rho, \kappa \rangle \twoheadrightarrow_e \langle Q', \sigma' \rangle}$$

[Apply Var Assign]
$$\frac{\begin{array}{c}\rho = \langle \pi, \theta, \mu, \beta \rangle \qquad \rho' = \langle \pi, \theta, \mu, \beta[x \mapsto v] \rangle \\ \langle \kappa \cdot v, i, Q, \sigma, \rho' \rangle \twoheadrightarrow_c \langle Q', \sigma' \rangle\end{array}}{\langle \langle x = [\ ], \kappa \rangle \cdot v, i, Q, \sigma, \rho \rangle \twoheadrightarrow_c \langle Q', \sigma' \rangle}$$

[Eval Field]
$$\frac{\langle M.(\text{quote}(f)), i, Q, \sigma, \rho, \kappa \rangle \twoheadrightarrow_e \langle Q', \sigma' \rangle}{\langle M.f, i, Q, \sigma, \rho, \kappa \rangle \twoheadrightarrow_e \langle Q', \sigma' \rangle}$$

[Eval Field Expr]
$$\frac{\langle M, i, Q, \sigma, \rho, \langle [\ ].(N), \kappa \rangle \rangle \twoheadrightarrow_e \langle Q', \sigma' \rangle}{\langle M.(N), i, Q, \sigma, \rho, \kappa \rangle \twoheadrightarrow_e \langle Q', \sigma' \rangle}$$

[Apply Field Expr]
$$\frac{v \notin \mathit{Objref} \qquad \langle \kappa \cdot \text{E\_TYPE}, i, Q, \sigma, \rho \rangle \twoheadrightarrow_c \langle Q', \sigma' \rangle}{\langle \langle [\ ].(M), \kappa \rangle \cdot v, i, Q, \sigma, \rho \rangle \twoheadrightarrow_c \langle Q', \sigma' \rangle}$$

$$\frac{o \in \mathit{Objref} \qquad \neg\,\text{valid}(o, \sigma) \qquad \langle \kappa \cdot \text{E\_INVIND}, i, Q, \sigma, \rho \rangle \twoheadrightarrow_c \langle Q', \sigma' \rangle}{\langle \langle [\ ].(M), \kappa \rangle \cdot o, i, Q, \sigma, \rho \rangle \twoheadrightarrow_c \langle Q', \sigma' \rangle}$$

$$\frac{\text{valid}(o, \sigma) \qquad \langle M, i, Q, \sigma, \rho, \langle o.([\ ]), \kappa \rangle \rangle \twoheadrightarrow_e \langle Q', \sigma' \rangle}{\langle \langle [\ ].(M), \kappa \rangle \cdot o, i, Q, \sigma, \rho \rangle \twoheadrightarrow_c \langle Q', \sigma' \rangle}$$

$$\frac{v \notin \mathit{String} \qquad \langle \kappa \cdot \text{E\_TYPE}, i, Q, \sigma, \rho \rangle \twoheadrightarrow_c \langle Q', \sigma' \rangle}{\langle \langle o.[\ ], \kappa \rangle \cdot v, i, Q, \sigma, \rho \rangle \twoheadrightarrow_c \langle Q', \sigma' \rangle}$$

$$\frac{\begin{array}{c}f \in \mathit{String} \qquad \text{lookup\_field}(\sigma(o), f, \sigma) = \langle \pi, \phi, v \rangle \qquad \text{can\_read}(\text{perm}(\rho), \pi, \phi, \sigma) \\ \langle \kappa \cdot v, i, Q, \sigma, \rho \rangle \twoheadrightarrow_c \langle Q', \sigma' \rangle\end{array}}{\langle \langle o.[\ ], \kappa \rangle \cdot f, i, Q, \sigma, \rho \rangle \twoheadrightarrow_c \langle Q', \sigma' \rangle}$$

$$\frac{\begin{array}{c}f \in \mathit{String} \qquad \text{lookup\_field}(\sigma(o), f, \sigma) = \langle \pi, \phi, v \rangle \qquad \neg\,\text{can\_read}(\text{perm}(\rho), \pi, \phi, \sigma) \\ \langle \kappa \cdot \text{E\_PERM}, i, Q, \sigma, \rho \rangle \twoheadrightarrow_c \langle Q', \sigma' \rangle\end{array}}{\langle \langle o.[\ ], \kappa \rangle \cdot f, i, Q, \sigma, \rho \rangle \twoheadrightarrow_c \langle Q', \sigma' \rangle}$$

$$\frac{\begin{array}{c}f \in \mathit{String} \qquad \text{lookup\_field}(\sigma(o), f, \sigma) = \text{E\_PROPNF} \\ \langle \kappa \cdot \text{E\_PROPNF}, i, Q, \sigma, \rho \rangle \twoheadrightarrow_c \langle Q', \sigma' \rangle\end{array}}{\langle \langle o.[\ ], \kappa \rangle \cdot f, i, Q, \sigma, \rho \rangle \twoheadrightarrow_c \langle Q', \sigma' \rangle}$$

[Eval Field Assign]

$$\frac{\langle M.(\mathrm{quote}(f)) = N, i, Q, \sigma, \rho, \kappa\rangle \twoheadrightarrow_e \langle Q', \sigma'\rangle}{\langle M.f = N, i, Q, \sigma, \rho, \kappa\rangle \twoheadrightarrow_e \langle Q', \sigma'\rangle}$$

[Eval Field Expr Assign]

$$\frac{\langle M, i, Q, \sigma, \rho, \langle [\ ].(N) = P, \kappa\rangle\rangle \twoheadrightarrow_e \langle Q', \sigma'\rangle}{\langle M.(N) = P, i, Q, \sigma, \rho, \kappa\rangle \twoheadrightarrow_e \langle Q', \sigma'\rangle}$$

[Apply Field Expr Assign]

$$\frac{v \notin \mathit{Objref} \qquad \langle \kappa \cdot \mathrm{E\_TYPE}, i, Q, \sigma, \rho\rangle \twoheadrightarrow_c \langle Q', \sigma'\rangle}{\langle\langle [\ ].(M) = N, \kappa\rangle \cdot v, i, Q, \sigma, \rho\rangle \twoheadrightarrow_c \langle Q', \sigma'\rangle}$$

$$\frac{o \in \mathit{Objref} \qquad \neg\,\mathrm{valid}(o, \sigma) \qquad \langle \kappa \cdot \mathrm{E\_INVIND}, i, Q, \sigma, \rho\rangle \twoheadrightarrow_c \langle Q', \sigma'\rangle}{\langle\langle [\ ].(M) = N, \kappa\rangle \cdot o, i, Q, \sigma, \rho\rangle \twoheadrightarrow_c \langle Q', \sigma'\rangle}$$

$$\frac{\mathrm{valid}(o, \sigma) \qquad \langle M, i, Q, \sigma, \rho, \langle o.([\ ]) = N, \kappa\rangle\rangle \twoheadrightarrow_e \langle Q', \sigma'\rangle}{\langle\langle [\ ].(M) = N, \kappa\rangle \cdot o, i, Q, \sigma, \rho\rangle \twoheadrightarrow_c \langle Q', \sigma'\rangle}$$

$$\frac{v \notin \mathit{String} \qquad \langle \kappa \cdot \mathrm{E\_TYPE}, i, Q, \sigma, \rho\rangle \twoheadrightarrow_c \langle Q', \sigma'\rangle}{\langle\langle o.([\ ]) = M, \kappa\rangle \cdot v, i, Q, \sigma, \rho\rangle \twoheadrightarrow_c \langle Q', \sigma'\rangle}$$

$$\frac{f \in \mathit{String} \qquad \langle M, i, Q, \sigma, \rho, \langle o.(f) = [\ ], \kappa\rangle\rangle \twoheadrightarrow_e \langle Q', \sigma'\rangle}{\langle\langle o.([\ ]) = M, \kappa\rangle \cdot f, i, Q, \sigma, \rho\rangle \twoheadrightarrow_c \langle Q', \sigma'\rangle}$$

$$\frac{\begin{array}{c}\sigma(o) = \langle \pi, \phi, p, F, M\rangle \\ \mathrm{lookup\_field}(\langle \pi, \phi, p, F, M\rangle, f, \sigma) = \langle \pi', \phi', v\rangle \qquad \mathrm{can\_write}(\mathrm{perm}(\rho), \pi', \phi', \sigma) \\ F' = F[f \mapsto \langle \pi', \phi', v'\rangle] \qquad \sigma' = \sigma[o \mapsto \langle \pi, \phi, p, F', M\rangle] \\ \langle \kappa \cdot v', i, Q, \sigma', \rho\rangle \twoheadrightarrow_c \langle Q', \sigma''\rangle\end{array}}{\langle\langle o.(f) = [\ ], \kappa\rangle \cdot v', i, Q, \sigma, \rho\rangle \twoheadrightarrow_c \langle Q', \sigma''\rangle}$$

$$\frac{\begin{array}{c}\mathrm{lookup\_field}(\sigma(o), f, \sigma) = \langle \pi, \phi, v\rangle \qquad \neg\,\mathrm{can\_write}(\mathrm{perm}(\rho), \pi, \phi, \sigma) \\ \langle \kappa \cdot \mathrm{E\_PERM}, i, Q, \sigma, \rho\rangle \twoheadrightarrow_c \langle Q', \sigma'\rangle\end{array}}{\langle\langle o.(f) = [\ ], \kappa\rangle \cdot v', i, Q, \sigma, \rho\rangle \twoheadrightarrow_c \langle Q', \sigma'\rangle}$$

$$\frac{\begin{array}{c}\mathrm{lookup\_field}(\sigma(o), f, \sigma) = \mathrm{E\_PROPNF} \\ \langle \kappa \cdot \mathrm{E\_PROPNF}, i, Q, \sigma, \rho\rangle \twoheadrightarrow_c \langle Q', \sigma'\rangle\end{array}}{\langle\langle o.(f) = [\ ], \kappa\rangle \cdot v, i, Q, \sigma, \rho\rangle \twoheadrightarrow_c \langle Q', \sigma'\rangle}$$

[Eval Method]

$$\frac{\langle M\!:\!(\mathrm{quote}(\mu))(\mathit{arg}_1,\ldots,\mathit{arg}_n), i, Q, \sigma, \rho, \kappa\rangle \twoheadrightarrow_e \langle Q', \sigma'\rangle}{\langle M\!:\!\mu(\mathit{arg}_1,\ldots,\mathit{arg}_n), i, Q, \sigma, \rho, \kappa\rangle \twoheadrightarrow_e \langle Q', \sigma'\rangle}$$

[Eval Method Expr]

$$\frac{\langle M, i, Q, \sigma, \rho, \langle[\;]\!:\!(N)(\mathit{arg}_1,\ldots,\mathit{arg}_n), \kappa\rangle\rangle \twoheadrightarrow_e \langle Q', \sigma'\rangle}{\langle M\!:\!(N)(\mathit{arg}_1,\ldots,\mathit{arg}_n), i, Q, \sigma, \rho, \kappa\rangle \twoheadrightarrow_e \langle Q', \sigma'\rangle}$$

[Apply Method Expr]

$$\frac{v \notin \mathit{Objref} \qquad \langle \kappa \cdot \mathrm{E\_TYPE}, i, Q, \sigma, \rho\rangle \twoheadrightarrow_c \langle Q', \sigma'\rangle}{\langle\langle[\;]\!:\!(M)(\mathit{arg}_1,\ldots,\mathit{arg}_n), \kappa\rangle \cdot v, i, Q, \sigma, \rho\rangle \twoheadrightarrow_c \langle Q', \sigma'\rangle}$$

$$\frac{o \in \mathit{Objref} \qquad \neg\,\mathrm{valid}(o,\sigma) \qquad \langle \kappa \cdot \mathrm{E\_INVIND}, i, Q, \sigma, \rho\rangle \twoheadrightarrow_c \langle Q', \sigma'\rangle}{\langle\langle[\;]\!:\!(M)(\mathit{arg}_1,\ldots,\mathit{arg}_n), \kappa\rangle \cdot o, i, Q, \sigma, \rho\rangle \twoheadrightarrow_c \langle Q', \sigma'\rangle}$$

$$\frac{\mathrm{valid}(o,\sigma) \qquad \langle M, i, Q, \sigma, \rho, \langle o\!:\!([\;])(\mathit{arg}_1,\ldots,\mathit{arg}_n), \kappa\rangle\rangle \twoheadrightarrow_e \langle Q', \sigma'\rangle}{\langle\langle[\;]\!:\!(M)(\mathit{arg}_1,\ldots,\mathit{arg}_n), \kappa\rangle \cdot o, i, Q, \sigma, \rho\rangle \twoheadrightarrow_c \langle Q', \sigma'\rangle}$$

$$\frac{v \notin \mathit{String} \qquad \langle \kappa \cdot \mathrm{E\_TYPE}, i, Q, \sigma, \rho\rangle \twoheadrightarrow_c \langle Q', \sigma'\rangle}{\langle\langle o\!:\!([\;])(\mathit{arg}_1,\ldots,\mathit{arg}_n), \kappa\rangle \cdot v, i, Q, \sigma, \rho\rangle \twoheadrightarrow_c \langle Q', \sigma'\rangle}$$

$$\frac{\mu \in \mathit{String} \qquad \langle \mathit{arg}_1, i, Q, \sigma, \rho, \langle o\!:\!(\mu)([\;], \mathit{arg}_2,\ldots,\mathit{arg}_n), \kappa\rangle\rangle \twoheadrightarrow_e \langle Q', \sigma'\rangle}{\langle\langle o\!:\!([\;])(\mathit{arg}_1,\ldots,\mathit{arg}_n), \kappa\rangle \cdot \mu, i, Q, \sigma, \rho\rangle \twoheadrightarrow_c \langle Q', \sigma'\rangle}$$

$$\frac{i < n \qquad \langle \mathit{arg}_{i+1}, i, Q, \sigma, \rho, \langle o\!:\!(\mu)(v_1,\ldots,v_i,[\;],\mathit{arg}_{i+2},\ldots,\mathit{arg}_n), \kappa\rangle\rangle \twoheadrightarrow_e \langle Q', \sigma'\rangle}{\langle\langle o\!:\!(\mu)(v_1,\ldots,v_{i-1},[\;],\mathit{arg}_{i+1},\ldots,\mathit{arg}_n), \kappa\rangle \cdot v_i, i, Q, \sigma, \rho\rangle \twoheadrightarrow_c \langle Q', \sigma'\rangle}$$

$$\frac{\begin{array}{c}\mathrm{lookup\_method}(\sigma(o),\mu,\sigma) = \langle \pi, \phi, \mathit{prog}\rangle \\ \beta = [\mathrm{this} \mapsto o, \mathrm{caller} \mapsto \mathrm{this}(\rho), \mathrm{verb} \mapsto \mu, \mathrm{args} \mapsto \langle v_1,\ldots,v_n\rangle] \\ \rho' = \langle \pi, o, \mu, \beta\rangle \\ \langle \mathit{prog}, i, Q, \sigma, \rho', \langle[\;]\rho, \kappa\rangle\rangle \twoheadrightarrow_s \langle Q', \sigma'\rangle\end{array}}{\langle\langle o\!:\!(\mu)(v_1,\ldots,v_{n-1},[\;]), \kappa\rangle \cdot v_n, i, Q, \sigma, \rho\rangle \twoheadrightarrow_c \langle Q', \sigma'\rangle}$$

$$\frac{\begin{array}{c}\mathrm{lookup\_method}(\sigma(o),\mu,\sigma) = \mathrm{E\_VERBNF} \\ \langle \kappa \cdot \mathrm{E\_VERBNF}, i, Q, \sigma, \rho\rangle \twoheadrightarrow_c \langle Q', \sigma'\rangle\end{array}}{\langle\langle o\!:\!(\mu)(v_1,\ldots,v_{n-1},[\;]), \kappa\rangle \cdot v_n, i, Q, \sigma, \rho\rangle \twoheadrightarrow_c \langle Q', \sigma'\rangle}$$

$$\frac{v \neq \diamond \qquad \langle \kappa \cdot v, i, Q, \sigma, \rho\rangle \twoheadrightarrow_c \langle Q', \sigma'\rangle}{\langle\langle[\;]\rho, \kappa\rangle \cdot v, i, Q, \sigma, \rho'\rangle \twoheadrightarrow_c \langle Q', \sigma'\rangle}$$

$$\frac{\langle \kappa \cdot 0, i, Q, \sigma, \rho\rangle \twoheadrightarrow_c \langle Q', \sigma'\rangle}{\langle\langle[\;]\rho, \kappa\rangle \cdot \diamond, i, Q, \sigma, \rho'\rangle \twoheadrightarrow_c \langle Q', \sigma'\rangle}$$

[Eval Pass]

$$\frac{\langle arg_1, i, Q, \sigma, \rho, \langle \mathrm{pass}([\,], arg_2, \ldots, arg_n), \kappa\rangle\rangle \twoheadrightarrow_e \langle Q', \sigma'\rangle}{\langle \mathrm{pass}(arg_1, \ldots, arg_n), i, Q, \sigma, \rho, \kappa\rangle \twoheadrightarrow_e \langle Q', \sigma'\rangle}$$

[Apply Pass]

$$\frac{i < n \qquad \langle arg_{i+1}, i, Q, \sigma, \rho, \langle \mathrm{pass}(v_1, \ldots, v_i, [\,], arg_{i+2}, \ldots, arg_n), \kappa\rangle\rangle \twoheadrightarrow_e \langle Q', \sigma'\rangle}{\langle\langle \mathrm{pass}(v_1, \ldots, v_{i-1}, [\,], arg_{i+1}, \ldots, arg_n), \kappa\rangle \cdot v_i, i, Q, \sigma, \rho\rangle \twoheadrightarrow_c \langle Q', \sigma'\rangle}$$

$$\frac{\begin{array}{c} \theta = \mathrm{this}(\rho) \qquad p = \mathrm{parent}(\sigma(\theta)) \qquad \mathrm{valid}(p, \sigma) \\ \mu = \mathrm{cur\_meth}(\rho) \qquad \mathrm{lookup\_method}(\sigma(p), \mu, \sigma) = \langle \pi, \phi, prog\rangle \\ \beta = [\mathrm{this} \mapsto \theta, \mathrm{caller} \mapsto \theta, \mathrm{verb} \mapsto \mu, \mathrm{args} \mapsto \langle v_1, \ldots, v_n\rangle] \\ \rho' = \langle \pi, \theta, \mu, \beta\rangle \\ \langle prog, i, Q, \sigma, \rho', \langle [\,]\rho, \kappa\rangle\rangle \twoheadrightarrow_s \langle Q', \sigma'\rangle \end{array}}{\langle\langle \mathrm{pass}(v_1, \ldots, v_{n-1}, [\,]), \kappa\rangle \cdot v_n, i, Q, \sigma, \rho\rangle \twoheadrightarrow_c \langle Q', \sigma'\rangle}$$

$$\frac{\begin{array}{c} p = \mathrm{parent}(\sigma(\theta)) \qquad \neg\, \mathrm{valid}(p, \sigma) \\ \langle \kappa \cdot \mathrm{E\_VERBNF}, i, Q, \sigma, \rho\rangle \twoheadrightarrow_c \langle Q', \sigma'\rangle \end{array}}{\langle\langle \mathrm{pass}(v_1, \ldots, v_{n-1}, [\,]), \kappa\rangle \cdot v_n, i, Q, \sigma, \rho\rangle \twoheadrightarrow_c \langle Q', \sigma'\rangle}$$

$$\frac{\begin{array}{c} p = \mathrm{parent}(\sigma(\theta)) \qquad \mathrm{valid}(p, \sigma) \\ \mathrm{lookup\_method}(\sigma(p), \mu, \sigma) = \mathrm{E\_VERBNF} \\ \langle \kappa \cdot \mathrm{E\_VERBNF}, i, Q, \sigma, \rho\rangle \twoheadrightarrow_c \langle Q', \sigma'\rangle \end{array}}{\langle\langle \mathrm{pass}(v_1, \ldots, v_{n-1}, [\,]), \kappa\rangle \cdot v_n, i, Q, \sigma, \rho\rangle \twoheadrightarrow_c \langle Q', \sigma'\rangle}$$

[Eval Create]

$$\frac{\langle M, i, Q, \sigma, \rho, \langle \mathrm{create}([\,]), \kappa\rangle\rangle \twoheadrightarrow_e \langle Q', \sigma'\rangle}{\langle \mathrm{create}(M,), i, Q, \sigma, \rho\rangle\kappa \twoheadrightarrow_e \langle Q', \sigma'\rangle}$$

[Apply Create]

$$\frac{\begin{array}{c} (\mathrm{valid}(o, \sigma) \wedge \mathrm{can\_create}(\mathrm{perm}(\rho), \mathrm{perm}(o), \sigma)) \vee o = \texttt{\#-1} \\ o' = \mathrm{fresh\_objid}(\sigma) \qquad \sigma' = \sigma[o' \mapsto \langle \mathrm{perm}(\rho), \emptyset, o, \emptyset, \emptyset\rangle] \\ \langle \kappa \cdot o', i, Q, \sigma', \rho\rangle \twoheadrightarrow_c \langle Q', \sigma''\rangle \end{array}}{\langle\langle \mathrm{create}([\,]), \kappa\rangle \cdot o, i, Q, \sigma, \rho\rangle \twoheadrightarrow_c \langle Q', \sigma''\rangle}$$

$$\frac{\begin{array}{c} o \in \mathit{Objref} \qquad \neg\, \mathrm{can\_create}(\mathrm{perm}(\rho), \mathrm{perm}(o), \sigma)) \qquad o \neq \texttt{\#-1} \\ \langle \kappa \cdot \mathrm{E\_PERM}, i, Q, \sigma, \rho\rangle \twoheadrightarrow_c \langle Q', \sigma'\rangle \end{array}}{\langle\langle \mathrm{create}([\,]), \kappa\rangle \cdot o, i, Q, \sigma, \rho\rangle \twoheadrightarrow_c \langle Q', \sigma'\rangle}$$

$$\frac{v \notin \mathit{Objref} \qquad \langle \kappa \cdot \mathrm{E\_PERM}, i, Q, \sigma, \rho\rangle \twoheadrightarrow_c \langle Q', \sigma'\rangle}{\langle\langle \mathrm{create}([\,]), \kappa\rangle \cdot v, i, Q, \sigma, \rho\rangle \twoheadrightarrow_c \langle Q', \sigma'\rangle}$$

[Eval Suspend]

$$\langle \text{suspend}(), i, Q, \sigma, \rho, \kappa\rangle \twoheadrightarrow_e \langle Q :: \langle i, \infty, \kappa, \rho\rangle, \sigma\rangle$$

[Eval Suspend Timed]

$$\frac{\langle M, i, Q, \sigma, \rho, \langle \text{suspend}([\ ]), \kappa\rangle\rangle \twoheadrightarrow_e \langle Q', \sigma'\rangle}{\langle \text{suspend}(M), i, Q, \sigma, \rho, \kappa\rangle \twoheadrightarrow_e \langle Q', \sigma'\rangle}$$

[Apply Suspend Timed]

$$\frac{v >= 0 \qquad t = \text{future}(v) \qquad \tau = \langle i, t, \kappa, \rho\rangle \qquad Q' = \text{enqueue\_task}(Q, \tau)}{\langle\langle \text{suspend}([\ ]), \kappa\rangle \cdot v, i, Q, \sigma, \rho\rangle \twoheadrightarrow_c \langle Q', \sigma\rangle}$$

$$\frac{v < 0 \qquad \langle \kappa \cdot \text{E\_INVARG}, i, Q, \sigma, \rho\rangle \twoheadrightarrow_c \langle Q', \sigma'\rangle}{\langle\langle \text{suspend}([\ ]), \kappa\rangle \cdot v, i, Q, \sigma, \rho\rangle \twoheadrightarrow_c \langle Q', \sigma'\rangle}$$

[Eval Taskid]

$$\frac{\langle \kappa \cdot i, i, Q, \sigma, \rho\rangle \twoheadrightarrow_c \langle Q', \sigma'\rangle}{\langle \text{taskid}(), i, Q, \sigma, \rho, \kappa\rangle \twoheadrightarrow_e \langle Q', \sigma'\rangle}$$

[Eval Set Task Perms]

$$\frac{\langle M, i, Q, \sigma, \rho, \langle \text{set\_task\_perms}([\ ]), \kappa\rangle\rangle \twoheadrightarrow_e \langle Q', \sigma'\rangle}{\langle \text{set\_task\_perms}(M), i, Q, \sigma, \rho, \kappa\rangle \twoheadrightarrow_e \langle Q', \sigma'\rangle}$$

[Apply Set Task Perms]

$$\frac{\begin{array}{c}\pi = \text{perm}(\rho) \qquad \text{controls}(\pi, \pi', \sigma) \\ \rho = \langle \pi, \theta, \mu, \beta\rangle \qquad \rho' = \langle \pi', \theta, \mu, \beta\rangle \\ \langle \kappa \cdot 0, i, Q, \sigma, \rho'\rangle \twoheadrightarrow_c \langle Q', \sigma'\rangle\end{array}}{\langle \text{set\_task\_perms}([\ ]) \cdot \pi', i, Q, \sigma, \rho\rangle \twoheadrightarrow_c \langle Q', \sigma'\rangle}$$

$$\frac{\begin{array}{c}\pi = \text{perm}(\rho) \qquad \neg\,\text{controls}(\pi, \pi', \sigma) \\ \langle \kappa \cdot \text{E\_PERM}, i, Q, \sigma, \rho\rangle \twoheadrightarrow_c \langle Q', \sigma'\rangle\end{array}}{\langle \text{set\_task\_perms}([\ ]) \cdot \pi', i, Q, \sigma, \rho\rangle \twoheadrightarrow_c \langle Q', \sigma'\rangle}$$

[Eval Resume]

$$\frac{\langle M, i, Q, \sigma, \rho, \langle \text{resume}([\ ], N), \kappa\rangle\rangle \twoheadrightarrow_e \langle Q', \sigma'\rangle}{\langle \text{resume}(M, N), i, Q, \sigma, \rho, \kappa\rangle \twoheadrightarrow_e \langle Q', \sigma'\rangle}$$

[Apply Resume]

$$\frac{\langle M, i', Q, \sigma, \rho, \langle \text{resume}(i, [\ ]), \kappa\rangle\rangle \twoheadrightarrow_e \langle Q', \sigma'\rangle}{\langle\langle \text{resume}([\ ], M), \kappa\rangle \cdot i, i', Q, \sigma, \rho\rangle \twoheadrightarrow_c \langle Q', \sigma'\rangle}$$

$$\frac{\begin{array}{c}\tau = \langle i, t, \kappa', \rho'\rangle \qquad \tau \in Q \qquad \text{controls}(\text{perm}(\rho), \text{perm}(\rho'), \sigma) \\ t' = \text{future}(0) \qquad \tau' = \langle i, t', \langle [\ ] v, \kappa'\rangle, \rho'\rangle \\ Q' = \text{enqueue\_task}(Q \setminus \tau, \tau') \qquad \langle \kappa \cdot 0, i', Q', \sigma, \rho\rangle \twoheadrightarrow_c \langle Q'', \sigma'\rangle\end{array}}{\langle\langle \text{resume}(i, [\ ]), \kappa\rangle \cdot v, i', Q, \sigma, \rho\rangle \twoheadrightarrow_c \langle Q'', \sigma'\rangle}$$

$$\frac{\begin{array}{c}\tau = \langle i, t, \kappa', \rho'\rangle \qquad \tau \in Q \qquad \neg\, \text{controls}(\text{perm}(\rho), \text{perm}(\rho'), \sigma) \\ \langle \kappa \cdot \text{E\_PERM}, i', Q, \sigma, \rho\rangle \twoheadrightarrow_c \langle Q', \sigma'\rangle\end{array}}{\langle\langle \text{resume}(i, [\ ]), \kappa\rangle \cdot v, i', Q, \sigma, \rho\rangle \twoheadrightarrow_c \langle Q', \sigma'\rangle}$$

$$\frac{\forall \tau \in Q, \text{taskid}(\tau) \neq i \qquad \langle \kappa \cdot \text{E\_INVARG}, i', Q, \sigma, \rho\rangle \twoheadrightarrow_c \langle Q', \sigma'\rangle}{\langle\langle \text{resume}(i, [\ ]), \kappa\rangle \cdot v, i', Q, \sigma, \rho\rangle \twoheadrightarrow_c \langle Q', \sigma'\rangle}$$

$$\frac{\langle \kappa \cdot v, i, Q, \sigma, \rho\rangle \twoheadrightarrow_c \langle Q', \sigma'\rangle}{\langle\langle [\ ] v, \kappa\rangle \cdot v', i, Q, \sigma, \rho\rangle \twoheadrightarrow_c \langle Q', \sigma'\rangle}$$

[Eval Fork]

$$\frac{\langle M, i, Q, \sigma, \rho, \langle \text{fork } x \ ([\ ]) \ S \ldots \ \text{endfork}, \kappa\rangle\rangle \twoheadrightarrow_e \langle Q', \sigma'\rangle}{\langle \text{fork } x \ (M) \ S \ldots \ \text{endfork}, i, Q, \sigma, \rho, \kappa\rangle \twoheadrightarrow_s \langle Q', \sigma'\rangle}$$

[Apply Fork]

$$\frac{\begin{array}{c}i = \text{fresh\_taskid}() \qquad t = \text{future}(v) \\ \rho = \langle \pi, \theta, \mu, \beta\rangle \qquad \rho' = \langle \pi, \theta, \mu, \beta[x \mapsto i]\rangle \\ \tau' = \langle i, t, \langle \diamond, \langle [\ ] S \ldots, \mathit{halt}\rangle\rangle, \rho'\rangle \\ Q' = \text{enqueue\_task}(Q, \tau') \qquad \langle \kappa \cdot \diamond, i', Q', \sigma, \rho'\rangle \twoheadrightarrow_c \langle Q'', \sigma'\rangle\end{array}}{\langle\langle \text{fork } x \ ([\ ]) \ S \ldots \ \text{endfork}, \kappa\rangle \cdot v, i', Q, \sigma, \rho\rangle \twoheadrightarrow_c \langle Q'', \sigma'\rangle}$$

[Eval Kill Task]

$$\frac{\langle M, i, Q, \sigma, \rho, \langle \text{kill\_task}([\;]), \kappa \rangle\rangle \twoheadrightarrow_c \langle Q', \sigma' \rangle}{\langle \text{kill\_task}(M), i, Q, \sigma, \rho, \kappa \rangle \twoheadrightarrow_e \langle Q', \sigma' \rangle}$$

[Apply Kill Task]

$$\frac{\begin{array}{c}\tau = \langle i, t, \kappa', \rho' \rangle \qquad \tau \in Q \qquad \text{controls}(\text{perm}(\rho), \text{perm}(\rho'), \sigma) \\ Q' = Q \setminus \tau \qquad \langle \kappa \cdot 0, i', Q', \sigma, \rho \rangle \twoheadrightarrow_c \langle Q'', \sigma' \rangle\end{array}}{\langle \langle \text{kill\_task}([\;]), \kappa \rangle \cdot i, i', Q, \sigma, \rho \rangle \twoheadrightarrow_c \langle Q'', \sigma' \rangle}$$

$$\frac{\begin{array}{c}\tau = \langle i, t, \kappa', \rho' \rangle \qquad \tau \in Q \qquad \neg\, \text{controls}(\text{perm}(\rho), \text{perm}(\rho'), \sigma) \\ \langle \kappa \cdot \text{E\_PERM}, i', Q, \sigma, \rho \rangle \twoheadrightarrow_c \langle Q', \sigma' \rangle\end{array}}{\langle \langle \text{kill\_task}([\;]), \kappa \rangle \cdot i, i', Q, \sigma, \rho \rangle \twoheadrightarrow_c \langle Q', \sigma' \rangle}$$

$$\frac{\begin{array}{c}\forall \tau \in Q, \text{taskid}(\tau) \neq i \qquad i \neq i' \\ \langle \kappa \cdot \text{E\_INVARG}, i', Q, \sigma, \rho \rangle \twoheadrightarrow_c \langle Q', \sigma' \rangle\end{array}}{\langle \langle \text{kill\_task}([\;]), \kappa \rangle \cdot i, i', Q, \sigma, \rho \rangle \twoheadrightarrow_c \langle Q', \sigma' \rangle}$$

[Apply Kill Self]

$$\langle \langle \text{kill\_task}([\;]), \kappa \rangle \cdot i, i, Q, \sigma, \rho \rangle \twoheadrightarrow_c \langle Q, \sigma \rangle$$

[Task Run]

$$\frac{t <= \text{future}(0) \qquad \langle \kappa \cdot 0, i, Q, \sigma, \rho \rangle \twoheadrightarrow_c \langle Q', \sigma' \rangle}{\langle \langle i, t, \kappa, \rho \rangle :: Q, \sigma \rangle \rightarrow_t \langle Q', \sigma' \rangle}$$