Temporal Higher-Order Contracts

Tim Disney, Cormac Flanagan, **Jay McCarthy**

This material is based upon work supported by the National Science Foundation under Grants 1016334 and 0905650.

sort : fun

sort : fun

4, true,
(list 1 2 3)

sort : fun

4, true,
(list 1 2 3)

quicksort,
insertionsort,
+, modulo

sort :

list(Int)

fun

→

list(Int)

sort :

list(Int)

fun

→

list(Int)

+, modulo

sort :

list(Int)

fun

→

list(Int)

+, modulo

quicksort,
insertionsort,
filter

$$\text{sort} : \text{list}(\text{Int})\ (\text{Int}\ \text{Int} \rightarrow \text{Bool}) \rightarrow \text{list}(\text{Int})$$

sort :
list(Int)
(Int Int → Bool)
→
list(Int)

filter, map

sort :

list(Int)

(Int Int → Bool)

→

list(Int)

filter, map

quicksort,
insertionsort

sort :

list(Int)

(Int Int → Bool)

→

list(Int)

[not re-entrant]

```
(define (cmp x y)
 (f x y
   (sort m <=)))

(sort l cmp)
```

sort :

list(Int)

(cmp : Int Int → Bool)

→

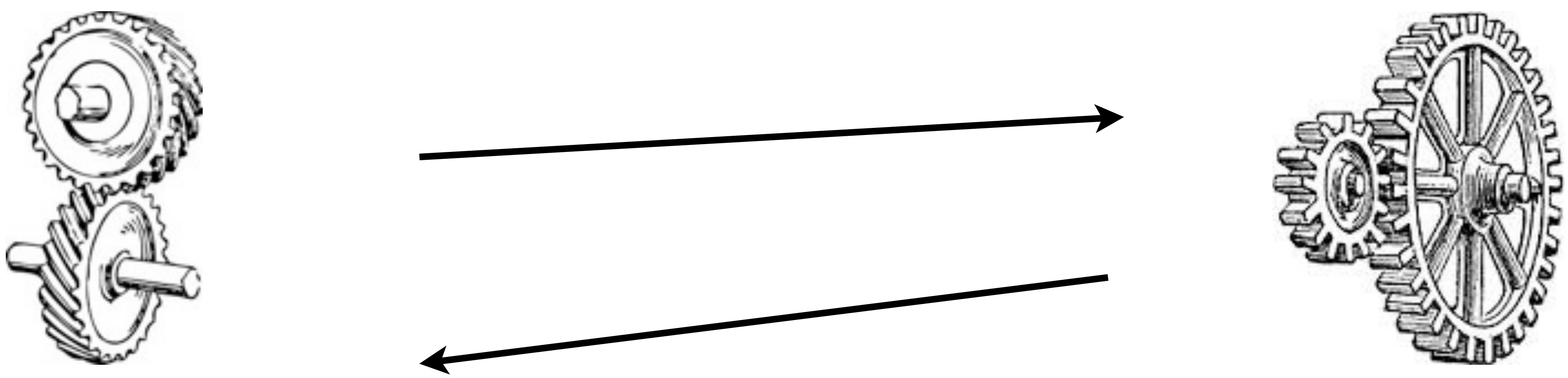
list(Int)

[not re-entrant, cmp is a capability]

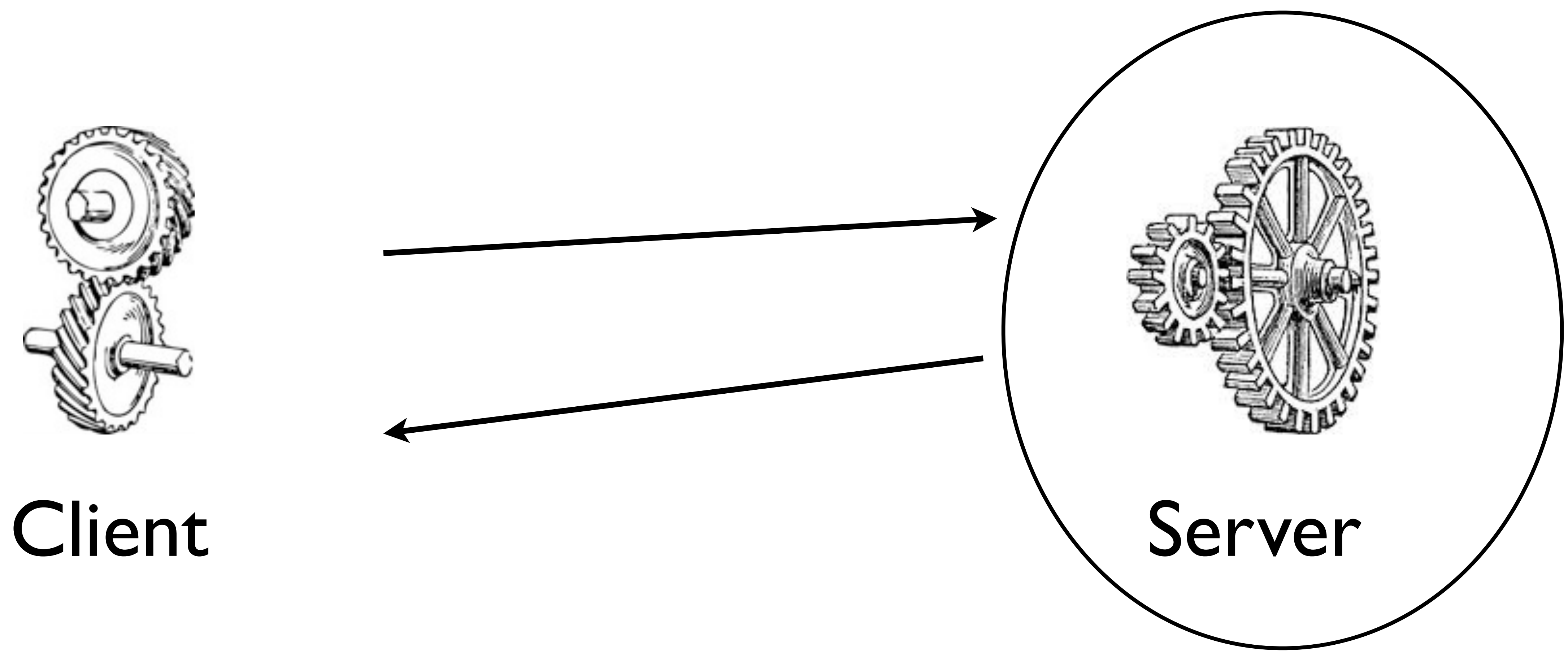
```
(define last-cmp #f)
(define (partner l cmp)
  (set! last-cmp cmp)
  (quicksort l cmp))
(define (thief x y)
  (last-cmp x y))
```

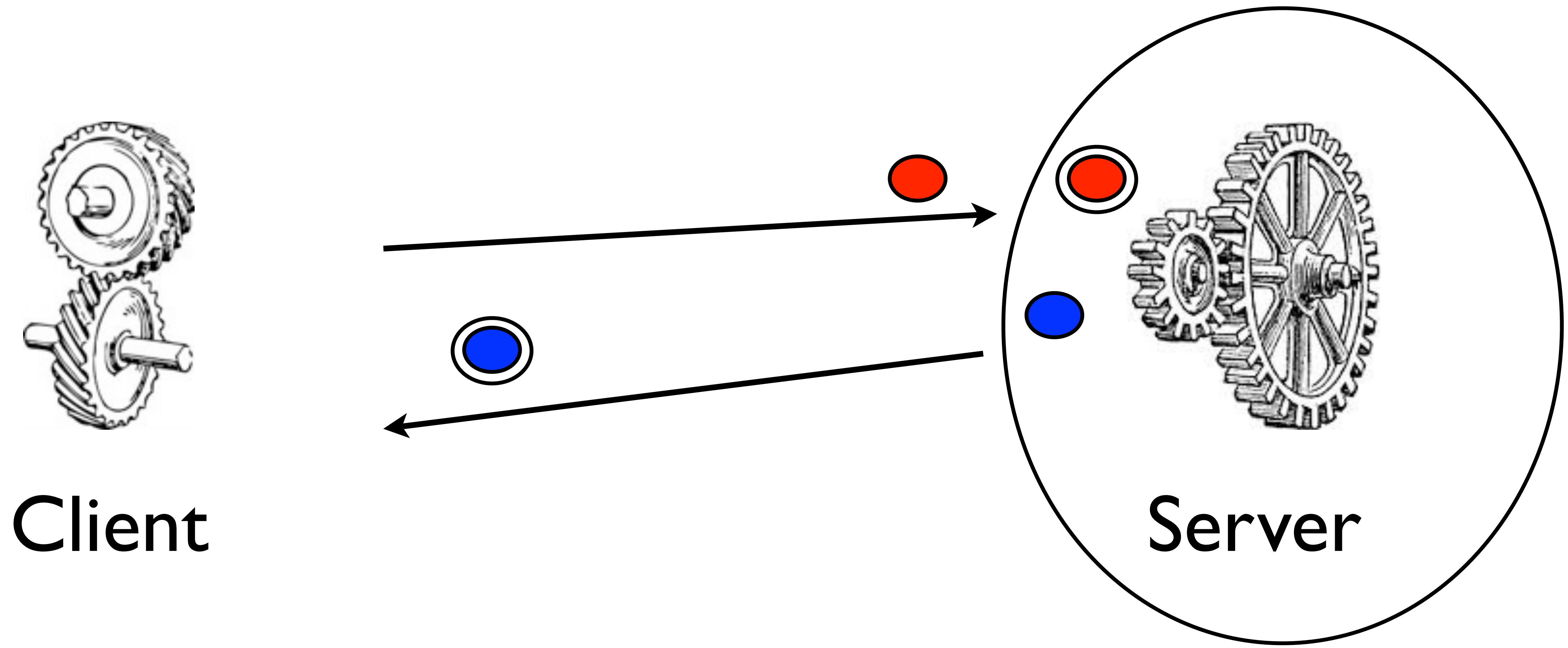
- first-order preconditions (Eiffel, etc)
- higher-order contracts (Racket, etc)
- first-order temporal contracts (MOP, etc)
- **higher-order temporal contracts**

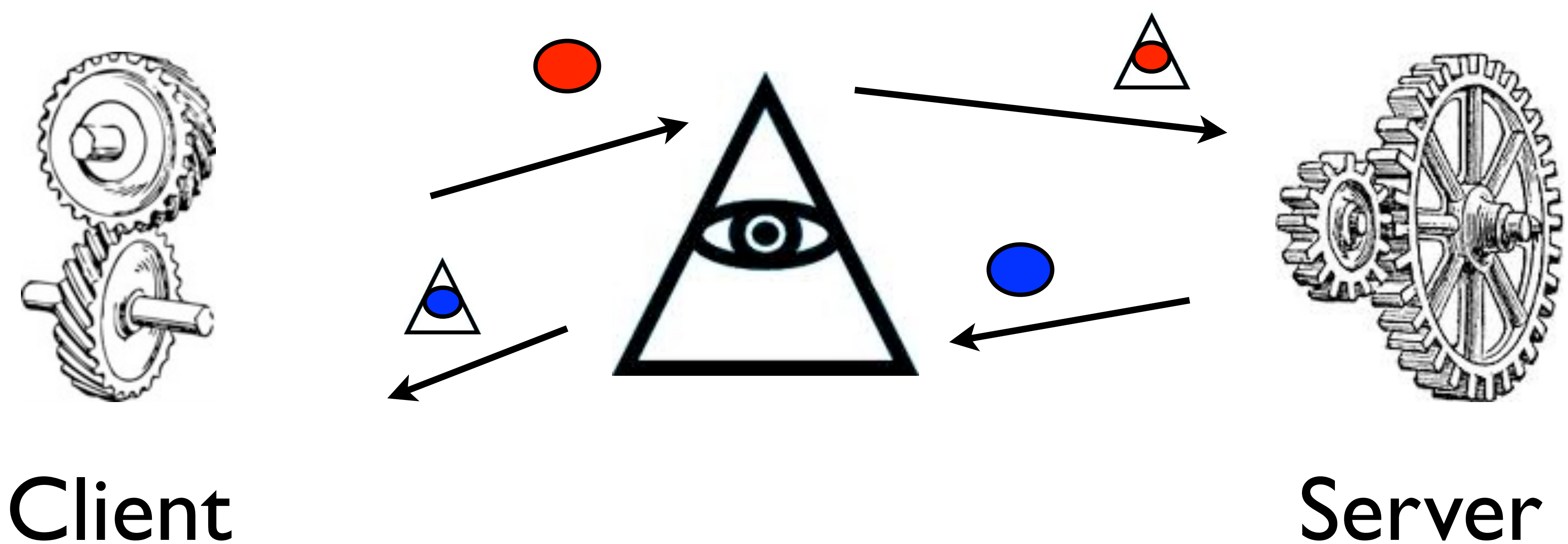
```
SortContract =
 sort : (List Pos)
        (cmp : Pos→Pos→Bool)
      → (List Pos)
  where
   not ... call(sort,_) !ret(sort,_)*
       call(sort,_)
  and
   not ... ret(sort,_)  ... call(cmp,_)
```



Client Server


Client
Server


Client
Server



Client
Server

Step 1: Game Semantics

Figure 1: CSI Machine

Domains

$$\begin{array}{llll}
\mathit{State} & \mathit{CSI} & \in & \mathit{Code} \times \mathit{Store} \times \mathit{Interface} \\
 & & & \\
\mathit{Code} & C & ::= & \mathcal{E}[e] \mid \mathcal{Q}[\bot] \\
\mathit{Store} & S & \in & \mathcal{P}(\mathit{Var} \times \mathit{Var} \times \mathit{Value}) \\
\mathit{Interface} & I & \in & \mathit{Var} \rightarrow \mathit{Value}
\end{array}$$

$$\begin{array}{llll}
\mathit{Evaluation\ context} & \mathcal{E} & ::= & \mathcal{E}[\bullet\ e] \mid \mathcal{E}[v\ \bullet] \mid \mathcal{E}[\texttt{ref}\ \bullet] \mid \mathcal{Q}[\texttt{rcv.call}_x\ \bullet] \\
\mathit{Quiescent\ context} & \mathcal{Q} & ::= & \bullet \mid \mathcal{E}[\texttt{send.call}_x\ \bullet] \\
\mathit{Value} & v & ::= & c \mid x \mid \lambda y.\, e \\
\mathit{Handle} & h & ::= & c \mid x \\
\mathit{Event} & a & ::= & \rho.\texttt{ret}(x, h) \mid \rho.\texttt{call}(x, h) \\
\mathit{Direction} & \rho & ::= & \texttt{send} \mid \texttt{rcv} \\
\mathit{Trace} & t & ::= & \vec{a}
\end{array}$$

Transition relation $(\rightarrow) \subseteq \mathit{State} \times \mathit{Event}_\perp \times \mathit{State}$

$$\begin{array}{lclll}
\langle\ \mathcal{E}[(\lambda x.\, e)\ v],\ S,\ I\ \rangle & \rightarrow & \langle\ \mathcal{E}[e[x := v]],\ S,\ I\ \rangle & & [\text{CALL}] \\
\langle\ \mathcal{E}[c\ v],\ S,\ I\ \rangle & \rightarrow & \langle\ \mathcal{E}[v'],\ S,\ I\ \rangle & v' = \delta(c, v) & [\text{PRIM}] \\
\langle\ \mathcal{E}[\texttt{ref}\ v],\ S,\ I\ \rangle & \rightarrow & \langle\ \mathcal{E}[\texttt{pair}\ x\ y],\ S[(x,y) \mapsto v],\ I\ \rangle & x, y \text{ fresh} & [\text{REF}] \\
\langle\ \mathcal{E}[x\ v],\ S[(x,y) \mapsto v'],\ I\ \rangle & \rightarrow & \langle\ \mathcal{E}[v'],\ S[(x,y) \mapsto v'],\ I\ \rangle & & [\text{GET}] \\
\langle\ \mathcal{E}[y\ v],\ S[(x,y) \mapsto v'],\ I\ \rangle & \rightarrow & \langle\ \mathcal{E}[v],\ S[(x,y) \mapsto v],\ I\ \rangle & & [\text{SET}] \\
 & & & & \\
\langle\ \mathcal{E}[x\ v],\ S,\ I\ \rangle & \rightarrow^{\texttt{send.call}(x,h)} & \langle\ \mathcal{E}[\texttt{send.call}_x\ \bot],\ S,\ I[h \triangleright v]\ \rangle & x \notin BV(S) & [\text{SEND-CALL}] \\
\langle \mathcal{E}[\texttt{send.call}_x\ \bot],\ S,\ I\ \rangle & \rightarrow^{\texttt{rcv.ret}(x,h)} & \langle\ \mathcal{E}[h],\ S,\ I\ \rangle & & [\text{RCV-RET}] \\
\langle\ \mathcal{Q}[\bot],\ S,\ I\ \rangle & \rightarrow^{\texttt{rcv.call}(x,h)} & \langle \mathcal{Q}[\texttt{rcv.call}_x\ (v\ h)],\ S,\ I\ \rangle & I(x) = v & [\text{RCV-CALL}] \\
\langle\ \mathcal{Q}[\texttt{rcv.call}_x\ v],\ S,\ I\ \rangle & \rightarrow^{\texttt{send.ret}(x,h)} & \langle\ \mathcal{Q}[\bot],\ S,\ I[h \triangleright v]\ \rangle & & [\text{SEND-RET}]
\end{array}$$

Figure 1: CSI Machine

Domains

$$
\begin{array}{llll}
\textit{State} & \boxed{CSI \;\in\; \textit{Code} \times \textit{Store} \times \textit{Interface}} & & \\
\textit{Code} & C & ::= & \mathcal{E}[e] \mid \mathcal{Q}[\bot] \\
\textit{Store} & S & \in & \mathcal{P}(\textit{Var} \times \textit{Var} \times \textit{Value}) \\
\textit{Interface} & I & \in & \textit{Var} \to \textit{Value}
\end{array}
$$

$$
\begin{array}{llll}
\textit{Evaluation context} & \mathcal{E} & ::= & \mathcal{E}[\bullet\ e] \mid \mathcal{E}[v\ \bullet] \mid \mathcal{E}[\mathtt{ref}\ \bullet] \mid \mathcal{Q}[\mathtt{rcv.call}_x\ \bullet] \\
\textit{Quiescent context} & \mathcal{Q} & ::= & \bullet \mid \mathcal{E}[\mathtt{send.call}_x\ \bullet] \\
\textit{Value} & v & ::= & c \mid x \mid \lambda y.\, e \\
\textit{Handle} & h & ::= & c \mid x \\
\textit{Event} & a & ::= & \rho.\mathtt{ret}(x, h) \mid \rho.\mathtt{call}(x, h) \\
\textit{Direction} & \rho & ::= & \mathtt{send} \mid \mathtt{rcv} \\
\textit{Trace} & t & ::= & \vec{a}
\end{array}
$$

Transition relation $(\to) \;\subseteq\; \textit{State} \times \textit{Event}_\bot \times \textit{State}$

$$
\begin{array}{lllllll}
\langle\ \mathcal{E}[(\lambda x.\, e)\ v], & S, & I\ \rangle & \to & \langle\ \mathcal{E}[e[x := v]], & S, & I\ \rangle & & [\text{CALL}] \\
\langle\ \mathcal{E}[c\ v], & S, & I\ \rangle & \to & \langle\ \mathcal{E}[v'], & S, & I\ \rangle & v' = \delta(c, v) & [\text{PRIM}] \\
\langle\ \mathcal{E}[\mathtt{ref}\ v], & S, & I\ \rangle & \to & \langle\ \mathcal{E}[\mathtt{pair}\ x\ y], & S[(x,y) \mapsto v], & I\ \rangle & x, y \text{ fresh} & [\text{REF}] \\
\langle\ \mathcal{E}[x\ v], & S[(x,y) \mapsto v'], & I\ \rangle & \to & \langle\ \mathcal{E}[v'], & S[(x,y) \mapsto v'], & I\ \rangle & & [\text{GET}] \\
\langle\ \mathcal{E}[y\ v], & S[(x,y) \mapsto v'], & I\ \rangle & \to & \langle\ \mathcal{E}[v], & S[(x,y) \mapsto v], & I\ \rangle & & [\text{SET}] \\
\\
\langle\ \mathcal{E}[x\ v], & S, & I\ \rangle & \to^{\mathtt{send.call}(x,h)} & \langle\ \mathcal{E}[\mathtt{send.call}_x\ \bot], & S, & I[h \triangleright v]\ \rangle & x \notin BV(S) & [\text{SEND-CALL}] \\
\langle \mathcal{E}[\mathtt{send.call}_x\ \bot], & S, & I\ \rangle & \to^{\mathtt{rcv.ret}(x,h)} & \langle\ \mathcal{E}[h], & S, & I\ \rangle & & [\text{RCV-RET}] \\
\langle\ \mathcal{Q}[\bot], & S, & I\ \rangle & \to^{\mathtt{rcv.call}(x,h)} & \langle \mathcal{Q}[\mathtt{rcv.call}_x\ (v\ h)], & S, & I\ \rangle & I(x) = v & [\text{RCV-CALL}] \\
\langle\ \mathcal{Q}[\mathtt{rcv.call}_x\ v], & S, & I\ \rangle & \to^{\mathtt{send.ret}(x,h)} & \langle\ \mathcal{Q}[\bot], & S, & I[h \triangleright v]\ \rangle & & [\text{SEND-RET}]
\end{array}
$$

Figure 1: CSI Machine

Domains

State	CSI	$\in$	$Code \times Store \times Interface$	*Evaluation context*	$\mathcal{E}$	$::=$	$\mathcal{E}[\bullet\ e] \mid \mathcal{E}[v\ \bullet] \mid \mathcal{E}[\texttt{ref}\ \bullet] \mid \mathcal{Q}[\texttt{rcv.call}_x\ \bullet]$
				Quiescent context	$\mathcal{Q}$	$::=$	$\bullet \mid \mathcal{E}[\texttt{send.call}_x\ \bullet]$
Code	C	$::=$	$\mathcal{E}[e] \mid \mathcal{Q}[\perp]$	*Value*	v	$::=$	$c \mid x \mid \lambda y.\, e$
Store	S	$\in$	$\mathcal{P}(Var \times Var \times Value)$	*Handle*	h	$::=$	$c \mid x$
Interface	I	$\in$	$Var \to Value$	*Event*	a	$::=$	$\rho.\texttt{ret}(x,h) \mid \rho.\texttt{call}(x,h)$
				Direction	ρ	$::=$	$\texttt{send} \mid \texttt{rcv}$
				Trace	t	$::=$	$\vec{a}$

Transition relation $(\to) \subseteq State \times Event_\perp \times State$

$\langle\ \mathcal{E}[(\lambda x.\, e)\ v],$	$S,$	$I\ \rangle$	$\to$	$\langle\ \mathcal{E}[e[x := v]],$	$S,$	$I\ \rangle$		[CALL]
$\langle\ \mathcal{E}[c\ v],$	$S,$	$I\ \rangle$	$\to$	$\langle\ \mathcal{E}[v'],$	$S,$	$I\ \rangle$	$v' = \delta(c,v)$	[PRIM]
$\langle\ \mathcal{E}[\texttt{ref}\ v],$	$S,$	$I\ \rangle$	$\to$	$\langle\ \mathcal{E}[\texttt{pair}\ x\ y],$	$S[(x,y) \mapsto v],$	$I\ \rangle$	x, y fresh	[REF]
$\langle\ \mathcal{E}[x\ v],$	$S[(x,y) \mapsto v'],$	$I\ \rangle$	$\to$	$\langle\ \mathcal{E}[v'],$	$S[(x,y) \mapsto v'],$	$I\ \rangle$		[GET]
$\langle\ \mathcal{E}[y\ v],$	$S[(x,y) \mapsto v'],$	$I\ \rangle$	$\to$	$\langle\ \mathcal{E}[v],$	$S[(x,y) \mapsto v],$	$I\ \rangle$		[SET]
$\langle\ \mathcal{E}[x\ v],$	$S,$	$I\ \rangle$	$\to^{\texttt{send.call}(x,h)}$	$\langle\ \mathcal{E}[\texttt{send.call}_x\ \perp],$	$S,$	$I[h \triangleright v]\ \rangle$	$x \notin BV(S)$	[SEND-CALL]
$\langle \mathcal{E}[\texttt{send.call}_x\ \perp],$	$S,$	$I\ \rangle$	$\to^{\texttt{rcv.ret}(x,h)}$	$\langle\ \mathcal{E}[h],$	$S,$	$I\ \rangle$		[RCV-RET]
$\langle\ \mathcal{Q}[\perp],$	$S,$	$I\ \rangle$	$\to^{\texttt{rcv.call}(x,h)}$	$\langle \mathcal{Q}[\texttt{rcv.call}_x\ (v\ h)],$	$S,$	$I\ \rangle$	$I(x) = v$	[RCV-CALL]
$\langle\ \mathcal{Q}[\texttt{rcv.call}_x\ v],$	$S,$	$I\ \rangle$	$\to^{\texttt{send.ret}(x,h)}$	$\langle\ \mathcal{Q}[\perp],$	$S,$	$I[h \triangleright v]\ \rangle$		[SEND-RET]

Figure 1: CSI Machine

Domains

$$
\begin{array}{llll}
\mathit{State} & \mathit{CSI} & \in & \mathit{Code} \times \mathit{Store} \times \mathit{Interface} \\
 & & & \\
\mathit{Code} & C & ::= & \mathcal{E}[e] \mid \mathcal{Q}[\bot] \\
\mathit{Store} & S & \in & \mathcal{P}(\mathit{Var} \times \mathit{Var} \times \mathit{Value}) \\
\mathit{Interface} & I & \in & \mathit{Var} \rightarrow \mathit{Value}
\end{array}
$$

$$
\begin{array}{llll}
\mathit{Evaluation\ context} & \mathcal{E} & ::= & \mathcal{E}[\bullet\ e] \mid \mathcal{E}[v\ \bullet] \mid \mathcal{E}[\texttt{ref}\ \bullet] \mid \mathcal{Q}[\texttt{rcv.call}_x\ \bullet] \\
\mathit{Quiescent\ context} & \mathcal{Q} & ::= & \bullet \mid \mathcal{E}[\texttt{send.call}_x\ \bullet] \\
\mathit{Value} & v & ::= & c \mid x \mid \lambda y.\, e \\
\mathit{Handle} & h & ::= & c \mid x \\
\mathit{Event} & a & ::= & \rho.\texttt{ret}(x, h) \mid \rho.\texttt{call}(x, h) \\
\mathit{Direction} & \rho & ::= & \texttt{send} \mid \texttt{rcv} \\
\mathit{Trace} & t & ::= & \vec{a}
\end{array}
$$

Transition relation $(\rightarrow) \subseteq \mathit{State} \times \mathit{Event}_\bot \times \mathit{State}$

$$
\begin{array}{lllclllll}
\langle & \mathcal{E}[(\lambda x.\, e)\ v], & S, & I\ \rangle & \rightarrow & \langle\ \mathcal{E}[e[x := v]], & S, & I\ \rangle & & [\textsc{Call}] \\
\langle & \mathcal{E}[c\ v], & S, & I\ \rangle & \rightarrow & \langle\ \mathcal{E}[v'], & S, & I\ \rangle & v' = \delta(c, v) & [\textsc{Prim}] \\
\langle & \mathcal{E}[\texttt{ref}\ v], & S, & I\ \rangle & \rightarrow & \langle\ \mathcal{E}[\texttt{pair}\ x\ y], & S[(x, y) \mapsto v], & I\ \rangle & x, y \text{ fresh} & [\textsc{Ref}] \\
\langle & \mathcal{E}[x\ v], & S[(x, y) \mapsto v'], & I\ \rangle & \rightarrow & \langle\ \mathcal{E}[v'], & S[(x, y) \mapsto v'], & I\ \rangle & & [\textsc{Get}] \\
\langle & \mathcal{E}[y\ v], & S[(x, y) \mapsto v'], & I\ \rangle & \rightarrow & \langle\ \mathcal{E}[v], & S[(x, y) \mapsto v], & I\ \rangle & & [\textsc{Set}] \\
 & & & & & & & & & \\
\langle & \mathcal{E}[x\ v], & S, & I\ \rangle & \rightarrow^{\texttt{send.call}(x,h)} & \langle\ \mathcal{E}[\texttt{send.call}_x\ \bot], & S, & I[h \triangleright v]\ \rangle & x \notin \mathit{BV}(S) & [\textsc{Send-Call}] \\
\langle & \mathcal{E}[\texttt{send.call}_x\ \bot], & S, & I\ \rangle & \rightarrow^{\texttt{rcv.ret}(x,h)} & \langle\ \mathcal{E}[h], & S, & I\ \rangle & & [\textsc{Rcv-Ret}] \\
\langle & \mathcal{Q}[\bot], & S, & I\ \rangle & \rightarrow^{\texttt{rcv.call}(x,h)} & \langle\ \mathcal{Q}[\texttt{rcv.call}_x\ (v\ h)], & S, & I\ \rangle & I(x) = v & [\textsc{Rcv-Call}] \\
\langle & \mathcal{Q}[\texttt{rcv.call}_x\ v], & S, & I\ \rangle & \rightarrow^{\texttt{send.ret}(x,h)} & \langle\ \mathcal{Q}[\bot], & S, & I[h \triangleright v]\ \rangle & & [\textsc{Send-Ret}]
\end{array}
$$

Figure 1: CSI Machine

Domains

$$
\begin{array}{llll}
\textit{State} & \mathit{CSI} & \in & \mathit{Code} \times \mathit{Store} \times \mathit{Interface} \\
\\
\textit{Code} & C & ::= & \mathcal{E}[e] \ \mid\ \mathcal{Q}[\bot] \\
\textit{Store} & S & \in & \mathcal{P}(\mathit{Var} \times \mathit{Var} \times \mathit{Value}) \\
\textit{Interface} & I & \in & \mathit{Var} \rightarrow \mathit{Value}
\end{array}
$$

$$
\begin{array}{llll}
\textit{Evaluation context} & \mathcal{E} & ::= & \mathcal{E}[\bullet\ e] \ \mid\ \mathcal{E}[v\ \bullet] \ \mid\ \mathcal{E}[\texttt{ref}\ \bullet] \ \mid\ \mathcal{Q}[\texttt{rcv.call}_x\ \bullet] \\
\textit{Quiescent context} & \mathcal{Q} & ::= & \bullet \ \mid\ \mathcal{E}[\texttt{send.call}_x\ \bullet] \\
\textit{Value} & v & ::= & c \mid x \mid \lambda y.\, e \\
\textit{Handle} & h & ::= & c \mid x \\
\textit{Event} & a & ::= & \rho.\texttt{ret}(x, h) \mid \rho.\texttt{call}(x, h) \\
\textit{Direction} & \rho & ::= & \texttt{send} \mid \texttt{rcv} \\
\textit{Trace} & t & ::= & \vec{a}
\end{array}
$$

Transition relation $(\rightarrow) \ \subseteq\ \mathit{State} \times \mathit{Event}_\bot \times \mathit{State}$

$$
\begin{array}{lllll}
\langle\ \mathcal{E}[(\lambda x.\, e)\ v],\ S,\ I\ \rangle & \rightarrow & \langle\ \mathcal{E}[e[x := v]],\ S,\ I\ \rangle & & [\textsc{Call}] \\
\langle\ \mathcal{E}[c\ v],\ S,\ I\ \rangle & \rightarrow & \langle\ \mathcal{E}[v'],\ S,\ I\ \rangle & v' = \delta(c, v) & [\textsc{Prim}] \\
\langle\ \mathcal{E}[\texttt{ref}\ v],\ S,\ I\ \rangle & \rightarrow & \langle\ \mathcal{E}[\texttt{pair}\ x\ y],\ S[(x,y) \mapsto v],\ I\ \rangle & x, y \text{ fresh} & [\textsc{Ref}] \\
\langle\ \mathcal{E}[x\ v],\ S[(x,y) \mapsto v'],\ I\ \rangle & \rightarrow & \langle\ \mathcal{E}[v'],\ S[(x,y) \mapsto v'],\ I\ \rangle & & [\textsc{Get}] \\
\langle\ \mathcal{E}[y\ v],\ S[(x,y) \mapsto v'],\ I\ \rangle & \rightarrow & \langle\ \mathcal{E}[v],\ S[(x,y) \mapsto v],\ I\ \rangle & & [\textsc{Set}] \\
\\
\langle\ \mathcal{E}[x\ v],\ S,\ I\ \rangle & \rightarrow^{\texttt{send.call}(x,h)} & \langle\ \mathcal{E}[\texttt{send.call}_x\ \bot],\ S,\ I[h \triangleright v]\ \rangle & x \notin \mathit{BV}(S) & [\textsc{Send-Call}] \\
\langle \mathcal{E}[\texttt{send.call}_x\ \bot],\ S,\ I\ \rangle & \rightarrow^{\texttt{rcv.ret}(x,h)} & \langle\ \mathcal{E}[h],\ S,\ I\ \rangle & & [\textsc{Rcv-Ret}] \\
\langle\ \mathcal{Q}[\bot],\ S,\ I\ \rangle & \rightarrow^{\texttt{rcv.call}(x,h)} & \langle \mathcal{Q}[\texttt{rcv.call}_x\ (v\ h)],\ S,\ I\ \rangle & I(x) = v & [\textsc{Rcv-Call}] \\
\langle\ \mathcal{Q}[\texttt{rcv.call}_x\ v],\ S,\ I\ \rangle & \rightarrow^{\texttt{send.ret}(x,h)} & \langle\ \mathcal{Q}[\bot],\ S,\ I[h \triangleright v]\ \rangle & & [\textsc{Send-Ret}]
\end{array}
$$

Figure 1: CSI Machine

Domains

$$
\begin{array}{llll}
\textit{State} & CSI & \in & \textit{Code} \times \textit{Store} \times \textit{Interface} \\
\textit{Code} & C & ::= & \mathcal{E}[e] \mid \mathcal{Q}[\bot] \\
\textit{Store} & S & \in & \mathcal{P}(\textit{Var} \times \textit{Var} \times \textit{Value}) \\
\textit{Interface} & I & \in & \textit{Var} \to \textit{Value}
\end{array}
$$

$$
\begin{array}{llll}
\textit{Evaluation context} & \mathcal{E} & ::= & \mathcal{E}[\bullet\ e] \mid \mathcal{E}[v\ \bullet] \mid \mathcal{E}[\texttt{ref}\ \bullet] \mid \mathcal{Q}[\texttt{rcv.call}_x\ \bullet] \\
\textit{Quiescent context} & \mathcal{Q} & ::= & \bullet \mid \mathcal{E}[\texttt{send.call}_x\ \bullet] \\
\textit{Value} & v & ::= & c \mid x \mid \lambda y.\, e \\
\textit{Handle} & h & ::= & c \mid x \\
\textit{Event} & a & ::= & \rho.\texttt{ret}(x,h) \mid \rho.\texttt{call}(x,h) \\
\textit{Direction} & \rho & ::= & \texttt{send} \mid \texttt{rcv} \\
\textit{Trace} & t & ::= & \vec{a}
\end{array}
$$

Transition relation $(\to) \subseteq \textit{State} \times \textit{Event}_\bot \times \textit{State}$

$$
\begin{array}{lllll}
\langle \mathcal{E}[(\lambda x.\, e)\ v],\ S,\ I \rangle & \to & \langle \mathcal{E}[e[x := v]],\ S,\ I \rangle & & [\textsc{Call}] \\
\langle \mathcal{E}[c\ v],\ S,\ I \rangle & \to & \langle \mathcal{E}[v'],\ S,\ I \rangle & v' = \delta(c,v) & [\textsc{Prim}] \\
\langle \mathcal{E}[\texttt{ref}\ v],\ S,\ I \rangle & \to & \langle \mathcal{E}[\texttt{pair}\ x\ y],\ S[(x,y) \mapsto v],\ I \rangle & x, y \text{ fresh} & [\textsc{Ref}] \\
\langle \mathcal{E}[x\ v],\ S[(x,y) \mapsto v'],\ I \rangle & \to & \langle \mathcal{E}[v'],\ S[(x,y) \mapsto v'],\ I \rangle & & [\textsc{Get}] \\
\langle \mathcal{E}[y\ v],\ S[(x,y) \mapsto v'],\ I \rangle & \to & \langle \mathcal{E}[v],\ S[(x,y) \mapsto v],\ I \rangle & & [\textsc{Set}] \\
\\
\langle \mathcal{E}[x\ v],\ S,\ I \rangle & \to^{\texttt{send.call}(x,h)} & \langle \mathcal{E}[\texttt{send.call}_x\ \bot],\ S,\ I[h \triangleright v] \rangle & x \notin BV(S) & [\textsc{Send-Call}] \\
\langle \mathcal{E}[\texttt{send.call}_x\ \bot],\ S,\ I \rangle & \to^{\texttt{rcv.ret}(x,h)} & \langle \mathcal{E}[h],\ S,\ I \rangle & & [\textsc{Rcv-Ret}] \\
\langle \mathcal{Q}[\bot],\ S,\ I \rangle & \to^{\texttt{rcv.call}(x,h)} & \langle \mathcal{Q}[\texttt{rcv.call}_x\ (v\ h)],\ S,\ I \rangle & I(x) = v & [\textsc{Rcv-Call}] \\
\langle \mathcal{Q}[\texttt{rcv.call}_x\ v],\ S,\ I \rangle & \to^{\texttt{send.ret}(x,h)} & \langle \mathcal{Q}[\bot],\ S,\ I[h \triangleright v] \rangle & & [\textsc{Send-Ret}]
\end{array}
$$

Figure 1: CSI Machine

Domains

$$\begin{array}{llll}
\textit{State} & CSI & \in & \textit{Code} \times \textit{Store} \times \textit{Interface} \\
 & & & \\
\textit{Code} & C & ::= & \mathcal{E}[e] \mid \mathcal{Q}[\bot] \\
\textit{Store} & S & \in & \mathcal{P}(\textit{Var} \times \textit{Var} \times \textit{Value}) \\
\textit{Interface} & I & \in & \textit{Var} \rightarrow \textit{Value}
\end{array}$$

$$\begin{array}{llll}
\textit{Evaluation context} & \mathcal{E} & ::= & \mathcal{E}[\bullet\ e] \mid \mathcal{E}[v\ \bullet] \mid \mathcal{E}[\texttt{ref}\ \bullet] \mid \mathcal{Q}[\texttt{rcv.call}_x\ \bullet] \\
\textit{Quiescent context} & \mathcal{Q} & ::= & \bullet \mid \mathcal{E}[\texttt{send.call}_x\ \bullet] \\
\textit{Value} & v & ::= & c \mid x \mid \lambda y.\, e \\
\textit{Handle} & h & ::= & c \mid x \\
\textit{Event} & a & ::= & \rho.\texttt{ret}(x, h) \mid \rho.\texttt{call}(x, h) \\
\textit{Direction} & \rho & ::= & \texttt{send} \mid \texttt{rcv} \\
\textit{Trace} & t & ::= & \vec{a}
\end{array}$$

Transition relation $(\rightarrow) \subseteq \textit{State} \times \textit{Event}_\bot \times \textit{State}$

$$\begin{array}{lllll}
\langle\, \mathcal{E}[(\lambda x.\, e)\ v],\ S,\ I\,\rangle & \rightarrow & \langle\, \mathcal{E}[e[x := v]],\ S,\ I\,\rangle & & [\textsc{Call}] \\
\langle\, \mathcal{E}[c\ v],\ S,\ I\,\rangle & \rightarrow & \langle\, \mathcal{E}[v'],\ S,\ I\,\rangle & v' = \delta(c, v) & [\textsc{Prim}] \\
\langle\, \mathcal{E}[\texttt{ref}\ v],\ S,\ I\,\rangle & \rightarrow & \langle\, \mathcal{E}[\texttt{pair}\ x\ y],\ S[(x,y) \mapsto v],\ I\,\rangle & x, y \text{ fresh} & [\textsc{Ref}] \\
\langle\, \mathcal{E}[x\ v],\ S[(x,y) \mapsto v'],\ I\,\rangle & \rightarrow & \langle\, \mathcal{E}[v'],\ S[(x,y) \mapsto v'],\ I\,\rangle & & [\textsc{Get}] \\
\langle\, \mathcal{E}[y\ v],\ S[(x,y) \mapsto v'],\ I\,\rangle & \rightarrow & \langle\, \mathcal{E}[v],\ S[(x,y) \mapsto v],\ I\,\rangle & & [\textsc{Set}] \\
 & & & & \\
\langle\, \mathcal{E}[x\ v],\ S,\ I\,\rangle & \rightarrow^{\texttt{send.call}(x,h)} & \langle\, \mathcal{E}[\texttt{send.call}_x\ \bot],\ S,\ I[h \triangleright v]\,\rangle & x \notin BV(S) & [\textsc{Send-Call}] \\
\langle \mathcal{E}[\texttt{send.call}_x\ \bot],\ S,\ I\,\rangle & \rightarrow^{\texttt{rcv.ret}(x,h)} & \langle\, \mathcal{E}[h],\ S,\ I\,\rangle & & [\textsc{Rcv-Ret}] \\
\langle\, \mathcal{Q}[\bot],\ S,\ I\,\rangle & \rightarrow^{\texttt{rcv.call}(x,h)} & \langle \mathcal{Q}[\texttt{rcv.call}_x\ (v\ h)],\ S,\ I\,\rangle & I(x) = v & [\textsc{Rcv-Call}] \\
\langle\, \mathcal{Q}[\texttt{rcv.call}_x\ v],\ S,\ I\,\rangle & \rightarrow^{\texttt{send.ret}(x,h)} & \langle\, \mathcal{Q}[\bot],\ S,\ I[h \triangleright v]\,\rangle & & [\textsc{Send-Ret}]
\end{array}$$

Figure 1: CSI Machine

Domains

State	CSI	$\in$	$Code \times Store \times Interface$
Code	C	$::=$	$\mathcal{E}[e] \mid \mathcal{Q}[\bot]$
Store	S	$\in$	$\mathcal{P}(Var \times Var \times Value)$
Interface	I	$\in$	$Var \rightarrow Value$
Evaluation context	$\mathcal{E}$	$::=$	$\mathcal{E}[\bullet\ e] \mid \mathcal{E}[v\ \bullet] \mid \mathcal{E}[\texttt{ref}\ \bullet] \mid \mathcal{Q}[\texttt{rcv.call}_x\ \bullet]$
Quiescent context	$\mathcal{Q}$	$::=$	$\bullet \mid \mathcal{E}[\texttt{send.call}_x\ \bullet]$
Value	v	$::=$	$c \mid x \mid \lambda y.\, e$
Handle	h	$::=$	$c \mid x$
Event	a	$::=$	$\rho.\texttt{ret}(x, h) \mid \rho.\texttt{call}(x, h)$
Direction	ρ	$::=$	$\texttt{send} \mid \texttt{rcv}$
Trace	t	$::=$	$\vec{a}$

Transition relation $(\rightarrow) \subseteq State \times Event_\bot \times State$

$$
\begin{array}{lllllll}
\langle\ \mathcal{E}[(\lambda x.\, e)\ v], & S, & I\ \rangle & \rightarrow & \langle\ \mathcal{E}[e[x := v]], & S, & I\ \rangle & & [\textsc{Call}] \\
\langle\ \mathcal{E}[c\ v], & S, & I\ \rangle & \rightarrow & \langle\ \mathcal{E}[v'], & S, & I\ \rangle & v' = \delta(c, v) & [\textsc{Prim}] \\
\langle\ \mathcal{E}[\texttt{ref}\ v], & S, & I\ \rangle & \rightarrow & \langle\ \mathcal{E}[\texttt{pair}\ x\ y], & S[(x, y) \mapsto v], & I\ \rangle & x, y \text{ fresh} & [\textsc{Ref}] \\
\langle\ \mathcal{E}[x\ v], & S[(x, y) \mapsto v'], & I\ \rangle & \rightarrow & \langle\ \mathcal{E}[v'], & S[(x, y) \mapsto v'], & I\ \rangle & & [\textsc{Get}] \\
\langle\ \mathcal{E}[y\ v], & S[(x, y) \mapsto v'], & I\ \rangle & \rightarrow & \langle\ \mathcal{E}[v], & S[(x, y) \mapsto v], & I\ \rangle & & [\textsc{Set}] \\
\\
\langle\ \mathcal{E}[x\ v], & S, & I\ \rangle & \rightarrow^{\texttt{send.call}(x,h)} & \langle\ \mathcal{E}[\texttt{send.call}_x\ \bot], & S, & I[h \rhd v]\ \rangle & x \notin BV(S) & [\textsc{Send-Call}] \\
\langle \mathcal{E}[\texttt{send.call}_x\ \bot], & S, & I\ \rangle & \rightarrow^{\texttt{rcv.ret}(x,h)} & \langle\ \mathcal{E}[h], & S, & I\ \rangle & & [\textsc{Rcv-Ret}] \\
\langle\ \mathcal{Q}[\bot], & S, & I\ \rangle & \rightarrow^{\texttt{rcv.call}(x,h)} & \langle \mathcal{Q}[\texttt{rcv.call}_x\ (v\ h)], & S, & I\ \rangle & I(x) = v & [\textsc{Rcv-Call}] \\
\langle\ \mathcal{Q}[\texttt{rcv.call}_x\ v], & S, & I\ \rangle & \rightarrow^{\texttt{send.ret}(x,h)} & \langle\ \mathcal{Q}[\bot], & S, & I[h \rhd v]\ \rangle & & [\textsc{Send-Ret}]
\end{array}
$$

Figure 1: CSI Machine

Domains

$$
\begin{array}{llcl}
\textit{State} & CSI & \in & \textit{Code} \times \textit{Store} \times \textit{Interface} \\
\\
\textit{Code} & C & ::= & \mathcal{E}[e] \mid \mathcal{Q}[\bot] \\
\textit{Store} & S & \in & \mathcal{P}(\textit{Var} \times \textit{Var} \times \textit{Value}) \\
\textit{Interface} & I & \in & \textit{Var} \to \textit{Value}
\end{array}
$$

$$
\begin{array}{llcl}
\textit{Evaluation context} & \mathcal{E} & ::= & \mathcal{E}[\bullet\ e] \mid \mathcal{E}[v\ \bullet] \mid \mathcal{E}[\texttt{ref}\ \bullet] \mid \mathcal{Q}[\texttt{rcv.call}_x\ \bullet] \\
\textit{Quiescent context} & \mathcal{Q} & ::= & \bullet \mid \mathcal{E}[\texttt{send.call}_x\ \bullet] \\
\textit{Value} & v & ::= & c \mid x \mid \lambda y.\, e \\
\textit{Handle} & h & ::= & c \mid x \\
\textit{Event} & a & ::= & \rho.\texttt{ret}(x, h) \mid \rho.\texttt{call}(x, h) \\
\textit{Direction} & \rho & ::= & \texttt{send} \mid \texttt{rcv} \\
\textit{Trace} & t & ::= & \vec{a}
\end{array}
$$

Transition relation $(\to) \subseteq \textit{State} \times \textit{Event}_\bot \times \textit{State}$

$$
\begin{array}{lllll}
\langle\ \mathcal{E}[(\lambda x.\, e)\ v],\ S,\ I\ \rangle & \to & \langle\ \mathcal{E}[e[x := v]],\ S,\ I\ \rangle & & [\text{CALL}] \\
\langle\ \mathcal{E}[c\ v],\ S,\ I\ \rangle & \to & \langle\ \mathcal{E}[v'],\ S,\ I\ \rangle & v' = \delta(c, v) & [\text{PRIM}] \\
\langle\ \mathcal{E}[\texttt{ref}\ v],\ S,\ I\ \rangle & \to & \langle\ \mathcal{E}[\texttt{pair}\ x\ y],\ S[(x, y) \mapsto v],\ I\ \rangle & x, y \text{ fresh} & [\text{REF}] \\
\langle\ \mathcal{E}[x\ v],\ S[(x, y) \mapsto v'],\ I\ \rangle & \to & \langle\ \mathcal{E}[v'],\ S[(x, y) \mapsto v'],\ I\ \rangle & & [\text{GET}] \\
\langle\ \mathcal{E}[y\ v],\ S[(x, y) \mapsto v'],\ I\ \rangle & \to & \langle\ \mathcal{E}[v],\ S[(x, y) \mapsto v],\ I\ \rangle & & [\text{SET}] \\
\\
\langle\ \mathcal{E}[x\ v],\ S,\ I\ \rangle & \to^{\texttt{send.call}(x,h)} & \langle\ \mathcal{E}[\texttt{send.call}_x\ \bot],\ S,\ I[h \triangleright v]\ \rangle & x \notin BV(S) & [\text{SEND-CALL}] \\
\langle \mathcal{E}[\texttt{send.call}_x\ \bot],\ S,\ I\ \rangle & \to^{\texttt{rcv.ret}(x,h)} & \langle\ \mathcal{E}[h],\ S,\ I\ \rangle & & [\text{RCV-RET}] \\
\langle\ \mathcal{Q}[\bot],\ S,\ I\ \rangle & \to^{\texttt{rcv.call}(x,h)} & \langle \mathcal{Q}[\texttt{rcv.call}_x\ (v\ h)],\ S,\ I\ \rangle & I(x) = v & [\text{RCV-CALL}] \\
\langle\ \mathcal{Q}[\texttt{rcv.call}_x\ v],\ S,\ I\ \rangle & \to^{\texttt{send.ret}(x,h)} & \langle\ \mathcal{Q}[\bot],\ S,\ I[h \triangleright v]\ \rangle & & [\text{SEND-RET}]
\end{array}
$$

Figure 2: Example of Linking and Running Two CSI Machines Concurrently

The two CSI machines shown below cooperate to evaluate $\mathit{linkRun}([\![H]\!], [\![\mathit{twice}]\!])$. After the initial bootstrapping, `send` events of one machine match `rcv` events of the other and vice-versa. The final $\texttt{send.ret}(y, 6)$ event reports the result of the execution is 6.

Evaluation of $H = (\lambda t.\, t\ (\lambda x.\, x\texttt{+}1)\ 4)$

$I_1 = [y \mapsto (\lambda t.\, t\ (\lambda x.\, x\texttt{+}1)\ 4)]$

$J_1 = [y \mapsto (\lambda t.\, t\ (\lambda x.\, x\texttt{+}1)\ 4),\ f \mapsto (\lambda x.\, x\texttt{+}1)]$

$$
\begin{array}{lll}
\langle\, \texttt{rcv.call}_{start}\ (\lambda t.\, t\ (\lambda x.\, x\texttt{+}1)\ 4), & \emptyset, \emptyset\ \rangle & \rightarrow^{\texttt{send.ret}(start, y)} \\
\langle\, \bot, & \emptyset, I_1\ \rangle & \rightarrow^{\texttt{rcv.call}(y, t)} \\
\langle\, \texttt{rcv.call}_y\ ((\lambda t.\, t\ (\lambda x.\, x\texttt{+}1)\ 4)\ t), & \emptyset, I_1\ \rangle & \rightarrow \\
\langle\, \texttt{rcv.call}_y\ ((t\ (\lambda x.\, x\texttt{+}1))\ 4), & \emptyset, I_1\ \rangle & \rightarrow^{\texttt{send.call}(t, f)} \\
\langle\, \texttt{rcv.call}_y\ (\texttt{send.call}_t\ \bot)\ 4), & \emptyset, J_1\ \rangle & \\
 & & \rightarrow^{\texttt{rcv.ret}(t, g)} \\
\langle\, \texttt{rcv.call}_y\ (g\ 4), & \emptyset, J_1\ \rangle & \rightarrow^{\texttt{send.call}(g, 4)} \\
\langle\, \texttt{rcv.call}_y\ (\texttt{send.call}_g\ \bot), & \emptyset, J_1\ \rangle & \\
 & & \rightarrow^{\texttt{rcv.call}(f, 4)} \\
\langle\, \texttt{rcv.call}_y\ (\texttt{send.call}_g\ (\texttt{rcv.call}_f\ ((\lambda x.\, x\texttt{+}1)\ 4))), & \emptyset, J_1\ \rangle & \rightarrow^2 \\
\langle\, \texttt{rcv.call}_y\ (\texttt{send.call}_g\ (\texttt{rcv.call}_f\ 5)), & \emptyset, J_1\ \rangle & \rightarrow^{\texttt{send.ret}(f, 5)} \\
\langle\, \texttt{rcv.call}_y\ (\texttt{send.call}_g\ \bot), & \emptyset, J_1\ \rangle & \rightarrow^{\texttt{rcv.call}(f, 5)} \\
\langle\, \texttt{rcv.call}_y\ (\texttt{send.call}_g\ (\texttt{rcv.call}_f\ ((\lambda x.\, x\texttt{+}1)\ 5))), & \emptyset, J_1\ \rangle & \rightarrow^2 \\
\langle\, \texttt{rcv.call}_y\ (\texttt{send.call}_g\ (\texttt{rcv.call}_f\ 6)), & \emptyset, J_1\ \rangle & \rightarrow^{\texttt{send.ret}(f, 6)} \\
\langle\, \texttt{rcv.call}_y\ (\texttt{send.call}_g\ \bot), & \emptyset, J_1\ \rangle & \rightarrow^{\texttt{rcv.ret}(g, 6)} \\
\langle\, \texttt{rcv.call}_y\ 6, & \emptyset, J_1\ \rangle & \rightarrow^{\texttt{send.ret}(y, 6)} \\
\langle\, \bot, & \emptyset, J_1\ \rangle &
\end{array}
$$

Evaluation of $\mathit{twice} = (\lambda f.\, \lambda x.\, f\ (f\ x))$

$I_2 = [t \mapsto (\lambda f.\, \lambda x.\, f\ (f\ x))]$

$J_2 = [t \mapsto (\lambda f.\, \lambda x.\, f\ (f\ x)),\ g \mapsto (\lambda x.\, f\ (f\ x))]$

$$
\begin{array}{lll}
 & \langle\, \texttt{rcv.call}_{start}\ (\lambda f.\, \lambda x.\, f\ (f\ x)), & \emptyset, \emptyset\ \rangle \\
\rightarrow^{\texttt{send.ret}(start, t)} & \langle\, \bot, & \emptyset, I_2\ \rangle \\
 & & \\
\rightarrow^{\texttt{rcv.call}(t, f)} & \langle\, \texttt{rcv.call}_t\ ((\lambda f.\, \lambda x.\, f\ (f\ x))\ f), & \emptyset, I_2\ \rangle \\
\rightarrow & \langle\, \texttt{rcv.call}_t\ (\lambda x.\, f\ (f\ x)), & \emptyset, I_2\ \rangle \\
\rightarrow^{\texttt{send.ret}(t, g)} & \langle\, \bot, & \emptyset, J_2\ \rangle \\
\rightarrow^{\texttt{rcv.call}(g, 4)} & \langle\, \texttt{rcv.call}_g\ ((\lambda x.\, f\ (f\ x))\ 4), & \emptyset, J_2\ \rangle \\
\rightarrow & \langle\, \texttt{rcv.call}_g\ (f\ (f\ 4)), & \emptyset, J_2\ \rangle \\
\rightarrow^{\texttt{send.call}(f, 4)} & \langle\, \texttt{rcv.call}_g\ (\texttt{send.call}_f\ (f\ \bot)), & \emptyset, J_2\ \rangle \\
 & & \\
\rightarrow^{\texttt{rcv.ret}(f, 5)} & \langle\, \texttt{rcv.call}_g\ (f\ 5), & \emptyset, J_2\ \rangle \\
\rightarrow^{\texttt{send.call}(f, 5)} & \langle\, \texttt{rcv.call}_g\ (\texttt{send.call}_f\ \bot), & \emptyset, J_2\ \rangle \\
 & & \\
\rightarrow^{\texttt{rcv.ret}(f, 6)} & \langle\, \texttt{rcv.call}_g\ 6, & \emptyset, J_2\ \rangle \\
\rightarrow^{\texttt{send.ret}(g, 6)} & \langle\, \bot, & \emptyset, J_2\ \rangle
\end{array}
$$

Figure 2: Example of Linking and Running Two CSI Machines Concurrently

The two CSI machines shown below cooperate to evaluate $\mathit{linkRun}([\![H]\!], [\![\mathit{twice}]\!])$. After the initial bootstrapping, `send` events of one machine match `rcv` events of the other and vice-versa. The final $\mathtt{send.ret}(y, 6)$ event reports the result of the execution is 6.

Evaluation of $H = (\lambda t.\, t\ (\lambda x.\, x{+}1)\ 4)$

$I_1 = [y \mapsto (\lambda t.\, t\ (\lambda x.\, x{+}1)\ 4)]$

$J_1 = [y \mapsto (\lambda t.\, t\ (\lambda x.\, x{+}1)\ 4),\ f \mapsto (\lambda x.\, x{+}1)]$

$$
\begin{array}{lll}
\langle\, \mathtt{rcv.call}_{start}\ (\lambda t.\, t\ (\lambda x.\, x{+}1)\ 4), & \emptyset, \emptyset\ \rangle & \rightarrow^{\mathtt{send.ret}(start,y)} \\
\langle\, \bot, & \emptyset, I_1\ \rangle & \rightarrow^{\mathtt{rcv.call}(y,t)} \\
\langle\, \mathtt{rcv.call}_y\ ((\lambda t.\, t\ (\lambda x.\, x{+}1)\ 4)\ t), & \emptyset, I_1\ \rangle & \rightarrow \\
\langle\, \mathtt{rcv.call}_y\ ((t\ (\lambda x.\, x{+}1))\ 4), & \emptyset, I_1\ \rangle & \rightarrow^{\mathtt{send.call}(t,f)} \\
\langle\, \mathtt{rcv.call}_y\ (\mathtt{send.call}_t\ \bot)\ 4), & \emptyset, J_1\ \rangle & \\
 & & \rightarrow^{\mathtt{rcv.ret}(t,g)} \\
\langle\, \mathtt{rcv.call}_y\ (g\ 4), & \emptyset, J_1\ \rangle & \rightarrow^{\mathtt{send.call}(g,4)} \\
\langle\, \mathtt{rcv.call}_y\ (\mathtt{send.call}_g\ \bot), & \emptyset, J_1\ \rangle & \\
 & & \rightarrow^{\mathtt{rcv.call}(f,4)} \\
\langle\, \mathtt{rcv.call}_y\ (\mathtt{send.call}_g\ (\mathtt{rcv.call}_f\ ((\lambda x.\, x{+}1)\ 4))), & \emptyset, J_1\ \rangle & \rightarrow^{2} \\
\langle\, \mathtt{rcv.call}_y\ (\mathtt{send.call}_g\ (\mathtt{rcv.call}_f\ 5)), & \emptyset, J_1\ \rangle & \rightarrow^{\mathtt{send.ret}(f,5)} \\
\langle\, \mathtt{rcv.call}_y\ (\mathtt{send.call}_g\ \bot), & \emptyset, J_1\ \rangle & \rightarrow^{\mathtt{rcv.call}(f,5)} \\
\langle\, \mathtt{rcv.call}_y\ (\mathtt{send.call}_g\ (\mathtt{rcv.call}_f\ ((\lambda x.\, x{+}1)\ 5))), & \emptyset, J_1\ \rangle & \rightarrow^{2} \\
\langle\, \mathtt{rcv.call}_y\ (\mathtt{send.call}_g\ (\mathtt{rcv.call}_f\ 6)), & \emptyset, J_1\ \rangle & \rightarrow^{\mathtt{send.ret}(f,6)} \\
\langle\, \mathtt{rcv.call}_y\ (\mathtt{send.call}_g\ \bot), & \emptyset, J_1\ \rangle & \rightarrow^{\mathtt{rcv.ret}(g,6)} \\
\langle\, \mathtt{rcv.call}_y\ 6, & \emptyset, J_1\ \rangle & \rightarrow^{\mathtt{send.ret}(y,6)} \\
\langle\, \bot, & \emptyset, J_1\ \rangle &
\end{array}
$$

Evaluation of $\mathit{twice} = (\lambda f.\, \lambda x.\, f\ (f\ x))$

$I_2 = [t \mapsto (\lambda f.\, \lambda x.\, f\ (f\ x))]$

$J_2 = [t \mapsto (\lambda f.\, \lambda x.\, f\ (f\ x)),\ g \mapsto (\lambda x.\, f\ (f\ x))]$

$$
\begin{array}{lll}
 & \langle\, \mathtt{rcv.call}_{start}\ (\lambda f.\, \lambda x.\, f\ (f\ x)), & \emptyset, \emptyset\ \rangle \\
\rightarrow^{\mathtt{send.ret}(start,t)} & \langle\, \bot, & \emptyset, I_2\ \rangle \\
\rightarrow^{\mathtt{rcv.call}(t,f)} & \langle\, \mathtt{rcv.call}_t\ ((\lambda f.\, \lambda x.\, f\ (f\ x))\ f), & \emptyset, I_2\ \rangle \\
\rightarrow & \langle\, \mathtt{rcv.call}_t\ (\lambda x.\, f\ (f\ x)), & \emptyset, I_2\ \rangle \\
\rightarrow^{\mathtt{send.ret}(t,g)} & \langle\, \bot, & \emptyset, J_2\ \rangle \\
\rightarrow^{\mathtt{rcv.call}(g,4)} & \langle\, \mathtt{rcv.call}_g\ ((\lambda x.\, f\ (f\ x))\ 4), & \emptyset, J_2\ \rangle \\
\rightarrow & \langle\, \mathtt{rcv.call}_g\ (f\ (f\ 4)), & \emptyset, J_2\ \rangle \\
\rightarrow^{\mathtt{send.call}(f,4)} & \langle\, \mathtt{rcv.call}_g\ (\mathtt{send.call}_f\ (f\ \bot)), & \emptyset, J_2\ \rangle \\
\rightarrow^{\mathtt{rcv.ret}(f,5)} & \langle\, \mathtt{rcv.call}_g\ (f\ 5), & \emptyset, J_2\ \rangle \\
\rightarrow^{\mathtt{send.call}(f,5)} & \langle\, \mathtt{rcv.call}_g\ (\mathtt{send.call}_f\ \bot), & \emptyset, J_2\ \rangle \\
\rightarrow^{\mathtt{rcv.ret}(f,6)} & \langle\, \mathtt{rcv.call}_g\ 6, & \emptyset, J_2\ \rangle \\
\rightarrow^{\mathtt{send.ret}(g,6)} & \langle\, \bot, & \emptyset, J_2\ \rangle
\end{array}
$$

Figure 2: Example of Linking and Running Two CSI Machines Concurrently

The two CSI machines shown below cooperate to evaluate $linkRun([\![H]\!], [\![twice]\!])$. After the initial bootstrapping, `send` events of one machine match `rcv` events of the other and vice-versa. The final $\texttt{send.ret}(y, 6)$ event reports the result of the execution is 6.

Evaluation of $H = (\lambda t.\, t\ (\lambda x.\, x{+}1)\ 4)$ $I_1 = [y \mapsto (\lambda t.\, t\ (\lambda x.\, x{+}1)\ 4)]$ $J_1 = [y \mapsto (\lambda t.\, t\ (\lambda x.\, x{+}1)\ 4),\ f \mapsto (\lambda x.\, x{+}1)]$			Evaluation of $twice = (\lambda f.\, \lambda x.\, f\ (f\ x))$ $I_2 = [t \mapsto (\lambda f.\, \lambda x.\, f\ (f\ x))]$ $J_2 = [t \mapsto (\lambda f.\, \lambda x.\, f\ (f\ x)),\ g \mapsto (\lambda x.\, f\ (f\ x))]$
$\langle \texttt{rcv.call}_{start}\ (\lambda t.\, t\ (\lambda x.\, x{+}1)\ 4),\ \emptyset, \emptyset \rangle$	$\rightarrow^{\texttt{send.ret}(start,y)}$		$\langle \texttt{rcv.call}_{start}\ (\lambda f.\, \lambda x.\, f\ (f\ x)),\ \emptyset, \emptyset \rangle$
$\langle \bot,\ \emptyset, I_1 \rangle$	$\rightarrow^{\texttt{rcv.call}(y,t)}$	$\rightarrow^{\texttt{send.ret}(start,t)}$	$\langle \bot,\ \emptyset, I_2 \rangle$
$\langle \texttt{rcv.call}_y\ ((\lambda t.\, t\ (\lambda x.\, x{+}1)\ 4)\ t),\ \emptyset, I_1 \rangle$	$\rightarrow$		
$\langle \texttt{rcv.call}_y\ ((t\ (\lambda x.\, x{+}1))\ 4),\ \emptyset, I_1 \rangle$	$\rightarrow^{\texttt{send.call}(t,f)}$	$\rightarrow^{\texttt{rcv.call}(t,f)}$	$\langle \texttt{rcv.call}_t\ ((\lambda f.\, \lambda x.\, f\ (f\ x))\ f),\ \emptyset, I_2 \rangle$
$\langle \texttt{rcv.call}_y\ (\texttt{send.call}_t\ \bot)\ 4),\ \emptyset, J_1 \rangle$		$\rightarrow$	$\langle \texttt{rcv.call}_t\ (\lambda x.\, f\ (f\ x)),\ \emptyset, I_2 \rangle$
	$\rightarrow^{\texttt{rcv.ret}(t,g)}$	$\rightarrow^{\texttt{send.ret}(t,g)}$	$\langle \bot,\ \emptyset, J_2 \rangle$
$\langle \texttt{rcv.call}_y\ (g\ 4),\ \emptyset, J_1 \rangle$	$\rightarrow^{\texttt{send.call}(g,4)}$	$\rightarrow^{\texttt{rcv.call}(g,4)}$	$\langle \texttt{rcv.call}_g\ ((\lambda x.\, f\ (f\ x))\ 4),\ \emptyset, J_2 \rangle$
$\langle \texttt{rcv.call}_y\ (\texttt{send.call}_g\ \bot),\ \emptyset, J_1 \rangle$		$\rightarrow$	$\langle \texttt{rcv.call}_g\ (f\ (f\ 4)),\ \emptyset, J_2 \rangle$
	$\rightarrow^{\texttt{rcv.call}(f,4)}$	$\rightarrow^{\texttt{send.call}(f,4)}$	$\langle \texttt{rcv.call}_g\ (\texttt{send.call}_f\ (f\ \bot)),\ \emptyset, J_2 \rangle$
$\langle \texttt{rcv.call}_y\ (\texttt{send.call}_g\ (\texttt{rcv.call}_f\ ((\lambda x.\, x{+}1)\ 4))),\ \emptyset, J_1 \rangle$	$\rightarrow^2$		
$\langle \texttt{rcv.call}_y\ (\texttt{send.call}_g\ (\texttt{rcv.call}_f\ 5)),\ \emptyset, J_1 \rangle$	$\rightarrow^{\texttt{send.ret}(f,5)}$	$\rightarrow^{\texttt{rcv.ret}(f,5)}$	$\langle \texttt{rcv.call}_g\ (f\ 5),\ \emptyset, J_2 \rangle$
$\langle \texttt{rcv.call}_y\ (\texttt{send.call}_g\ \bot),\ \emptyset, J_1 \rangle$	$\rightarrow^{\texttt{rcv.call}(f,5)}$	$\rightarrow^{\texttt{send.call}(f,5)}$	$\langle \texttt{rcv.call}_g\ (\texttt{send.call}_f\ \bot),\ \emptyset, J_2 \rangle$
$\langle \texttt{rcv.call}_y\ (\texttt{send.call}_g\ (\texttt{rcv.call}_f\ ((\lambda x.\, x{+}1)\ 5))),\ \emptyset, J_1 \rangle$	$\rightarrow^2$		
$\langle \texttt{rcv.call}_y\ (\texttt{send.call}_g\ (\texttt{rcv.call}_f\ 6)),\ \emptyset, J_1 \rangle$	$\rightarrow^{\texttt{send.ret}(f,6)}$	$\rightarrow^{\texttt{rcv.ret}(f,6)}$	$\langle \texttt{rcv.call}_g\ 6,\ \emptyset, J_2 \rangle$
$\langle \texttt{rcv.call}_y\ (\texttt{send.call}_g\ \bot),\ \emptyset, J_1 \rangle$	$\rightarrow^{\texttt{rcv.ret}(g,6)}$	$\rightarrow^{\texttt{send.ret}(g,6)}$	$\langle \bot,\ \emptyset, J_2 \rangle$
$\langle \texttt{rcv.call}_y\ 6,\ \emptyset, J_1 \rangle$	$\rightarrow^{\texttt{send.ret}(y,6)}$		
$\langle \bot,\ \emptyset, J_1 \rangle$			

Step 2: Contracts are Trace Predicates

$$[\![e]\!]$$

$$K \cap [\![e]\!]$$

Universal Contract:

$$[\![(\lambda x.\ x)]\!]$$

Properties

- Complete: Any computable predicate
- Non-interference: No new behaviors

Step 3: Programming Simple Contracts

$$\tau = (Constant \to Bool) \times (Unit \to (False + (\tau \times \tau)))$$

```
guard M =
 λx. if (constant? x) then
       assert ((fst M) x) != false
       x
     else
       let MM = (snd M)()
       assert MM != false
       λy. (guard (snd MM) (x (guard (fst MM) y)))
```

$$\tau = (Constant \rightarrow Bool) \times (Unit \rightarrow (False + (\tau \times \tau)))$$

```
guard M =
 λx. if (constant? x) then
         assert ((fst M) x) != false
         x
       else
         let MM = (snd M)()
         assert MM != false
         λy. (guard (snd MM) (x (guard (fst MM) y)))
```

$$\tau = (Constant \rightarrow Bool) \times (Unit \rightarrow (False + (\tau \times \tau)))$$

```
guard M =
 λx. if (constant? x) then
        assert ((fst M) x) != false
        x
      else
        let MM = (snd M)()
        assert MM != false
        λy. (guard (snd MM) (x (guard (fst MM) y)))
```

$$\tau = (Constant \to Bool) \times (Unit \to (False + (\tau \times \tau)))$$

```
guard M =
 λx. if (constant? x) then
         assert ((fst M) x) != false
         x
       else
         let MM = (snd M)()
         assert MM != false
         λy. (guard (snd MM) (x (guard (fst MM) y)))
```

$$\tau = (Constant \rightarrow Bool) \times (Unit \rightarrow (False + (\tau \times \tau)))$$

```
guard M =
 λx. if (constant? x) then
        assert ((fst M) x) != false
        x
      else
        let MM = (snd M)()
        assert MM != false
        λy. (guard (snd MM) (x (guard (fst MM) y)))
```

$$\tau = (Constant \rightarrow Bool) \times (Unit \rightarrow (False + (\tau \times \tau)))$$

```
guard M =
 λx. if (constant? x) then
        assert ((fst M) x) != false
        x
      else
        let MM = (snd M)()
        assert MM != false
        λy. (guard (snd MM) (x (guard (fst MM) y)))
```

$$\tau = (Constant \rightarrow Bool) \times (Unit \rightarrow (False + (\tau \times \tau)))$$

```
guard M =
 λx. if (constant? x) then
        assert ((fst M) x) != false
        x
      else
        let MM = (snd M)()
        assert MM != false
        λy. (guard (snd MM) (x (guard (fst MM) y)))
```

$$\tau = (Constant \to Bool) \times (Unit \to (False + (\tau \times \tau)))$$

```
guard M =
 λx. if (constant? x) then
       assert ((fst M) x) != false
       x
     else
       let MM = (snd M)()
       assert MM != false
       λy. (guard (snd MM) (x (guard (fst MM) y)))
```

$$\tau = (Constant \rightarrow Bool) \times (Unit \rightarrow (False + (\tau \times \tau)))$$

```
guard M =
 λx. if (constant? x) then
        assert ((fst M) x) != false
        x
      else
        let MM = (snd M)()
        assert MM != false
        λy. (guard (snd MM) (x (guard (fst MM) y)))
```

$$\tau = (Constant \rightarrow Bool) \times (Unit \rightarrow (False + (\tau \times \tau)))$$

```
guard M =
 λx. if (constant? x) then
       assert ((fst M) x) != false
       x
     else
       let MM = (snd M)()
       assert MM != false
       λy. (guard (snd MM) (x (guard (fst MM) y)))
```

Step 4: Programming Temporal Contracts

Figure 3: Declarative HOT Contracts

$$
\begin{array}{rcll}
M & ::= & S\ \texttt{where}\ R & \text{HOT contract} \\
S & ::= & \mathit{flat}(e) \mid n : S_1 \mapsto S_2 & \text{Structural contract} \\
R & ::= & A \mid !A \mid RR \mid R^* \mid \texttt{not}\ R \mid R \cup R & \text{Temporal contract} \\
 & \mid & \ldots \mid \texttt{call}(n, ?x)\ R \mid \texttt{ret}(n, ?x)\ R & \\
A & ::= & \texttt{call}(n, p) \mid \texttt{ret}(n, p) & \text{Event patterns} \\
p & ::= & \_ \mid x \mid c & \text{Value patterns} \\
n & \in & \mathit{Name} & \text{Function names}
\end{array}
$$

Figure 3: Declarative HOT Contracts

$$
\begin{array}{rcll}
M & ::= & S\ \texttt{where}\ R & \text{HOT contract} \\
S & ::= & \mathit{flat}(e) \mid n : S_1 \mapsto S_2 & \text{Structural contract} \\
R & ::= & A \mid !A \mid RR \mid R^* \mid \texttt{not}\ R \mid R \cup R & \text{Temporal contract} \\
 & \mid & \ldots \mid \texttt{call}(n, ?x)\ R \mid \texttt{ret}(n, ?x)\ R & \\
A & ::= & \texttt{call}(n, p) \mid \texttt{ret}(n, p) & \text{Event patterns} \\
p & ::= & \_ \mid x \mid c & \text{Value patterns} \\
n & \in & \mathit{Name} & \text{Function names}
\end{array}
$$

$$
\begin{array}{lcl}
\texttt{let s} & = & s_0 \\
\quad \texttt{call}_n & = & \lambda i. \cdots \text{check and update } \texttt{s} \text{ appropriately } \cdots \\
\quad \ldots & & \ldots \\
\quad \texttt{ret}_n & = & \lambda o. \cdots \text{check and update } \texttt{s} \text{ appropriately } \cdots \\
\quad \ldots & & \ldots \\
\multicolumn{3}{l}{\texttt{in } \mathit{compile}(\lambda\texttt{x.true}, S)}
\end{array}
$$

$$\mathit{compile} : (\mathit{Constant} \cup \{\lambda\texttt{x.x}\} \mapsto \mathit{Bool}) \times S \mapsto \mathit{Monitor}$$

$\mathit{compile}(f, \mathit{flat}(e)) \stackrel{\texttt{def}}{=}$

$$
\begin{array}{l}
\texttt{pair } (\lambda\texttt{x. } (e \texttt{ x}) \texttt{ \&\& } (f \texttt{ x})) \\
\qquad\;\; (\lambda\texttt{. false})
\end{array}
$$

$\mathit{compile}(f, n : S_1 \mapsto S_2) \stackrel{\texttt{def}}{=}$

$$
\begin{array}{l}
\texttt{pair } (\lambda\texttt{x. false}) \\
\qquad\;\; (\lambda\texttt{. } (f \texttt{ } (\lambda\texttt{x.x})) \texttt{ \&\& } (\texttt{pair } \mathit{compile}(\texttt{call}_n, S_1) \\
\qquad\qquad\qquad\qquad\qquad\qquad\qquad\quad \mathit{compile}(\texttt{ret}_n, S_2))))
\end{array}
$$

$\mathit{compile}(f, y) \stackrel{\texttt{def}}{=}$

$$
\begin{array}{l}
\texttt{let (chkconst,chkfn) = } y \\
\texttt{pair } (\lambda\texttt{x. } (f \texttt{ x}) \quad\quad \texttt{ \&\& (chkconst x))} \\
\qquad\;\; (\lambda\texttt{. } (f \texttt{ } (\lambda\texttt{x.x})) \texttt{ \&\& (chkfn()))}
\end{array}
$$

$$
\begin{array}{lll}
\texttt{let s} & = & s_0 \\
\quad \texttt{call}_n & = & \lambda i. \cdots \text{check and update } \texttt{s} \text{ appropriately } \cdots \\
\quad \ldots & & \ldots \\
\quad \texttt{ret}_n & = & \lambda o. \cdots \text{check and update } \texttt{s} \text{ appropriately } \cdots \\
\quad \ldots & & \ldots \\
\multicolumn{3}{l}{\texttt{in } \mathit{compile}(\lambda \texttt{x.true}, S)}
\end{array}
$$

$$\mathit{compile} : (\mathit{Constant} \cup \{\lambda \texttt{x.x}\} \mapsto \mathit{Bool}) \times S \mapsto \mathit{Monitor}$$

$\mathit{compile}(f, \mathit{flat}(e)) \stackrel{\texttt{def}}{=}$

$$
\begin{array}{l}
\texttt{pair } (\lambda \texttt{x. } (e\ \texttt{x}) \ \texttt{\&\&}\ (f\ \texttt{x})) \\
\qquad\quad (\lambda \texttt{. false})
\end{array}
$$

$\mathit{compile}(f, n : S_1 \mapsto S_2) \stackrel{\texttt{def}}{=}$

$$
\begin{array}{l}
\texttt{pair } (\lambda \texttt{x. false}) \\
\qquad\quad (\lambda \texttt{. } (f\ (\lambda \texttt{x.x}))\ \texttt{\&\&}\ (\texttt{pair } \mathit{compile}(\texttt{call}_n, S_1) \\
\qquad\qquad\qquad\qquad\qquad\qquad\qquad\quad \mathit{compile}(\texttt{ret}_n,\ S_2))))
\end{array}
$$

$\mathit{compile}(f, y) \stackrel{\texttt{def}}{=}$

$$
\begin{array}{l}
\texttt{let (chkconst,chkfn) = } y \\
\texttt{pair } (\lambda \texttt{x. } (f\ \texttt{x}) \qquad \texttt{\&\& (chkconst x))} \\
\qquad\quad (\lambda \texttt{. } (f\ (\lambda \texttt{x.x}))\ \texttt{\&\& (chkfn()))}
\end{array}
$$

$$
\begin{array}{lll}
\texttt{let s} & = & s_0 \\
\quad \texttt{call}_n & = & \lambda i. \cdots \text{check and update } \texttt{s} \text{ appropriately } \cdots \\
\quad \ldots & & \ldots \\
\quad \texttt{ret}_n & = & \lambda o. \cdots \text{check and update } \texttt{s} \text{ appropriately } \cdots \\
\quad \ldots & & \ldots \\
\multicolumn{3}{l}{\texttt{in } \mathit{compile}(\lambda\texttt{x.true}, S)}
\end{array}
$$

$$\mathit{compile} : (\mathit{Constant} \cup \{\lambda\texttt{x.x}\} \mapsto \mathit{Bool}) \times S \mapsto \mathit{Monitor}$$

$$\mathit{compile}(f, \mathit{flat}(e)) \stackrel{\text{def}}{=}$$
$$\quad \texttt{pair } (\lambda\texttt{x. } (e \texttt{ x) \&\& } (f \texttt{ x)})$$
$$\quad\quad\quad (\lambda\texttt{. false})$$

$$\mathit{compile}(f, n : S_1 \mapsto S_2) \stackrel{\text{def}}{=}$$
$$\quad \texttt{pair } (\lambda\texttt{x. false})$$
$$\quad\quad\quad (\lambda\texttt{. } (f \ (\lambda\texttt{x.x})) \texttt{ \&\& (pair } \mathit{compile}(\texttt{call}_n, S_1)$$
$$\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad \mathit{compile}(\texttt{ret}_n, \ S_2)))$$

$$\mathit{compile}(f, y) \stackrel{\text{def}}{=}$$
$$\quad \texttt{let (chkconst,chkfn) = } y$$
$$\quad \texttt{pair } (\lambda\texttt{x. } (f \texttt{ x)} \quad \texttt{\&\& (chkconst x))}$$
$$\quad\quad\quad (\lambda\texttt{. } (f \ (\lambda\texttt{x.x})) \texttt{ \&\& (chkfn()))}$$

$$
\begin{array}{lcl}
\texttt{let s} & = & s_0 \\
\quad \texttt{call}_n & = & \lambda i. \cdots \text{check and update } \texttt{s} \text{ appropriately } \cdots \\
\quad \ldots & & \ldots \\
\quad \texttt{ret}_n & = & \lambda o. \cdots \text{check and update } \texttt{s} \text{ appropriately } \cdots \\
\quad \ldots & & \ldots \\
\multicolumn{3}{l}{\texttt{in } \mathit{compile}(\lambda \texttt{x.true}, S)}
\end{array}
$$

$$\mathit{compile} : (\mathit{Constant} \cup \{\lambda \texttt{x.x}\} \mapsto \mathit{Bool}) \times S \mapsto \mathit{Monitor}$$

$$\mathit{compile}(f, \mathit{flat}(e)) \stackrel{\text{def}}{=}$$

```
pair (λx. (e x) && (f x))
     (λ.  false)
```

$$\mathit{compile}(f, n : S_1 \mapsto S_2) \stackrel{\text{def}}{=}$$

```
pair (λx. false)
     (λ.  (f (λx.x)) && (pair compile(call_n, S_1)
                              compile(ret_n,  S_2))))
```

$$\mathit{compile}(f, y) \stackrel{\text{def}}{=}$$

```
let (chkconst,chkfn) = y
pair (λx. (f x)        && (chkconst x))
     (λ.  (f (λx.x)) && (chkfn()))
```

$$
\begin{array}{lll}
\texttt{let s} & = & s_0 \\
\quad \texttt{call}_n & = & \lambda i. \cdots \text{check and update } \texttt{s} \text{ appropriately } \cdots \\
\quad \ldots & & \ldots \\
\quad \texttt{ret}_n & = & \lambda o. \cdots \text{check and update } \texttt{s} \text{ appropriately } \cdots \\
\quad \ldots & & \ldots \\
\multicolumn{3}{l}{\texttt{in } \mathit{compile}(\lambda \texttt{x.true}, S)}
\end{array}
$$

$$\mathit{compile} : (\mathit{Constant} \cup \{\lambda\texttt{x.x}\} \mapsto \mathit{Bool}) \times S \mapsto \mathit{Monitor}$$

$$
\begin{array}{l}
\mathit{compile}(f, \mathit{flat}(e)) \stackrel{\texttt{def}}{=} \\
\quad \texttt{pair } (\lambda\texttt{x. } (e\ \texttt{x}) \texttt{ \&\& } (f\ \texttt{x})) \\
\qquad\quad (\lambda\texttt{. false}) \\
\mathit{compile}(f, n{:}S_1 \mapsto S_2) \stackrel{\texttt{def}}{=} \\
\quad \texttt{pair } (\lambda\texttt{x. false}) \\
\qquad\quad (\lambda\texttt{. } (f\ (\lambda\texttt{x.x})) \texttt{ \&\& } (\texttt{pair } \mathit{compile}(\texttt{call}_n, S_1) \\
\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \mathit{compile}(\texttt{ret}_n, S_2)))) \\
\mathit{compile}(f, y) \stackrel{\texttt{def}}{=} \\
\quad \texttt{let (chkconst,chkfn) = } y \\
\quad \texttt{pair } (\lambda\texttt{x. } (f\ \texttt{x}) \texttt{ \&\& (chkconst x))} \\
\qquad\quad (\lambda\texttt{. } (f\ (\lambda\texttt{x.x})) \texttt{ \&\& (chkfn()))}
\end{array}
$$

Step 5: Racket Implementation

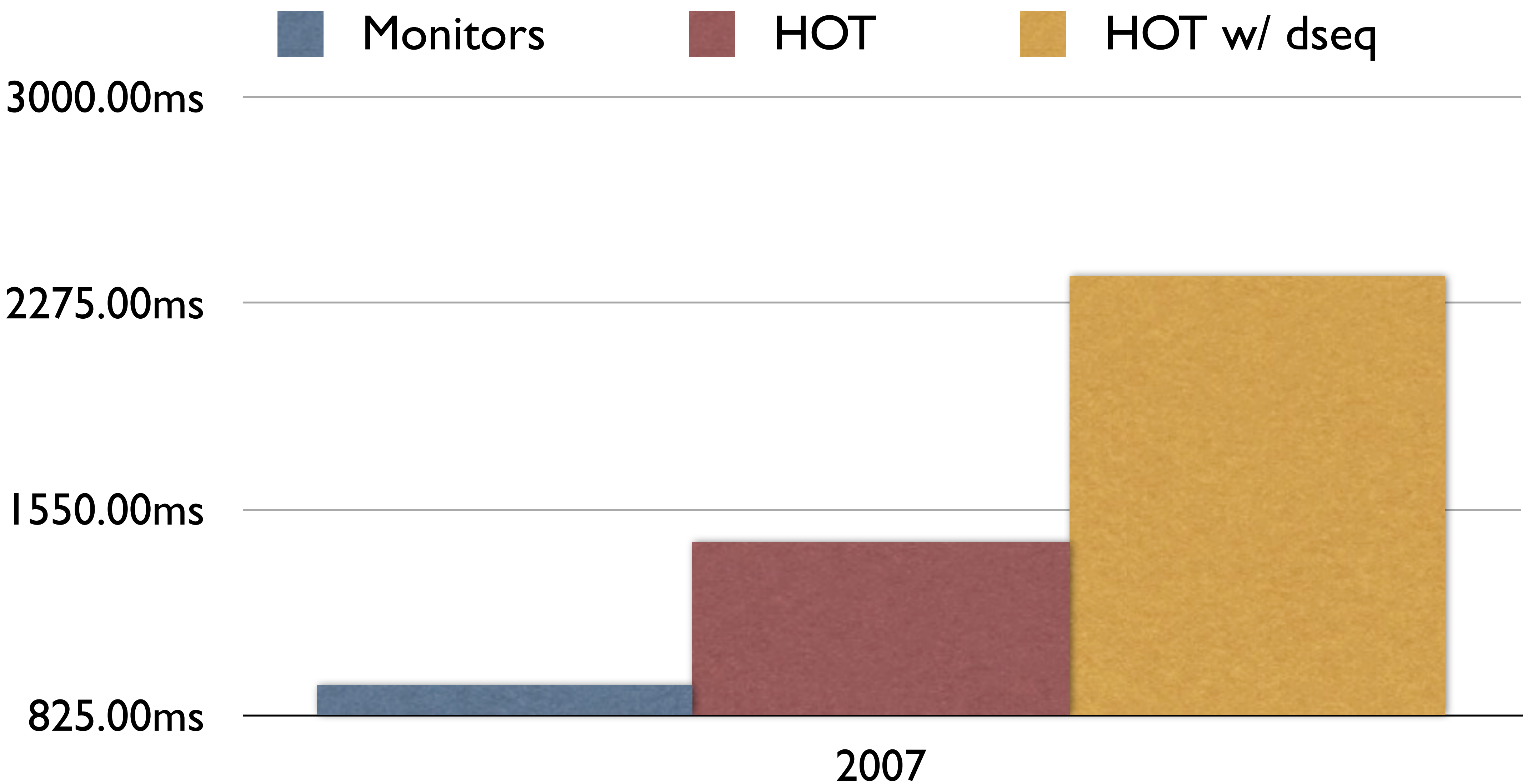
Differences

- No non-interference guarantee (inherited from Racket contracts)
- Continuations allow 0, 1, and many returns
- Mutation intermediation through standard contracts
- Safe w.r.t concurrency with kill-safe server
- Temporal formula macros like **and** using De Morgan's law
- Recursive temporal formulas with delays

Step 6: Evaluation

Racket Standard Library

Atomic	519	number?
Transient	51	map
Anti-transient	17	curry
Unconstrained	13	apply


Monitors
HOT
HOT w/ dseq
3000.00ms
2275.00ms
1550.00ms
825.00ms
2007

Atomicity

Adversarial Defense

- Tic-Tac-Toe game
- Player : Board → Board
- May only call board-set once
- May not call board-set with same arguments in one game
- Catches cheating humans and AIs with contracts

Future Work

- Interaction with types
- Concurrency
- Explain contract violations