Simplified CML Semantics

Andrew W. Appel

December 7, 1999

This is a simplification of Reppy's CML Semantics (*Concurrent Programming in ML*, Appendix B).

1 Syntax

Expressions: e

e	$\rightarrow$	x	variables
	$\rightarrow$	b	constants
	$\rightarrow$	$\texttt{fn}\ x \Rightarrow e$	function abstraction
	$\rightarrow$	$e_1\ e_2$	function application
	$\rightarrow$	$\texttt{spawn}\ e$	thread creation
	$\rightarrow$	$\texttt{channel}\ ()$	channel creation
	$\rightarrow$	$\texttt{sync}\ e$	synchronization
	$\rightarrow$	κ	channel names
	$\rightarrow$	$e?$	receive event
	$\rightarrow$	$e_1!e_2$	send event
	$\rightarrow$	$e_1 \Rightarrow e_2$	wrap event
	$\rightarrow$	$e_1 \oplus e_2$	choose
	$\rightarrow$	Λ	never

Values: v

v	$\rightarrow$	b	constants
	$\rightarrow$	$\texttt{fn}\ x \Rightarrow e$	function abstraction
	$\rightarrow$	κ	channel names
	$\rightarrow$	ev	event values

Event Values: ev

ev	$\rightarrow$	$\kappa?$	
	$\rightarrow$	$\kappa!v$	send event
	$\rightarrow$	$ev \Rightarrow v$	wrap event
	$\rightarrow$	$ev_1 \oplus ev_2$	choose
	$\rightarrow$	Λ	never

2 Small-step Dynamic Semantics

Evaluation contexts: E (instead of search rules)

$$
\begin{array}{rcl}
E & \rightarrow & [\,] \mid E\ e \mid v\ E \mid \texttt{spawn}\ E \mid \texttt{sync}\ E \\
& \mid & E\,? \mid E\,!\,e \mid ev\,!\,E \mid E \Rightarrow e \mid ev \Rightarrow E \mid E \oplus e \mid ev \oplus E
\end{array}
$$

Sequential evaluation: $\hookrightarrow$

$$
\frac{\text{for various } b, b'}{E[b\ v] \hookrightarrow E[b']}\ \text{EvalOp}
\qquad
\frac{}{E[(\texttt{fn}\ x \texttt{ => } e)\ v] \hookrightarrow E[[v/x]e]}\ \text{CallFun}
$$

Concurrent evaluation: $\Rightarrow$

$$
\frac{e \hookrightarrow e' \qquad \pi \notin \mathrm{dom}(\mathcal{P})}{\mathcal{P} \cup \{\langle \pi, e\rangle\} \Rightarrow \mathcal{P} \cup \{\langle \pi, e'\rangle\}}\ \text{EvalSeq}
$$

$$
\frac{\kappa \text{ not free in } \mathcal{P} \cup \{\langle \pi, E[\texttt{channel}\ ()]\rangle}{\mathcal{P} \cup \{\langle \pi, E[\texttt{channel}\ ()]\rangle\} \Rightarrow \mathcal{P} \cup \{\langle \pi, E[\kappa]\rangle\}}\ \text{EvalChannel}
$$

$$
\frac{\pi \notin \mathrm{dom}(\mathcal{P}) \qquad \pi' \notin \mathrm{dom}(\mathcal{P}) \qquad \pi \neq \pi'}{\mathcal{P} \cup \{\langle \pi, E[\texttt{spawn}\ v]\rangle\} \Rightarrow \mathcal{P} \cup \{\langle \pi, E[()]\rangle, \langle \pi',\ v()\rangle\}}\ \text{EvalSpawn}
$$

$$
\frac{(ev_1, ev_2) \rightsquigarrow (e_1, e_2)}{\mathcal{P} \cup \{\langle \pi, E_1[\texttt{sync}\ ev_1]\rangle, \langle \pi, E_2[\texttt{sync}\ ev_2]\rangle\} \Rightarrow \mathcal{P} \cup \{\langle \pi, E_1[e_1]\rangle, \langle \pi, E_2[e_2]\rangle\}}\ \text{EvalSync}
$$

Event Matching: $\rightsquigarrow$

$$
\frac{}{(\kappa!v, \kappa?) \rightsquigarrow ((), v)}\ \text{MatchChan}
$$

$$
\frac{(ev_1, ev_2) \rightsquigarrow (e_1, e_2)}{(ev_1 \oplus ev', ev_2) \rightsquigarrow (e_1, e_2)}\ \text{MatchChooseL}
\qquad
\frac{(ev_1, ev_2) \rightsquigarrow (e_1, e_2)}{(ev' \oplus ev_1, ev_2) \rightsquigarrow (e_1, e_2)}\ \text{MatchChooseR}
$$

$$
\frac{(ev_1, ev_2) \rightsquigarrow (e_1, e_2)}{(ev_1 \Rightarrow f,\ ev_2) \rightsquigarrow (f(e_1),\ e_2)}\ \text{MatchWrap}
$$

$$
\frac{(ev_1, ev_2) \rightsquigarrow (e_1, e_2)}{(ev_2, ev_1) \rightsquigarrow (e_2, e_1)}\ \text{MatchPermute}
$$