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 | 2 3)

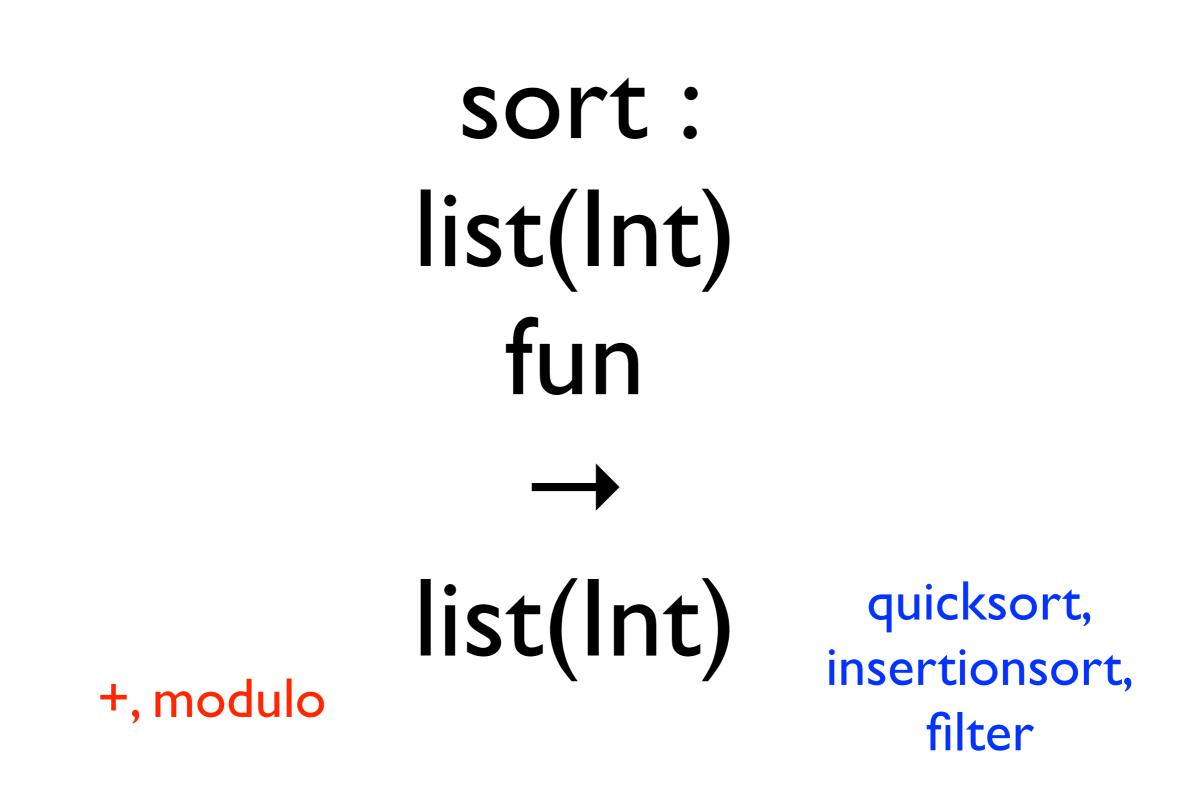
sort : fun

4, true, (list | 2 3) quicksort, insertionsort, +, modulo

sort: list(Int) fun list(Int)

sort: list(Int) fun list(Int)

+, modulo



sort: list(Int) (Int Int \rightarrow Bool) list(Int)

sort: list(Int) (Int Int \rightarrow Bool) list(Int)

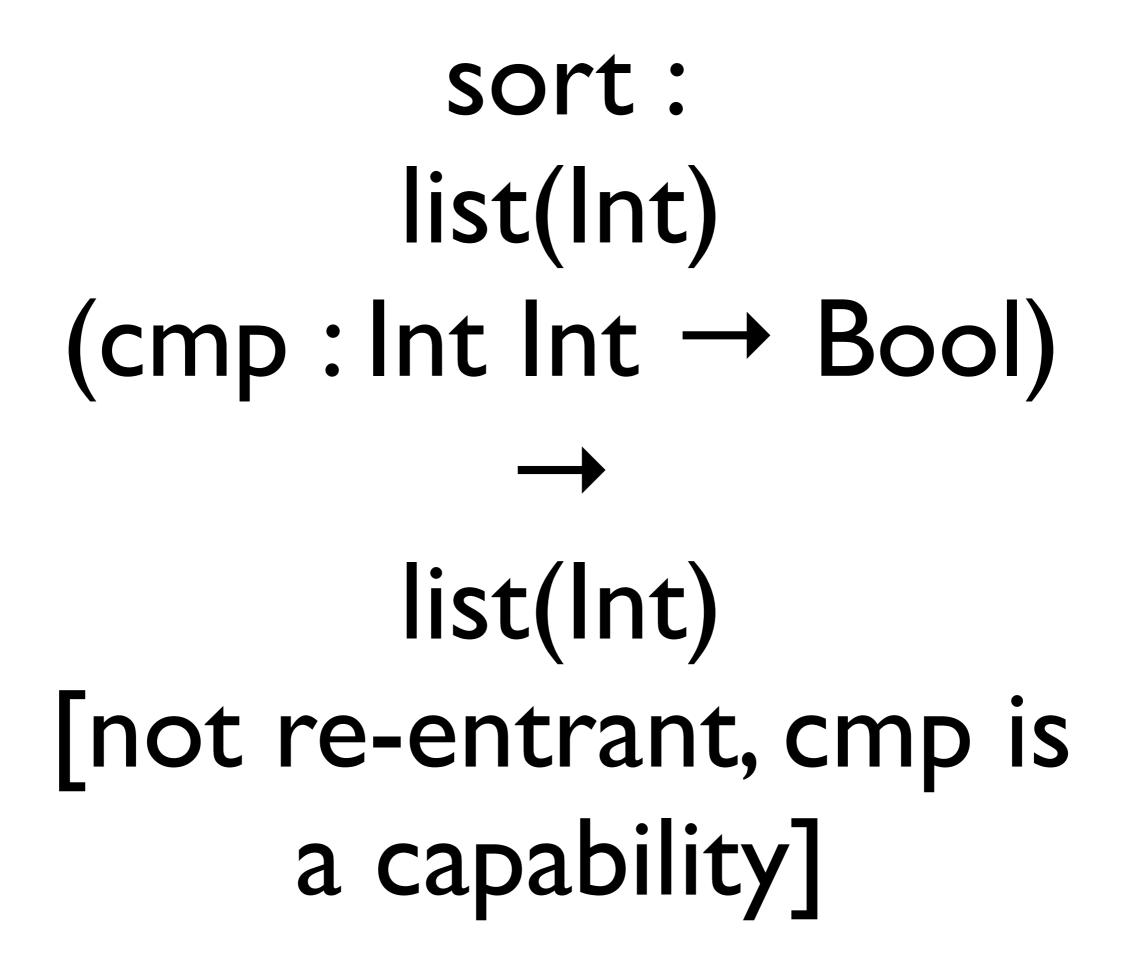
filter, map

sort: list(Int) (Int Int \rightarrow Bool) list(Int) quicksort, filter, map insertionsort

sort: list(Int) (Int Int \rightarrow Bool) list(Int) [not re-entrant]

(define (cmp x y)
 (f x y
 (sort m <=)))</pre>

(sort l cmp)

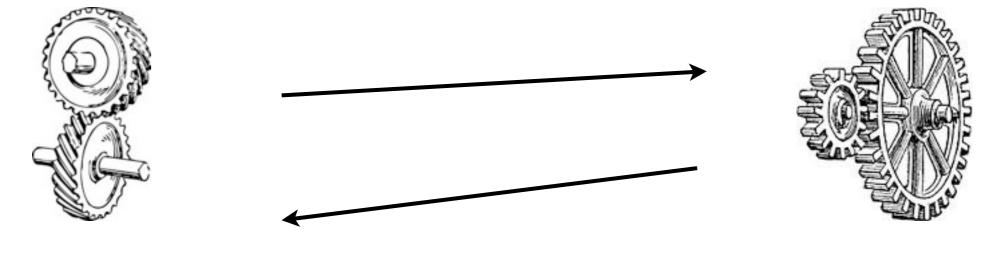


(define last-cmp #f) (define (partner l cmp) (set! last-cmp cmp) (quicksort l cmp)) (define (thief x y) (last-cmp x y))

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

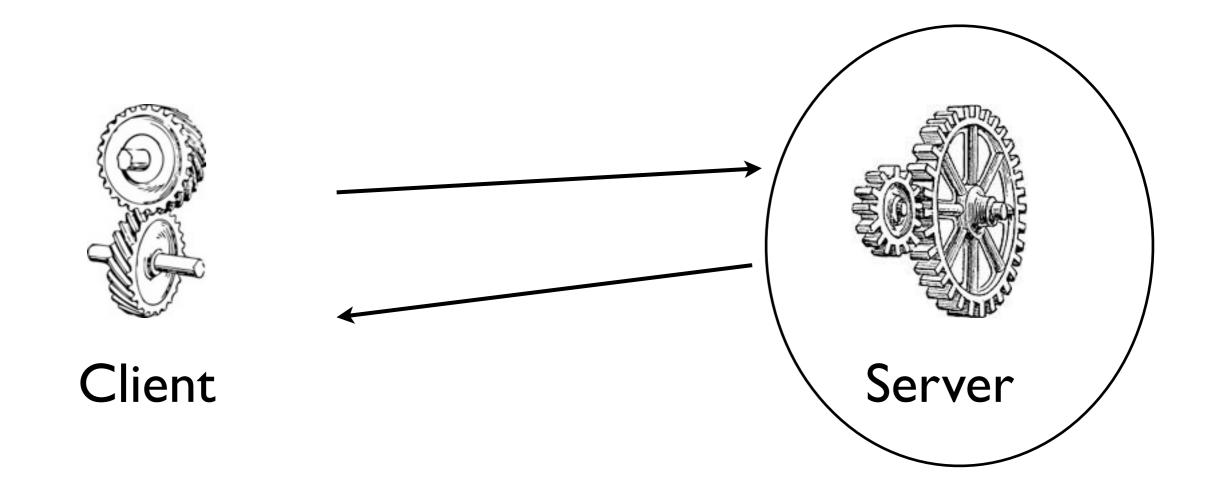
```
SortContract =
 sort : (List Pos)
           (cmp : Pos \rightarrow Pos \rightarrow Bool)
       \rightarrow (List Pos)
  where
     not ... call(sort, ) !ret(sort, )*
          call(sort, )
  and
```

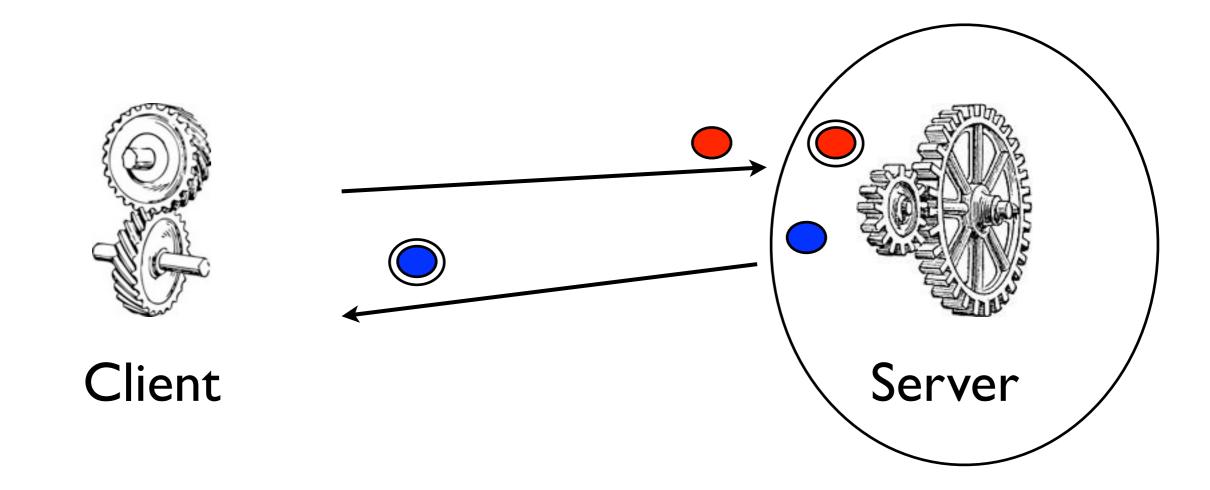
```
not ... ret(sort,_) ... call(cmp,_)
```

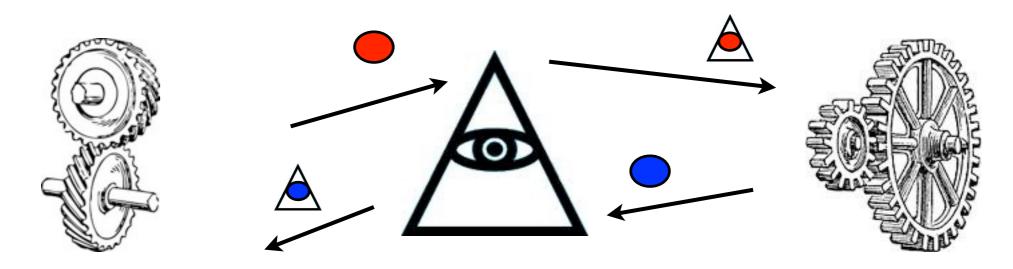


Client

Server







Client

Server

Step I: Game Semantics

Domains State Code Store Interface	CSI C S I	\in	$Code \times Stor$ $\mathcal{E}[e] \mid \mathcal{Q}[\perp]$ $\mathcal{P}(Var \times Var)$ $Var \rightarrow Value$	$ar \times Value)$		Evaluation con Quiescent cont Value Handle Event Direction Trace		$\begin{array}{ccc} \mathcal{Q} & :: \\ v & :: \\ h & :: \\ a & :: \\ \rho & :: \end{array}$	$ \begin{array}{c} := & \bullet \\ := & c \\ := & c \\ := & \rho. \end{array} $	$ \begin{array}{c c} & \mathcal{E}[\texttt{send.} \\ & x \mid \lambda y. e \\ & x \\ \texttt{ret}(x, h) \\ \texttt{end} \mid \texttt{rcv} \end{array} $	<u> </u>	$ \mathcal{Q}[\texttt{rcv.call}_x \bullet]$
$\mathcal{E}[(\lambda x. e)]$ $\langle \mathcal{E}[c v],$ $\langle \mathcal{E}[c v],$ $\langle \mathcal{E}[ref v],$ $\langle \mathcal{E}[x v],$ $\langle \mathcal{E}[y v],$	v],],	S[(x,y)]	$\begin{array}{ll} State \times Eve \\ S, & I \\ S, & I \\ S, & I \\ O \mapsto v'], I \\ O \mapsto v'], I \end{array}$	$ \begin{array}{l} \underset{\leftarrow}{\rightarrow} \\ \rightarrow \\ \rightarrow \\ \rightarrow \\ \rightarrow \\ \rightarrow \\ \rightarrow \end{array} $		$\begin{array}{l} \mathcal{E}[e[x:=v]],\\ \mathcal{E}[v'],\\ \mathcal{E}[\texttt{pair}\;x\;y],\\ \mathcal{E}[v'],\\ \mathcal{E}[v], \end{array}$	S[(x,	$y) \mapsto v$	v], v'], v'], v'], v'], v'], v'], v'], v	$egin{array}{ccc} I & ight angle \\ I & ight angle \end{array}$	$v' = \delta(c, v)$ x, y fresh	[CALL] [PRIM] [REF] [GET] [SET]
$egin{array}{c} & \mathcal{E}[xv], \ & \langle \mathcal{E}[t{send.call}, \ & \langle \mathcal{Q}[\bot], \ & \langle \mathcal{Q}[t{rcv.call}] \end{array} \end{array}$	$\lfloor_x \perp],$		$egin{array}{ccc} S, & & I \ \end{array}$	$ \rightarrow \frac{\operatorname{send.call}(x,h)}{\rightarrow} \operatorname{rcv.ret}(x,h) \\ \rightarrow \operatorname{rcv.call}(x,h) \\ \rightarrow \operatorname{send.ret}(x,h) $	\langle	$\mathcal{E}[\texttt{send.call}_x \perp], \ \mathcal{E}[h], \ [\texttt{rcv.call}_x (v \ h)], \ \mathcal{Q}[\perp],$,	S, S, S, S,		$ert v] angle \ I \ angle \ I \ angle \ v] angle \ ert v] angle$	$x \not\in BV(S)$ $I(x) = v$	[SEND-CALL] [RCV-RET] [RCV-CALL] [SEND-RET]

Domains State Code Store Interface	CSI C S I	::= €	$\mathcal{E}[e] \mid \mathcal{Q}$ $\mathcal{P}(Var)$	$2[\perp] \times Var \times$	Interface Value)		Evaluation con Quiescent cont Value Handle Event Direction Trace		$\begin{array}{c} \mathcal{Q} & : \\ v & : \\ h & : \\ a & : \\ \rho & : \end{array}$:= := := :=	• $\hat{\mathcal{E}}[\text{send.} c \mid x \mid \lambda y. e]$ $c \mid x$		$\mid \mathcal{Q}[\texttt{rcv.call}_x \bullet]$
$\mathcal{E}[(\lambda x. e)]$ $\langle \mathcal{E}[c v]]$ $\langle \mathcal{E}[c v]]$ $\langle \mathcal{E}[ref v]]$ $\langle \mathcal{E}[x v]]$ $\langle \mathcal{E}[y v]]$	v],	S[(x,y)]	State \times S, S, S, S, S, $D \mapsto v'],$ $D \mapsto v'],$	$\begin{array}{ccc} I \rangle & \rightarrow \\ I \rangle & \rightarrow \\ I \rangle & \rightarrow \\ I \rangle & \rightarrow \end{array}$	_ × State		$\mathcal{E}[v'],$	S[(x, S)]))]))])])])])])])])]))]))))))))))))	$y) \mapsto c$	v'],	$egin{array}{ccc} I & ight angle \ I & ight $	$v' = \delta(c, v)$ x, y fresh	[CALL] [PRIM] [REF] [GET] [SET]
$egin{array}{ccc} & \mathcal{E}[x v] \ & \langle \mathcal{E}[extsf{send.call}, \ & & \mathcal{Q}[ot], \ & \langle & \mathcal{Q}[ot], \ & \langle & \mathcal{Q}[ot] \end{array}$	$\lfloor x \perp],$		5, 5,	$\left. \begin{array}{c} I \\ I \end{array} \right\rangle {\rightarrow}{}^{r} \\ I \\ I \end{array} \right\rangle {\rightarrow}{}^{r}$	$\begin{aligned} \texttt{end.call}(x,h) \\ \texttt{cv.ret}(x,h) \\ \texttt{cv.call}(x,h) \\ \texttt{end.ret}(x,h) \end{aligned}$	<	$\mathcal{E}[extsf{send.call}_x ot], \ \mathcal{E}[h], \ \mathcal{C}[extsf{rcv.call}_x (v \ h)], \ \mathcal{Q}[ot],$		$egin{array}{l} S,\ S,\ S,\ S,\end{array}$		$egin{array}{ccc} I[h arphi v] & \ I & $	$x \not\in BV(S)$ I(x) = v	[SEND-CALL] [RCV-RET] [RCV-CALL] [SEND-RET]

Domains State CSI	\in Cod	$e \times Store \times Ia$	nterface	Evaluation con Quiescent cont			$\mathcal{E}[\bullet e] \mid \mathcal{E}[\bullet e]$		$\mid \mathcal{Q}[\texttt{rcv.call}_x ullet$
CodeCStoreSInterfaceI	$\begin{array}{ll} ::= & \mathcal{E}[e] \\ \in & \mathcal{P}(V) \\ \in & Var \end{array}$	$Var \times Var \times V$	Value)	Value Handle Event Direction Trace	v	$\begin{array}{ccc} \vdots & \vdots \\ \vdots & \vdots \\ \vdots & \vdots \\ \vdots & \vdots \end{array}$	$x \mid \lambda y. e$ $x \mid x$ b.ret(x, h) send rcv	-	
Transition relation (–	$\Rightarrow) \subseteq Stat$	$te \times Event_{\perp} >$	< State						
$ \begin{array}{ll} \langle & \mathcal{E}[(\lambda x. e) \ v], \\ \langle & \mathcal{E}[c \ v], \\ \langle & \mathcal{E}[ref \ v], \\ \langle & \mathcal{E}[x \ v], \\ \langle & \mathcal{E}[y \ v], \\ \rangle \end{array} \right. $		_ · · ·		$\mathcal{E}[v'],$	$S,$ $S,$ $S[(x,y) \vdash S](x,y) \vdash S$ $S[(x,y) \vdash S](x,y) \vdash S$	$\rightarrow v'],$		$v' = \delta(c, v)$ x, y fresh	[CALL] [PRIM] [REF] [GET] [SET]
$ \langle \begin{array}{c} \mathcal{E}[x \ v], \\ \langle \mathcal{E}[\texttt{send.call}_x \perp], \end{array} $	S, S, S,	± / '	$(1. call(x,h)) \langle \mathcal{E} $ $ret(x,h) \langle \mathcal{E} $	$\mathcal{E}[\texttt{send.call}_x \perp], \ \mathcal{E}[h],$	S, S, S,	I[$egin{array}{c} h \triangleright v \end{bmatrix} angle \ I & angle \end{array}$	$x \not\in BV(S)$	[SEND-CALL] [RCV-RET]
$\langle \begin{array}{c} \mathcal{Q}[\bot], \\ \langle \begin{array}{c} \mathcal{Q}[\bot], \\ \langle \begin{array}{c} \mathcal{Q}[\texttt{rcv.call}_x v], \end{array} \rangle \\ \rangle$	S, S, S, S,	$I \rangle \rightarrow^{\mathrm{rev}}$		$\mathcal{Q}[\perp],$ $\mathcal{Q}[\perp],$,		$ \begin{array}{c} I \\ I \\ b \triangleright v \end{array} \right) $	I(x) = v	[RCV-CALL] [SEND-RET]

DomainsState $CSI \in Code \times Store \times Interface$ $Code$ $C ::= \mathcal{E}[e] \mid \mathcal{Q}[\bot]$ Store $S \in \mathcal{P}(Var \times Var \times Value)$ Interface $I \in Var \rightarrow Value$	Quiescent context Value Handle Event Direction	$\begin{array}{llllllllllllllllllllllllllllllllllll$	$\operatorname{call}_x \bullet]$	$\mathcal{Q}[\texttt{rcv.call}_x \bullet]$
$\begin{array}{llllllllllllllllllllllllllllllllllll$	$ \begin{array}{ccc} & & \mathcal{E}[\texttt{pair}xy], & & S[(x,y)] \\ & & & \mathcal{E}[v'], & & S[(x,y)] \end{array} \end{array} $	$egin{array}{cccc} S, & I & \ S, & I & \ y) \mapsto v], & I & \ y) \mapsto v'], & I & \ y) \mapsto v'], & I & \ y) \mapsto v], & I & \ angle$	$v' = \delta(c, v)$ x, y fresh	[CALL] [PRIM] [REF] [GET] [SET]
$ \begin{array}{cccc} \langle & \mathcal{E}[x v], & S, & I \rangle & \rightarrow^{\texttt{send.call}(x,h)} \\ \langle \mathcal{E}[\texttt{send.call}_x \perp], & S, & I \rangle & \rightarrow^{\texttt{rcv.ret}(x,h)} \\ \langle & \mathcal{Q}[\perp], & S, & I \rangle & \rightarrow^{\texttt{rcv.call}(x,h)} \\ \langle & \mathcal{Q}[\texttt{rcv.call}_x v], & S, & I \rangle & \rightarrow^{\texttt{send.ret}(x,h)} \end{array} $	$\langle \mathcal{E}[h], \\ \langle \mathcal{Q}[\texttt{rcv.call}_x (v \ h)], \rangle$	$\begin{array}{ll} S, & I[h \triangleright v] \rangle \\ S, & I & \rangle \\ S, & I & \rangle \\ S, & I[h \triangleright v] \rangle \end{array}$	$x \not\in BV(S)$ I(x) = v	[SEND-CALL] [RCV-RET] [RCV-CALL] [SEND-RET]

Domains State	CSI	E	$Code \times Sto$	ore \times Interface		Evaluation con Quiescent cont			$\mathcal{E}[\bullet e] \mid \mathcal{E}[e] \\ \bullet \mid \mathcal{E}[e]$		$\mid \mathcal{Q}[\texttt{rcv.call}_x \bullet]$
$Code \\ Store$	$C \\ S$		$\begin{array}{c c} \mathcal{E}[e] & \mathcal{Q}[\bot \\ \mathcal{P}(Var \times V \end{array} \end{array}$] Var × Value)		Value Handle		v ::= h ::=	$\begin{array}{rcl} c & x & \lambda y. e \\ c & x \end{array}$		
Interface	Ι	E	$Var \rightarrow Val$	ue		Event Direction Trace		ρ ::=	$ \begin{array}{l} \rho.\texttt{ret}(x,h) \\ \texttt{send} \mid \texttt{rcv} \\ \texttt{s} \end{array} $	$\mid \rho.\texttt{call}(x,h)$	
Transition relat	tion (-	$\rightarrow)$ \subseteq	$State \times Ei$	$vent_{\perp} \times State$							
$\begin{array}{ccc} \langle & \mathcal{E}[(\lambda x. e) \\ \langle & \mathcal{E}[c v] \\ \langle & \mathcal{E}[\texttt{ref } v] \\ \langle & \mathcal{E}[x v] \\ \langle & \mathcal{E}[y v] \end{array}$, , ,	S[(x,y)]	$egin{array}{llllllllllllllllllllllllllllllllllll$	$\begin{array}{c} \rightarrow \\ \rightarrow \\ \rightarrow \\ \rightarrow \\ \rightarrow \\ \rightarrow \end{array}$		$\mathcal{E}[v'],$	S[(x, y)]	$v) \mapsto v'$	$, I angle \ , I angle$	$v' = \delta(c, v)$ x, y fresh	[CALL] [PRIM] [REF] [GET] [SET]
$\langle \mathcal{E}[x v] \rangle$			$S, I\rangle$	\rightarrow send.call (x,h) \rightarrow rcv.ret (x,h)	ζ ε	$C[\texttt{send.call}_x \perp],$		S,	$I[h \triangleright v] \rangle$	$x \not\in \mathit{BV}(S)$	[SEND-CALL]
$egin{array}{llllllllllllllllllllllllllllllllllll$,		,	$\rightarrow^{\operatorname{rcv.call}(x,h)}$ $\rightarrow^{\operatorname{send.ret}(x,h)}$	$\langle \mathcal{Q} \\ \langle \mathcal{Q} \\ \langle$	$\mathcal{E}[h], \ [\texttt{rcv.call}_x \ (v \ h)], \ \mathcal{Q}[\perp],$,	S, S, S, S, S,	$egin{array}{cc} I & angle \ I & angle \ I & angle \ I \left[h arphi v ight] angle \end{array}$	I(x) = v	[RCV-RET] [RCV-CALL] [SEND-RET]

\in Cooler $\mathcal{E}[e]$	le imes Store :	\times Interface	Evalu					
	$le \times Store$:	\times Interface	Evalu					
$::= \mathcal{E}[e]$				ation context				$\mid \mathcal{Q}[\texttt{rcv.call}_x \bullet]$
$::= \mathcal{E}[e]$			•	scent context	-	$= \bullet \mathcal{E}[\text{send}]$	-	
		$(\mathbf{T}_{\mathbf{T}})$	Value			$= c \mid x \mid \lambda y. e$		
	$Var \times Var$	\times Value)	Hand			= c x	(1 - 1)	1
\in Var	\rightarrow value					•		
					,			
			11400		ι	_ u		
,		$_{\perp} \times State$						
S,	$I \rangle \rightarrow$		$\langle \mathcal{E}[e x]$:= v]],	S,	$I \rightarrow$		[CALL]
S,	$I \rangle \rightarrow$		$\langle \mathcal{E}[u]$	v'],	S,	I \rangle	$v' = \delta(c, v)$	[PRIM]
S,	$I \rangle \rightarrow$		\ L=	$\mathbf{r} x y$], $S[(x,$	$,y)\mapsto y$	$v], I \rangle$	x, y fresh	[REF]
$S[(x,y) \mapsto$	$v'], I \rangle \rightarrow$		•	2 · · · · · · · · · · · · · · · · · · ·	- /	1		[GET]
$S[(x,y)\mapsto$	$v'], I \rangle \rightarrow$		$\langle \mathcal{E} $	$v], \qquad S[(x,$	$(y)\mapsto y$	v], I >		[SET]
S,	$I \rangle \rightarrow$	send.call(x,h)	$\langle \ {\cal E}[{\tt send.c}]$	$[all_x \perp],$	S,	$I[h \triangleright v] \rangle$	$x \not\in BV(S)$	[SEND-CALL]
S,	$I \rangle \rightarrow$	rcv.ret(x,h)	$\langle \mathcal{E}[$	h],	S,	$I \rangle$		[RCV-RET]
	$I \rangle \rightarrow$	rcv.call(x,h)	· -	-		$I \rightarrow$	I(x) = v	[RCV-CALL]
S,	$I \rightarrow$	send.ret(x,h)	· -		S,	$I[h \triangleright v]$		[SEND-RET]
	$ \begin{array}{rcl} \rightarrow) & \subseteq & Sta \\ & S, \\ & S, \\ & S, \\ S[(x,y) \mapsto \\ S[(x,y) \mapsto \\ & S, \end{array} $	$ \begin{array}{rcl} \rightarrow) & \subseteq & State \times Event \\ S, & I \rangle & \rightarrow \\ S, & I \rangle & \rightarrow \\ S, & I \rangle & \rightarrow \\ S[(x,y) \mapsto v'], $	$ \begin{array}{rcl} \rightarrow &) & \subseteq & State \times Event_{\perp} \times State \\ & & S, & I \\ & & S[(x,y) \mapsto v'], I \\ & & \rightarrow \\ & S[(x,y) \mapsto v'], I \\ & & \rightarrow \\ & S[(x,y) \mapsto v'], I \\ & & \rightarrow \\ & S[(x,y) \mapsto v'], I \\ & & \rightarrow \\ & S[(x,y) \mapsto v'], I \\ & & & \rightarrow \\ & S[(x,y) \mapsto v'], I \\ & & & \rightarrow \\ & S[(x,y) \mapsto v'], I \\ & & & \rightarrow \\ & S[(x,y) \mapsto v'], I \\ & & & & & \rightarrow \\ & S[(x,y) \mapsto v'], I \\ & & & & & \rightarrow \\ & S[(x,y) \mapsto v'], I \\ & & & & & \rightarrow \\ & S[(x,y) \mapsto v'], I \\ & & & & & & \rightarrow \\ & S[(x,y) \mapsto v'], I \\ & & & & & & & \rightarrow \\ & S[(x,y) \mapsto v'], I \\ & & & & & & & & & & & & \\ & S[(x,y) \mapsto v'], I \\ & & & & & & & & & & & & & & \\ & S[(x,y) \mapsto v'], I \\ & & & & & & & & & & & & & & & & & &$	$\begin{array}{rcl} Direc \\ Trace \\ \end{array} \\ \rightarrow) &\subseteq \ State \times Event_{\perp} \times State \\ & S, & I \\ S, & I \\ S, & I \\ \end{array} \\ \rightarrow & \langle & \mathcal{E}[e]x \\ & \mathcal{E}[a] \\ & \mathcal{E}[paix \\ S[(x,y) \mapsto v'], I \\ \rightarrow & \langle & \mathcal{E}[a] \\ & S[(x,y) \mapsto v'], I \\ \rightarrow & \langle & \mathcal{E}[a] \\ & S[(x,y) \mapsto v'], I \\ \rightarrow & \langle & \mathcal{E}[a] \\ & \mathcal{E}[a] \\ & S[(x,y) \mapsto v'], I \\ \rightarrow & \langle & \mathcal{E}[a] \\ & \mathcal{E}[a]$	$\begin{array}{rcl} Direction \\ Trace \\ \end{array} \\ \rightarrow) &\subseteq \ State \times Event_{\perp} \times State \\ \\ \begin{array}{rcl} S, & I \\ S, & I \\ S, & I \\ \end{array} \\ \rightarrow & & \langle & \mathcal{E}[e[x := v]], \\ & \mathcal{E}[v'], \\ & \mathcal{E}[v'], \\ & \mathcal{E}[pair \ x \ y], \\ S[(x, y) \mapsto v'], I \\ \end{array} \\ \rightarrow & & \langle & \mathcal{E}[v'], \\ S[(x, y) \mapsto v'], I \\ \end{array} \\ \begin{array}{rcl} S, & I \\ S, & I \\ \end{array} \\ \begin{array}{rcl} S, & I \\ S, & I \\ \end{array} \\ \begin{array}{rcl} S, & I \\ S, & I \\ \end{array} \\ \begin{array}{rcl} S, & I \\ S, & I \\ \end{array} \\ \begin{array}{rcl} S, & I \\ \end{array} \\ \end{array} \\ \begin{array}{rcl} S, & I \\ \end{array} \\ \end{array} \\ \begin{array}{rcl} S, & I \\ \end{array} \\ \end{array} \\ \begin{array}{rcl} S, & I \\ \end{array} \\ \begin{array}{rcl} S, & I \\ \end{array} \\ \end{array} \\ \begin{array}{rcl} S, & I \\ \end{array} \\ \end{array} $ \\ \begin{array}{rcl} S, & I \\ \end{array} \\ \begin{array}{rcl} S, & I \\ \end{array} \\ \end{array} \\ \begin{array}{rc	$\begin{array}{rcl} Direction & \rho & ::\\ Trace & t & ::\\ \end{array} \\ \rightarrow & \subseteq \ State \times Event_{\perp} \times State \\ \\ \begin{array}{rcl} S, & I \\ S, & I \\ S, & I \\ \end{array} \\ \rightarrow & \\ S[(x,y) \mapsto v'], I \\ \rightarrow & \\ S[(x,y) \mapsto v'], I \\ \end{array} \\ \rightarrow & \\ S[(x,y) \mapsto v'], I \\ \end{array} \\ \rightarrow & \\ \begin{array}{rcl} S[(x,y) \mapsto v'], I \\ S[(x,y) \mapsto v'], I \\ \end{array} \\ \rightarrow & \\ \begin{array}{rcl} S[(x,y) \mapsto v'], I \\ S[(x,y) \mapsto v'], I \\ \end{array} \\ \rightarrow & \\ \begin{array}{rcl} S[(x,y) \mapsto v'], I \\ S[(x,y) \mapsto v'], I \\ \end{array} \\ \rightarrow & \\ \begin{array}{rcl} S[(x,y) \mapsto v'], I \\ S[(x,y) \mapsto v'], I \\ \end{array} \\ \rightarrow & \\ \begin{array}{rcl} S[(x,y) \mapsto v'], I \\ \end{array} \\ \begin{array}{rcl} S[(x,y) \mapsto v'], $	$\begin{array}{cccccccccccccccccccccccccccccccccccc$	$\begin{array}{cccccccccccccccccccccccccccccccccccc$

Domains											
State	CSI	$\in C$	$ode \times Sto$	re imes Interface		Evaluation cont		\mathcal{E} ::=			$\mid \mathcal{Q}[\texttt{rcv.call}_x]$
	~					Quiescent conte		\mathcal{Q} ::=		$\texttt{call}_x \bullet]$	
Code	$C_{\widetilde{C}}$		$e] \mid \mathcal{Q}[\perp]$			Value			$c \mid x \mid \lambda y. e$		
Store	S_{\perp}		`	$ar \times Value)$		Handle		h ::=			
Interface	1	$\in Ve$	$ar \rightarrow Val$	ue		Event			$\rho.\texttt{ret}(x,h)$	$\mid ho.\mathtt{call}(x,h)$	
						Direction		ρ ::=	1		
						Trace		t ::=	\vec{a}		
$ \begin{array}{c} \langle \mathcal{E}[(\lambda x. e) \\ \langle \mathcal{E}[c v] \end{array} \end{array} $	v],	,	$ \begin{array}{c} I \\ I \\ I \\ I \\ I \\ \end{array} $	$\begin{array}{c} ent_{\perp} \times State \\ \rightarrow \\ \rightarrow \end{array}$	< l>	$ \begin{array}{l} \mathcal{E}[e[x:=v]],\\ \mathcal{E}[v'], \end{array} $,	S, S, S,	$egin{array}{cc} I & angle \\ I & angle \end{array}$	$v' = \delta(c, v)$	[CALL] [PRIM]
	_)									
$\langle \mathcal{E}[\texttt{ref } v]$,	S',	$I\rangle$	\rightarrow	$\langle \rangle$			$(v) \mapsto v]_{v}$		x, y fresh	[REF]
$\langle \mathcal{E}[\texttt{ref } v]$, ,	S, $S[(x,y) \mapsto S[(x,y) :$	$\rightarrow v'], I \rangle$	\rightarrow \rightarrow \rightarrow	< < <	$\mathcal{E}[v'],$	S[(x, y	$egin{aligned} & () \mapsto v], \ & () \mapsto v'], \ & () \mapsto v'], \ & () \mapsto v], \end{aligned}$	$, I \rangle$	x, y fresh	[REF] [GET] [SET]
$\begin{array}{ccc} \langle & \mathcal{E}[\texttt{ref } v] \\ \langle & \mathcal{E}[x v] \end{array}$)], , 2	$S[(x,y) \mapsto S[(x,y) \mapsto X]$	$\rightarrow v'], I \rangle$	$ \begin{array}{c} \rightarrow \\ \rightarrow \\ \rightarrow \\ \rightarrow \end{array} \\ \rightarrow \\ \begin{array}{c} \end{array} \\ \end{array} \\ \text{send.call}(x,h) \end{array} $	ζ ζ	$\mathcal{E}[v'],$	$S[(x, y)]{S[(x, y)]}$	$() \mapsto v']$ $() \mapsto v]$	$, I \rangle$	x, y fresh $x \not\in BV(S)$	[GET]
$\begin{array}{ccc} & \mathcal{E}[\texttt{ref} v] \\ & \mathcal{E}[x v] \\ & \mathcal{E}[y v] \\ & \mathcal{E}[x v] \\ & \mathcal{E}[x v] \end{array}$, 2 , 2	$S[(x,y) \mapsto S[(x,y) \mapsto S[(x,y) \mapsto S]$	ightarrow v'], I angle ightarrow v'], I angle ightarrow v'], I angle		ζ ζ	$\mathcal{E}[v'], \ \mathcal{E}[v],$ send.call $_x \perp],$	S[(x, y)] S[(x, y)]	$(v') \mapsto v'$ $(v) \mapsto v$, $(v) \mapsto v$,	$, I \rangle \\ I \rangle$		[GET] [SET]
$\begin{array}{ccc} & \mathcal{E}[\texttt{ref} v] \\ & & \mathcal{E}[x v] \\ & & \mathcal{E}[y v] \end{array}$	[j], , , , , , , , , , , , , , , , , , ,	$S[(x,y) \mapsto S[(x,y) \mapsto X]$	ightarrow v'], I angle ightarrow v'], I angle ightarrow v'], I angle I angle	$\rightarrow^{\texttt{send.call}(x,h)}$	ζ ζ ζ ε[$egin{array}{llllllllllllllllllllllllllllllllllll$	S[(x, y)] S[(x, y)]	$() \mapsto v']$ $() \mapsto v]$	$egin{array}{ccc} & I & \lambda & \ & I & \lambda & \ & I & \lambda & \ & I[h \triangleright v] \lambda \end{array}$		[GET] [SET] [SEND-CALL]

Domains State Code Store Interface		\in	$\mathcal{E}[e] \mid \mathcal{Q}[\bot$	$\dot{Var} imes Value$		Evaluation con Quiescent con Value Handle Event Direction Trace		$\begin{array}{ccc} \mathcal{Q} & : \\ v & : \\ h & : \\ a & : \\ \rho & : \end{array}$	$ \begin{array}{rrrr} := & \bullet \mid \mathcal{E}[set] \\ := & c \mid x \mid \lambda y \\ := & c \mid x \end{array} $	$\begin{array}{l} \text{nd.call}_x \bullet]\\ \cdot e\\ h) \mid \rho.\text{call}(x,h) \end{array}$	$\mid \mathcal{Q}[\texttt{rcv.call}_x \bullet]$
Transition relat $\langle \mathcal{E}[(\lambda x. e)]$ $\langle \mathcal{E}[c v],$ $\langle \mathcal{E}[c v],$ $\langle \mathcal{E}[x v],$ $\langle \mathcal{E}[x v],$ $\langle \mathcal{E}[y v],$	v],],	S[(x,y)]	$\begin{array}{l} State \times E \\ S, & I \\ S, & I \\ S, & I \\ S, & I \\) \mapsto v'], I \\) \mapsto v'], I \end{array}$	$\begin{array}{c} \operatorname{vent}_{\perp} \times \operatorname{Str} \\ \\ \\ \\ \\ \end{array}$	ate (((($\mathcal{E}[v'],$	S[(x,	$y) \mapsto c$	$egin{array}{ccc} v], & I & angle \ v'], & I & angle \end{array}$	$v' = \delta(c, v)$ x, y fresh	[CALL] [PRIM] [REF] [GET] [SET]
$egin{array}{ccc} & \mathcal{E}[x v], \ & \langle \mathcal{E}[t send.call \ & & \mathcal{Q}[\bot], \ & \langle \ \mathcal{Q}[t rcv.call] \end{array}$	$\lfloor_x \perp],$		$\begin{array}{ll}S, & I \rangle\\S, & I \rangle\\S, & I \rangle\\S, & I \rangle\\S, & I \rangle\end{array}$	\rightarrow send.cal \rightarrow rcv.ret(\rightarrow rcv.call \rightarrow send.ret	$egin{array}{ccc} x,h) & \langle & & \ (x,h) & \langle & & \ & \ & & \ &$	$[\texttt{send.call}_x \perp], \ \mathcal{E}[h], \ \texttt{rcv.call}_x (v h)] \ \mathcal{Q}[\perp],$		S, S, S, S, S,	$egin{array}{c} I[h \triangleright v] \ & \ I \ & \ & \ & \ & \ & \ & \ & \ &$	$x \not\in BV(S)$ I(x) = v	[SEND-CALL] [RCV-RET] [RCV-CALL] [SEND-RET]

Domains State Code Store Interface	CSI C S I	::= €	$Code \times Store \\ \mathcal{E}[e] \mid \mathcal{Q}[\bot] \\ \mathcal{P}(Var \times Var) \\ Var \rightarrow Value \\ \end{cases}$, , , , , , , , , , , , , , , , , , ,		Evaluation con Quiescent cont Value Handle Event Direction Trace		$egin{array}{c} \mathcal{Q} & & \ v & \ h & \ a & \ ho \end{array}$::= ::= ::= ::=	• $\mathcal{E}[ser c \mid x \mid \lambda y. c \mid x$	hd.call _x •] e h) ρ .call(x, h)	$\mid \mathcal{Q}[\texttt{rcv.call}_x \bullet]$
	v],],	S S S S	$\begin{array}{ccc} State \times Event \\ f, & I \rangle & \rightarrow \\ f, & I \rangle & \rightarrow \\ f, & I \rangle & \rightarrow \\ h & v'], I \rangle & \rightarrow \\ h & v'], I \rangle & \rightarrow \end{array}$	-	<	$ \begin{array}{l} \mathcal{E}[e[x:=v]],\\ \mathcal{E}[v'],\\ \mathcal{E}[\texttt{pair } x \ y],\\ \mathcal{E}[v'],\\ \mathcal{E}[v'],\\ \mathcal{E}[v], \end{array} $	S[(x,	$y) \mapsto$	$v],\ v'],$	$egin{array}{ccc} I & angle \\ I & angle \\ I & angle \\ I & angle \\ I & angle \end{array}$	$v' = \delta(c, v)$ x, y fresh	[CALL] [PRIM] [REF] [GET] [SET]
$egin{array}{cc} & \mathcal{E}[xv], \ & \langle \mathcal{E}[extsf{send.call} \ & & \mathcal{Q}[\perp], \ & \langle \ & \mathcal{Q}[extsf{rcv.call}] \end{array}$	$[x \perp],$	S S S	$\begin{array}{ccc} , & I \\ , & I \\ , & I \\ , & I \\ \end{array} \rightarrow$	send.call (x,h) rcv.ret (x,h) rcv.call (x,h) send.ret (x,h)	\langle	$\begin{array}{l} \texttt{send.call}_x \perp], \\ \mathcal{E}[h], \\ \texttt{cv.call}_x \ (v \ h)], \\ \mathcal{Q}[\perp], \end{array}$		S, S, S, S,	·	$egin{array}{c l} [h arphi v] \ & \ I & \ & \ I & \ & \ & \ & \ & \ &$	$x \not\in BV(S)$ I(x) = v	[SEND-CALL] [RCV-RET] [RCV-CALL] [SEND-RET]

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

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

Evaluation of $H = (\lambda t. t (\lambda x. x+1) 4)$		Evaluation of twa	$ice = (\lambda f. \lambda x. f (f x))$	
$I_1 = [y \mapsto (\lambda t. t \ (\lambda x. x+1) \ 4)]$		$I_2 = [t \mapsto (\lambda f.$	$\lambda x. f(f x))]$	
$J_1 = [y \mapsto (\lambda t. t (\lambda x. x+1) 4), f \mapsto (\lambda x. x+1)]$		$J_2 = [t \mapsto (\lambda f.$	$\lambda x. f(f x)), g \mapsto (\lambda x. f(f x))]$	
$\langle \texttt{rcv.call}_{start} (\lambda t. t (\lambda x. x+1) 4),$	$\emptyset, \emptyset \rangle \rightarrow^{\texttt{send.ret}(start, y)}$		$\langle \texttt{rcv.call}_{start} (\lambda f. \lambda x. f(f x)) \rangle$), $\emptyset, \emptyset \rangle$
$\langle \perp,$	$\emptyset, I_1 \rangle \rightarrow^{\texttt{rcv.call}(y,t)}$	$\rightarrow^{\texttt{send.ret}(start,t)}$	(\perp, \perp, \perp)	$\emptyset, I_2 \rangle$
$\langle \texttt{rcv.call}_y ((\lambda t. t (\lambda x. x+1) 4) t), \rangle$	$\emptyset, I_1 \rangle \rightarrow$			· · ·
$\langle \texttt{rcv.call}_y ((t (\lambda x. x+1)) 4),$	$\emptyset, I_1 \rangle \rightarrow^{\texttt{send.call}(t,f)}$	$\rightarrow^{\texttt{rcv.call}(t,f)}$	$\langle \texttt{rcv.call}_t ((\lambda f. \lambda x. f (f x)) f$	$), \emptyset, I_2 \rangle$
$\langle \texttt{rcv.call}_y (\texttt{send.call}_t \perp) 4 \rangle,$	$\emptyset, J_1 \rangle$	\rightarrow	$\langle \texttt{rcv.call}_t \ (\lambda x. f \ (f \ x)),$	$\emptyset, I_2 \rangle$
	$\rightarrow^{\texttt{rcv.ret}(t,g)}$	$\rightarrow^{\texttt{send.ret}(t,g)}$	$\langle \perp,$	\emptyset, J_2
$\langle \texttt{rcv.call}_y \ (g \ 4),$	$\emptyset, J_1 \rangle \to^{\texttt{send.call}(g,4)}$	$\rightarrow^{\texttt{rcv.call}(g,4)}$	$\langle \texttt{rcv.call}_g ((\lambda x. f(f x)) 4),$	$\langle 0, J_2 \rangle$
$\langle \texttt{rcv.call}_y \ (\texttt{send.call}_g \perp),$	$\emptyset, J_1 \rangle$	\rightarrow	$\langle \texttt{rcv.call}_g (f (f 4)),$	$\emptyset, J_2 \rangle$
	$\rightarrow^{\texttt{rcv.call}(f,4)}$	$\rightarrow^{\texttt{send.call}(f,4)}$	$\langle \texttt{rcv.call}_g (\texttt{send.call}_f (f \perp)) \rangle$), $\emptyset, J_2 \rangle$
$\langle \texttt{rcv.call}_y (\texttt{send.call}_g (\texttt{rcv.call}_f ((\lambda x.x+1) 4)) \rangle$	$))), \emptyset, J_1 \rangle \rightarrow^2$			· · · ·
$\langle \text{rcv.call}_y \text{ (send.call}_g \text{ (rcv.call}_f 5)), \rangle$	$\emptyset, J_1 \rangle \rightarrow^{\texttt{send.ret}(f,5)}$	$\rightarrow^{\texttt{rcv.ret}(f,5)}$	$\langle \texttt{rcv.call}_g \ (f \ 5),$	$\emptyset, J_2 \rangle$
$\langle \texttt{rcv.call}_y \ (\texttt{send.call}_g \perp),$	$(\emptyset, J_1) \to rcv.call(f, 5)$	$\rightarrow^{\texttt{send.call}(f,5)}$	$\langle \texttt{rcv.call}_g \ (\texttt{send.call}_f \perp),$	$\emptyset, J_2 \rangle$
$\langle \texttt{rcv.call}_y (\texttt{send.call}_g (\texttt{rcv.call}_f ((\lambda x.x+1)5))) \rangle$	$))), \emptyset, J_1 \rangle \rightarrow^2$			
$\langle \texttt{rcv.call}_y (\texttt{send.call}_g (\texttt{rcv.call}_f 6)),$	$\emptyset, J_1 \rangle \rightarrow^{\texttt{send.ret}(f,6)}$	$\rightarrow^{\texttt{rcv.ret}(f,6)}$	$\langle \texttt{rcv.call}_g 6,$	$\emptyset, J_2 \rangle$
$\langle \texttt{rcv.call}_y \ (\texttt{send.call}_g \perp),$	$(\emptyset, J_1) \to rcv.ret(g, 6)$	$\rightarrow^{\texttt{send.ret}(g,6)}$	$\langle \perp,$	$\emptyset, J_2 \rangle$
$\langle \texttt{rcv.call}_y 6,$	$\emptyset, J_1 \rangle \rightarrow^{\texttt{send.ret}(y,6)}$			
$\langle \perp,$	$\emptyset, J_1 \rangle$			
		•		

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

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

Evaluation of $H = (\lambda t. t (\lambda x. x+1) 4)$		Evaluation of tw	$ice = (\lambda f. \lambda x. f (f x))$	
$I_1 = [y \mapsto (\lambda t. t \ (\lambda x. x+1) \ 4)]$		$I_2 = [t \mapsto (\lambda f.$	$\lambda x. f(f x))]$	
$J_1 = \begin{bmatrix} y \\ y \\ \end{pmatrix} (\lambda t. t (\lambda x. x+1) 4), f \\ \mapsto (\lambda x. x+1) \end{bmatrix}$		E (-	$\lambda x. f(fx)), g \mapsto (\lambda x. f(fx))$	
$\langle \texttt{rcv.call}_{start} (\lambda t. t (\lambda x. x+1) 4),$	$\emptyset, \emptyset \rightarrow \Rightarrow (start, y)$		$\langle \texttt{rcv.call}_{start} (\lambda f. \lambda x. f(f x)) \rangle$	$\langle \emptyset, \emptyset \rangle$
$\langle \perp,$	$\emptyset, I_1 \rangle \rightarrow^{\texttt{rcv.call}(y,t)}$	$\rightarrow^{\texttt{send.ret}(start,t)}$	$() (\perp,$	$\emptyset, I_2 \rangle$
$\langle \texttt{rcv.call}_{y} ((\lambda t. t (\lambda x. x+1) 4) t),$	$\emptyset, I_1 \rangle \rightarrow$			
$\langle \text{rcv.call}_y ((t (\lambda x. x+1)) 4), \rangle$	$\emptyset, I_1 \rangle \rightarrow^{\texttt{send.call}(t,f)}$	$\rightarrow^{\texttt{rcv.call}(t,f)}$	$\langle \texttt{rcv.call}_t ((\lambda f. \lambda x. f (f x)) f$	$), \emptyset, I_2 \rangle$
$\langle \texttt{rcv.call}_y (\texttt{send.call}_t \perp) 4 \rangle,$	\emptyset, J_1	\rightarrow	$\langle \texttt{rcv.call}_t (\lambda x. f(f x)), \rangle$	$\langle \emptyset, I_2 \rangle$
	$\rightarrow^{\texttt{rcv.ret}(t,g)}$	$\rightarrow^{\texttt{send.ret}(t,g)}$	$\langle \perp,$	$\emptyset, J_2 \rangle$
$\langle \texttt{rcv.call}_y (g 4),$	$\emptyset, J_1 \rangle \to^{\texttt{send.call}(g,4)}$	$\rightarrow^{\texttt{rcv.call}(g,4)}$	$\langle \texttt{rcv.call}_{g} ((\lambda x. f(f x)) 4),$	$\emptyset, J_2 \rangle$
$\langle \texttt{rcv.call}_y \ (\texttt{send.call}_g \perp),$	$\emptyset, J_1 \rangle$	\rightarrow	$\langle \texttt{rcv.call}_g (f (f 4)),$	$\emptyset, J_2 \rangle$
	$\rightarrow^{\texttt{rcv.call}(f,4)}$	$\rightarrow^{\texttt{send.call}(f,4)}$	$\langle \texttt{rcv.call}_q (\texttt{send.call}_f (f \perp)) \rangle$	$\langle \emptyset, J_2 \rangle$
$\langle \texttt{rcv.call}_y (\texttt{send.call}_g (\texttt{rcv.call}_f ((\lambda x. x+1) 4)) \rangle$	$)), \emptyset, J_1 \rangle \rightarrow^2$			· · · ·
$\langle \text{rcv.call}_{y} (\text{send.call}_{g} (\text{rcv.call}_{f} 5)),$	$\emptyset, J_1 \rangle \rightarrow^{\texttt{send.ret}(f,5)}$	$\rightarrow^{\texttt{rcv.ret}(f,5)}$	$\langle \texttt{rcv.call}_g \ (f \ 5),$	$\emptyset, J_2 \rangle$
$\langle \text{rcv.call}_y \text{ (send.call}_q \bot), \rangle$	$\emptyset, J_1 \rangle \rightarrow^{\texttt{rcv.call}(f,5)}$	$\rightarrow^{\texttt{send.call}(f,5)}$	$\langle \texttt{rcv.call}_{g} (\texttt{send.call}_{f} \perp),$	$\emptyset, J_2 \rangle$
$\langle \texttt{rcv.call}_y \ (\texttt{send.call}_g \ (\texttt{rcv.call}_f \ ((\lambda x. x+1) \ 5))) \rangle$	$)), \emptyset, J_1 \rangle \rightarrow^2$			
$\langle \text{rcv.call}_y \text{ (send.call}_q \text{ (rcv.call}_f 6)),$	$\emptyset, J_1 \rangle \rightarrow^{\texttt{send.ret}(f,6)}$	$\rightarrow^{\texttt{rcv.ret}(f,6)}$	$\langle \texttt{rcv.call}_g 6,$	$\emptyset, J_2 \rangle$
$\langle \texttt{rcv.call}_y (\texttt{send.call}_g \perp),$	$\emptyset, J_1 \rangle \rightarrow^{\texttt{rcv.ret}(g,6)}$	$\rightarrow^{\texttt{send.ret}(g,6)}$	$\langle \perp,$	$\emptyset, J_2 \rangle$
$\langle \texttt{rcv.call}_y 6,$	$\emptyset, J_1 \rangle \rightarrow^{\texttt{send.ret}(y, 6)}$			
$\langle \perp,$	$\emptyset, J_1 \rangle$			
		•		

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

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

Evaluation of $H = (\lambda t. t (\lambda x. x+1) 4)$		Evaluation of tw	$ice = (\lambda f. \lambda x. f (f x))$	
$I_1 = [y \mapsto (\lambda t. t \ (\lambda x. x+1) \ 4)]$		$I_2 = [t \mapsto (\lambda f.$	$\lambda x. f(f x))]$	
$J_1 = \begin{bmatrix} y \mapsto (\lambda t. t (\lambda x. x+1) 4), f \mapsto (\lambda x. x+1) \end{bmatrix}$		$J_2 = [t \mapsto (\lambda f)]$	$\lambda x. f(fx)), g \mapsto (\lambda x. f(fx))$	
$\langle \texttt{rcv.call}_{start} (\lambda t. t (\lambda x. x+1) 4),$	$\emptyset, \emptyset \rightarrow \overset{\texttt{send.ret}(start, y)}{\rightarrow}$		$\langle \texttt{rcv.call}_{start} (\lambda f. \lambda x. f(f x)) \rangle$), \emptyset, \emptyset >
$\langle \perp,$	$\emptyset, I_1 \rangle \rightarrow^{\texttt{rcv.call}(y,t)}$	$\rightarrow^{\texttt{send.ret}(start,t)}$	$(\bot,$	\emptyset, I_2
$\langle \texttt{rcv.call}_y ((\lambda t. t (\lambda x. x+1) 4) t), \rangle$	$\emptyset, I_1 \rangle \rightarrow$			
$\langle \texttt{rcv.call}_y ((t (\lambda x. x+1)) 4),$	$\emptyset, I_1 \rangle \rightarrow^{\texttt{send.call}(t,f)}$	$\rightarrow^{\texttt{rcv.call}(t,f)}$	$\langle \texttt{rcv.call}_t ((\lambda f. \lambda x. f (f x)) f$	$), \emptyset, I_2 \rangle$
$\langle \texttt{rcv.call}_y \; (\texttt{send.call}_t \perp) \; 4),$	$\emptyset, J_1 \rangle$	\rightarrow	$\langle \texttt{rcv.call}_t \ (\lambda x. f \ (f \ x)),$	$\langle \emptyset, I_2 \rangle$
	$\rightarrow^{\texttt{rcv.ret}(t,g)}$	$\rightarrow^{\texttt{send.ret}(t,g)}$	$\langle \perp,$	$\emptyset, J_2 \rangle$
$\langle \texttt{rcv.call}_y (g 4),$	$\emptyset, J_1 \rangle \to^{\texttt{send.call}(g,4)}$	$\rightarrow^{\texttt{rcv.call}(g,4)}$	$\langle \texttt{rcv.call}_q ((\lambda x. f(f x)) 4),$	\emptyset, J_2
$\langle \texttt{rcv.call}_y \; (\texttt{send.call}_g \perp),$	\emptyset, J_1	\rightarrow	$\langle \texttt{rcv.call}_g (f (f 4)), \rangle$	$\langle \emptyset, J_2 \rangle$
	$\rightarrow^{\texttt{rcv.call}(f,4)}$	$\rightarrow^{\texttt{send.call}(f,4)}$	$\langle \texttt{rcv.call}_g (\texttt{send.call}_f (f \perp)) \rangle$), $\emptyset, J_2 \rangle$
$\langle \texttt{rcv.call}_y (\texttt{send.call}_g (\texttt{rcv.call}_f ((\lambda x. x+1) 4)) \rangle$)), $\emptyset, J_1 \rangle \rightarrow^2$			
$\langle \texttt{rcv.call}_{y} (\texttt{send.call}_{q} (\texttt{rcv.call}_{f} 5)),$	$\emptyset, J_1 \rangle \rightarrow^{\texttt{send.ret}(f,5)}$	$\rightarrow^{\texttt{rcv.ret}(f,5)}$	$\langle \texttt{rcv.call}_g \ (f \ 5),$	$\emptyset, J_2 \rangle$
$\langle \text{rcv.call}_y \text{ (send.call}_q \bot), \rangle$	$(\emptyset, J_1) \to^{\texttt{rcv.call}(f,5)}$	$\rightarrow^{\texttt{send.call}(f,5)}$	$\langle \texttt{rcv.call}_q (\texttt{send.call}_f \perp),$	$\emptyset, J_2 \rangle$
$\langle \texttt{rcv.call}_y \ (\texttt{send.call}_g \ (\texttt{rcv.call}_f \ ((\lambda x. x+1) \ 5))) \rangle$)), $\emptyset, J_1 \rangle \rightarrow^2$			
$\langle \text{rcv.call}_y \text{ (send.call}_g \text{ (rcv.call}_f 6)), \rangle$	$\emptyset, J_1 \rangle \rightarrow^{\texttt{send.ret}(f,6)}$	$\rightarrow^{\texttt{rcv.ret}(f,6)}$	$\langle \texttt{rcv.call}_g 6,$	$\emptyset, J_2 \rangle$
$\langle \text{rcv.call}_y \text{ (send.call}_q \bot), \rangle$	$(\emptyset, J_1) \to rcv.ret(g, 6)$	$\rightarrow^{\texttt{send.ret}(g,6)}$	$\langle \bot,$	$\langle 0, J_2 \rangle$
$\langle \text{rcv.call}_y 6, \rangle$	$\emptyset, J_1 \rangle \rightarrow^{\texttt{send.ret}(y, 6)}$			
$\langle \perp, \rangle$	$\langle 0, J_1 \rangle$			

Step 2: Contracts are Trace Predicates

[[e]]

K ∩ [[e]]

Universal Contract: $[(\lambda x. x)]$

Properties

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

Step 3: Programming Simple Contracts

```
1 guard M =

2 \lambda x. if (constant? x) then

3 assert ((fst M) x) != false

4 x

5 else

6 let MM = (snd M)()

7 assert MM != false

8 \lambda y. (guard (snd MM) (x (guard (fst MM) y)))
```

```
1 guard M =

2 \lambda x. if (constant? x) then

3 assert ((fst M) x) != false

4 x

5 else

6 let MM = (snd M)()

7 assert MM != false

8 \lambda y. (guard (snd MM) (x (guard (fst MM) y)))
```

```
1 guard M =

2 \lambda x. if (constant? x) then

3 assert ((fst M) x) != false

4 X

5 else

6 let MM = (snd M)()

7 assert MM != false

8 \lambda y. (guard (snd MM) (x (guard (fst MM) y)))
```

```
1 guard M =

2 \lambda x. if (constant? x) then

3 assert ((fst M) x) != false

4 x

5 else

6 let MM = (snd M)()

7 assert MM != false

8 \lambda y. (guard (snd MM) (x (guard (fst MM) y)))
```

```
1 guard M =

2 \lambda x. if (constant? x) then

3 assert ((fst M) x) != false

4 x

5 else

6 let MM = (snd M)()

7 assert MM != false

8 \lambda y. (guard (snd MM) (x (guard (fst MM) y)))
```

```
1 guard M =

2 \lambda x. if (constant? x) then

3 assert ((fst M) x) != false

4 x

5 else

6 let MM = (snd M)()

7 assert MM != false

8 \lambda y. (guard (snd MM) (x (guard (fst MM) y)))
```

```
1 guard M =

2 \lambda x. if (constant? x) then

3 assert ((fst M) x) != false

4 x

5 else

6 let MM = (snd M)()

7 assert MM != false

8 \lambda y. (guard (snd MM) (x (guard (fst MM) y)))
```

```
1 guard M =

2 \lambda x. if (constant? x) then

3 assert ((fst M) x) != false

4 x

5 else

6 let MM = (snd M)()

7 assert MM != false

8 \lambda y. (guard (snd MM) (x (guard (fst MM) y)))
```

```
1 guard M =

2 \lambda x. if (constant? x) then

3 assert ((fst M) x) != false

4 x

5 else

6 let MM = (snd M)()

7 assert MM != false

8 \lambda y. (guard (snd MM) (x (guard (fst MM) y)))
```

```
1 guard M =

2 \lambda x. if (constant? x) then

3 assert ((fst M) x) != false

4 x

5 else

6 let MM = (snd M)()

7 assert MM != false

8 \lambda y. (guard (snd MM) (x (guard (fst MM) y)))
```

Step 4: Programming Temporal Contracts

Figure 3: Declarative HOT Contracts

$$M ::= S \text{ where } R$$

$$S ::= flat(e) \mid n:S_1 \mapsto S_2$$

$$R ::= A \mid !A \mid RR \mid R^* \mid \text{not } R \mid R \cup R$$

$$\mid \dots \mid \text{call}(n, ?x) \mid R \mid \text{ret}(n, ?x) \mid R$$

$$A ::= \text{call}(n, p) \mid \text{ret}(n, p)$$

$$p ::= _ \mid x \mid c$$

$$n \in Name$$

HOT contract Structural contract Temporal contract

Event patterns Value patterns Function names

Figure 3: Declarative HOT Contracts

$$M ::= S \text{ where } R$$

$$S ::= flat(e) \mid n:S_1 \mapsto S_2$$

$$R ::= A \mid !A \mid RR \mid R^* \mid \text{not } R \mid R \cup R$$

$$\mid \dots \mid \texttt{call}(n,?x) \mid \texttt{ret}(n,?x) \mid R$$

$$A ::= \texttt{call}(n,p) \mid \texttt{ret}(n,p)$$

$$p ::= _ \mid x \mid c$$

$$n \in Name$$

HOT contract Structural contract Temporal contract

Event patterns Value patterns Function names let s $= s_0$ $call_n = \lambda i \cdots check$ and update s appropriately \cdots • • • $ret_n = \lambda o. \cdots$ check and update s appropriately \cdots in $compile(\lambda x.true, S)$ compile : (Constant $\cup \{\lambda \mathbf{x} . \mathbf{x}\} \mapsto Bool) \times S \mapsto Monitor$ $compile(f, flat(e)) \stackrel{\text{def}}{=}$ pair (λ x. (e x) && (f x)) $(\lambda$. false) $compile(f, n: S_1 \mapsto S_2) \stackrel{\text{def}}{=}$

pair (
$$\lambda$$
x. false)
(λ . (f (λ x.x)) && (pair compile(call_n, S₁)
compile(ret_n, S₂))))

$$\begin{array}{l} compile(f,y) \stackrel{\text{def}}{=} \\ \text{let (chkconst,chkfn)} = y \\ \text{pair } (\lambda x. (f x) & \&\& (chkconst x)) \\ (\lambda. (f (\lambda x. x)) \&\& (chkfn())) \end{array}$$

let s = s_0 $call_n = \lambda i...check and update s appropriately ...$ $<math>ret_n = \lambda o...check and update s appropriately ...$ $in compile(<math>\lambda x.true, S$)

$$\begin{array}{l} compile : (Constant \cup \{\lambda \mathbf{x} . \mathbf{x}\} \mapsto Bool) \times S \mapsto Monitor\\ compile(f, flat(e)) \stackrel{\text{def}}{=}\\ pair (\lambda \mathbf{x}. (e \ \mathbf{x}) \ \text{\&\&} \ (f \ \mathbf{x}))\\ (\lambda. \ \text{false})\\ compile(f, n : S_1 \mapsto S_2) \stackrel{\text{def}}{=}\\ pair (\lambda \mathbf{x}. \ \text{false})\\ (\lambda. \ (f \ (\lambda \mathbf{x} . \mathbf{x})) \ \text{\&\&} \ (pair \ compile(call_n, S_1)\\ compile(ret_n, \ S_2)))) \end{array}$$

$$\begin{array}{l} compile(f,y) \stackrel{\text{def}}{=} \\ \text{let (chkconst,chkfn)} = y \\ \text{pair } (\lambda x. (f x) & \&\& (chkconst x)) \\ (\lambda. (f (\lambda x. x)) \&\& (chkfn())) \end{array}$$

let s = s_0 $call_n = \lambda i...check and update s appropriately ...$ $ret_n = \lambda o...check and update s appropriately ...$ in $compile(\lambda x.true, S)$

$$\begin{array}{l} compile : (Constant \cup \{\lambda \mathtt{x} . \mathtt{x}\} \mapsto Bool) \times S \mapsto Monitor\\ compile(f, flat(e)) \stackrel{\text{def}}{=}\\ pair (\lambda \mathtt{x}. (e \ \mathtt{x}) \ \&\& \ (f \ \mathtt{x}))\\ (\lambda. \ false)\\ compile(f, n: S_1 \mapsto S_2) \stackrel{\text{def}}{=}\\ pair (\lambda \mathtt{x}. \ false)\\ (\lambda. \ (f \ (\lambda \mathtt{x} . \mathtt{x})) \ \&\& \ (pair \ compile(call_n, S_1)\\ compile(ret_n, \ S_2)))) \end{array}$$

$$\begin{array}{l} compile(f,y) \stackrel{\text{def}}{=} \\ \text{let (chkconst,chkfn)} = y \\ \text{pair } (\lambda x. (f x) & \&\& (chkconst x)) \\ (\lambda. (f (\lambda x. x)) \&\& (chkfn())) \end{array}$$

let s = s_0 $call_n = \lambda i...check and update s appropriately ...$ $ret_n = \lambda o...check and update s appropriately ...$ in $compile(\lambda x.true, S)$

$$\begin{array}{l} compile : (Constant \cup \{\lambda \mathbf{x} . \mathbf{x}\} \mapsto Bool) \times S \mapsto Monitor\\ compile(f, flat(e)) \stackrel{\texttt{def}}{=}\\ \texttt{pair} \ (\lambda \mathbf{x} . \ (e \ \mathbf{x}) \ \texttt{\&\&} \ (f \ \mathbf{x}))\\ (\lambda . \ \texttt{false})\\ compile(f, n: S_1 \mapsto S_2) \stackrel{\texttt{def}}{=}\\ \texttt{pair} \ (\lambda \mathbf{x} . \ \texttt{false})\\ (\lambda . \ (f \ (\lambda \mathbf{x} . \mathbf{x})) \ \texttt{\&\&} \ (\texttt{pair} \ compile(\texttt{call}_n, S_1)\\ compile(\texttt{ret}_n, \ S_2)))) \end{array}$$

$$\begin{array}{l} compile(f,y) \stackrel{\text{\tiny def}}{=} \\ \text{let (chkconst,chkfn)} = y \\ \text{pair } (\lambda x. (f x) & \&\& (chkconst x)) \\ (\lambda . (f (\lambda x. x)) \&\& (chkfn())) \end{array}$$

let s $= s_0$ $call_n = \lambda i \cdots check$ and update s appropriately \cdots . . . $ret_n = \lambda o. \cdots$ check and update s appropriately \cdots in $compile(\lambda x.true, S)$ compile : (Constant $\cup \{\lambda \mathbf{x} . \mathbf{x}\} \mapsto Bool) \times S \mapsto Monitor$ $compile(f, flat(e)) \stackrel{\texttt{def}}{=}$ pair (λx . (e x) && (f x)) $(\lambda$. false) $compile(f, n: S_1 \mapsto S_2) \stackrel{\text{def}}{=}$ pair ($\lambda x.$ false) $(\lambda. (f (\lambda x.x)) \&\& (pair compile(call_n, S_1))$ $compile(ret_n, S_2))))$ $compile(f, y) \stackrel{\text{def}}{=}$ let (chkconst, chkfn) = y

pair (λx . (f x) && (chkconst x)) (λ . (f ($\lambda x.x$)) && (chkfn()))

Step 5: Racket Implementation

Differences

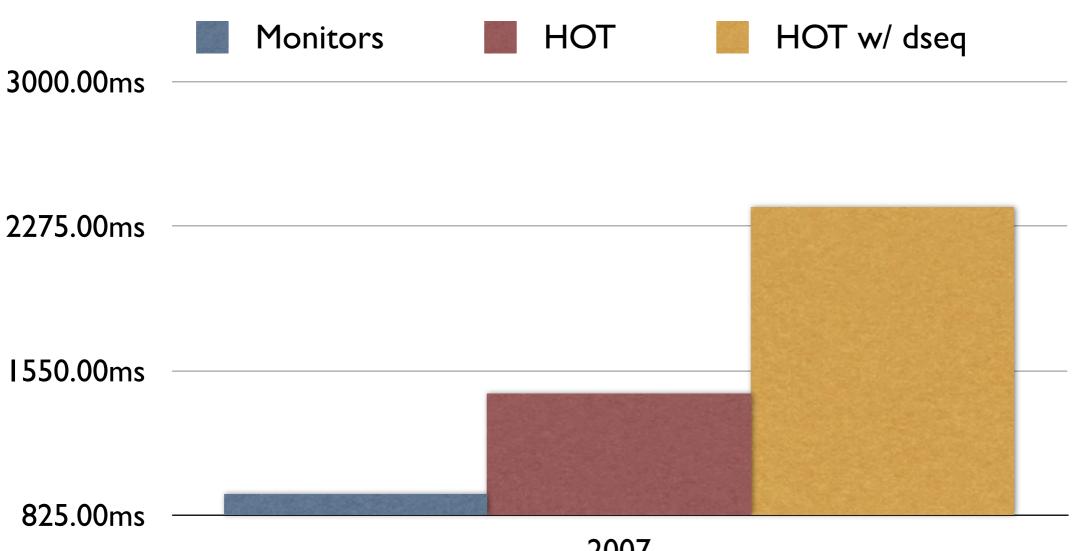
- No non-interference guarantee (inherited from Racket contracts)
- Continuations allow 0,
 I, and many returns
- Mutation intermediation through standard contracts

- 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	Ι7	curry
Unconstrained	13	apply



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 Als with contracts

Future Work

- Interaction with types
- Concurrency
- Explain contract violations