Higher-order asynchronous effects

Values

V, W	$::= x$	variable
	$\mid ()$	unit
	$\mid (V, W)$	pair
	$\mid \mathsf{inl}_{X,Y}\, V \mid \mathsf{inr}_{X,Y}\, V$	injection
	$\mid \mathsf{fun}\ (x : X) \mapsto M$	function abstraction
	$\mid \langle V \rangle$	fulfilled promise
	$\mid [V]$	boxed mobile value

Computations

M, N	$::= \mathsf{return}\ V$	value
	$\mid \mathsf{let}\ x = M\ \mathsf{in}\ N$	sequential composition
	$\mid \mathsf{let\ rec}\ f\ x : X \to Y \,!\, (o, \iota) \triangleleft C = M\ \mathsf{in}\ N$	recursive definition
	$\mid V\, W$	application
	$\mid \mathsf{match}\ V\ \mathsf{with}\ \{(x, y) \mapsto M\}$	product elimination
	$\mid \mathsf{match}\ V\ \mathsf{with}\ \{\}_{Z!(o,\iota)\triangleleft C}$	empty elimination
	$\mid \mathsf{match}\ V\ \mathsf{with}\ \{\mathsf{inl}\ x \mapsto M, \mathsf{inr}\ y \mapsto N\}$	sum elimination
	$\mid \uparrow \mathrm{op}\ (V, M)$	outgoing signal
	$\mid \downarrow \mathrm{op}\ (V, M)$	incoming interrupt
	$\mid \mathsf{promise}\ (\mathrm{op}\ x \mapsto M)\ \mathsf{as}\ p\ \mathsf{in}\ N$	interrupt handler
	$\mid \mathsf{await}\ V\ \mathsf{until}\ \langle x \rangle\ \mathsf{in}\ M$	await a promise to be fulfilled
	$\mid \mathsf{unbox}\ V\ \mathsf{as}\ [x]\ \mathsf{in}\ M$	unbox a mobile value
	$\mid \mathsf{spawn}\ (M, N)$	spawn a new process

Fig. 1. Values and computations.

Standard computation rules

$$(\mathsf{fun}\ (x:X) \mapsto M)\ V \rightsquigarrow M[V/x]$$

$$\mathsf{let}\ x = \mathsf{return}\ V\ \mathsf{in}\ N \rightsquigarrow N[V/x]$$

$$\mathsf{match}\ (V, W)\ \mathsf{with}\ \{(x, y) \mapsto M\} \rightsquigarrow M[V/x, W/y]$$

$$\mathsf{match}\ (\mathsf{inl}_{X,Y}\ V)\ \mathsf{with}\ \{\mathsf{inl}\ x \mapsto M, \mathsf{inr}\ y \mapsto N\} \rightsquigarrow M[V/x]$$

$$\mathsf{match}\ (\mathsf{inr}_{X,Y}\ W)\ \mathsf{with}\ \{\mathsf{inl}\ x \mapsto M, \mathsf{inr}\ y \mapsto N\} \rightsquigarrow N[W/y]$$

$$\mathsf{let\ rec}\ f\ x : X \to Y!(o, \iota) \lhd C = M\ \mathsf{in}\ N \rightsquigarrow N[\mathsf{fun}\ (x:X) \mapsto \mathsf{let\ rec}\ f\ x : X \to Y!(o, \iota) \lhd C = M\ \mathsf{in}\ M/f]$$

Unboxing mobile values

$$\mathsf{unbox}\ [V]\ \mathsf{as}\ [x]\ \mathsf{in}\ M \rightsquigarrow M[V/x]$$

Algebraicity of signals, interrupt handlers, and process spawning

$$\mathsf{let}\ x = (\uparrow \mathsf{op}\ (V, M))\ \mathsf{in}\ N \rightsquigarrow \uparrow \mathsf{op}\ (V, \mathsf{let}\ x = M\ \mathsf{in}\ N)$$

$$\mathsf{let}\ x = (\mathsf{promise}\ (\mathsf{op}\ y \mapsto M)\ \mathsf{as}\ p\ \mathsf{in}\ N_1)\ \mathsf{in}\ N_2 \rightsquigarrow \mathsf{promise}\ (\mathsf{op}\ y \mapsto M)\ \mathsf{as}\ p\ \mathsf{in}\ (\mathsf{let}\ x = N_1\ \mathsf{in}\ N_2)$$

$$\mathsf{let}\ x = (\mathsf{spawn}\ (M, N_1))\ \mathsf{in}\ N_2 \rightsquigarrow \mathsf{spawn}\ (M, \mathsf{let}\ x = N_1\ \mathsf{in}\ N_2)$$

Commutativity of interrupt handlers with signals and process spawning

$$\mathsf{promise}\ (\mathsf{op}\ x \mapsto M)\ \mathsf{as}\ p\ \mathsf{in} \uparrow \mathsf{op}'\ (V, N) \rightsquigarrow \uparrow \mathsf{op}'\ (V, \mathsf{promise}\ (\mathsf{op}\ x \mapsto M)\ \mathsf{as}\ p\ \mathsf{in}\ N)$$

$$\mathsf{promise}\ (\mathsf{op}\ x \mapsto M)\ \mathsf{as}\ p\ \mathsf{in}\ \mathsf{spawn}\ (N_1, N_2) \rightsquigarrow \mathsf{spawn}\ (N_1, \mathsf{promise}\ (\mathsf{op}\ x \mapsto M)\ \mathsf{as}\ p\ \mathsf{in}\ N_2)$$

Interrupt propagation

$$\downarrow \mathsf{op}\ (V, \mathsf{return}\ W) \rightsquigarrow \mathsf{return}\ W$$

$$\downarrow \mathsf{op}\ (V, \uparrow \mathsf{op}'\ (W, M)) \rightsquigarrow \uparrow \mathsf{op}'\ (W, \downarrow \mathsf{op}\ (V, M))$$

$$\downarrow \mathsf{op}\ (V, \mathsf{promise}\ (\mathsf{op}\ x \mapsto M)\ \mathsf{as}\ p\ \mathsf{in}\ N) \rightsquigarrow \mathsf{let}\ p = M[V/x]\ \mathsf{in} \downarrow \mathsf{op}\ (V, N)$$

$$\downarrow \mathsf{op}'\ (V, \mathsf{promise}\ (\mathsf{op}\ x \mapsto M)\ \mathsf{as}\ p\ \mathsf{in}\ N) \rightsquigarrow \mathsf{promise}\ (\mathsf{op}\ x \mapsto M)\ \mathsf{as}\ p\ \mathsf{in} \downarrow \mathsf{op}'\ (V, N) \qquad (\mathsf{op} \neq \mathsf{op}')$$

$$\downarrow \mathsf{op}\ (V, \mathsf{spawn}\ (M, N)) \rightsquigarrow \mathsf{spawn}\ (M, \downarrow \mathsf{op}\ (V, N))$$

Awaiting a promise to be fulfilled

$$\mathsf{await}\ \langle V \rangle\ \mathsf{until}\ \langle x \rangle\ \mathsf{in}\ M \rightsquigarrow M[V/x]$$

Evaluation context rule

$$\frac{M \rightsquigarrow N}{\mathcal{E}[M] \rightsquigarrow \mathcal{E}[N]}$$

where

$$\begin{aligned} \mathcal{E} \ ::= &\ [\,] \\ \mid &\ \mathsf{let}\ x = \mathcal{E}\ \mathsf{in}\ N \\ \mid &\ \uparrow \mathsf{op}\ (V, \mathcal{E}) \\ \mid &\ \downarrow \mathsf{op}\ (V, \mathcal{E}) \\ \mid &\ \mathsf{promise}\ (\mathsf{op}\ x \mapsto M)\ \mathsf{as}\ p\ \mathsf{in}\ \mathcal{E} \\ \mid &\ \mathsf{spawn}\ (M, \mathcal{E}) \end{aligned}$$

Fig. 2. Small-step operational semantics of computations.

$$\text{Mobile type } A, B \ ::= \mathsf{b} \ \big| \ 1 \ \big| \ 0 \ \big| \ A \times B \ \big| \ A + B \ \big| \ [X]$$

Signal or interrupt signature: $\mathsf{op} : A_{\mathsf{op}}$

Outgoing signal annotations: $o \in O$

Interrupt handler annotations: $\iota \in I$

$$\text{Value type } X, Y \ ::= A \ \big| \ X \times Y \ \big| \ X + Y \ \big| \ X \to Y \,!\, (o, \iota) \triangleleft C \ \big| \ \langle X \rangle \ \big| \ [X]$$

Computation type: $X \,!\, (o, \iota) \triangleleft C$

Fig. 3. Value and computation types

$$\textsc{TyVal-Var}\ \frac{X \text{ is mobile} \quad \text{or} \quad \blacksquare \notin \Gamma'}{\Gamma, x : X, \Gamma' \vdash x : X} \qquad \textsc{TyVal-Unit}\ \frac{}{\Gamma \vdash () : 1} \qquad \textsc{TyVal-Pair}\ \frac{\Gamma \vdash V : X \qquad \Gamma \vdash W : Y}{\Gamma \vdash (V, W) : X \times Y}$$

$$\textsc{TyVal-Inl}\ \frac{\Gamma \vdash V : X}{\Gamma \vdash \mathsf{inl}_{X,Y}\, V : X + Y} \qquad \textsc{TyVal-Inr}\ \frac{\Gamma \vdash W : Y}{\Gamma \vdash \mathsf{inr}_{X,Y}\, W : X + Y} \qquad \textsc{TyVal-Fun}\ \frac{\Gamma, x : X \vdash M : Y \,!\, (o, \iota) \triangleleft C}{\Gamma \vdash \mathsf{fun}\ (x : X) \mapsto M : X \to Y \,!\, (o, \iota) \triangleleft C}$$

$$\textsc{TyVal-Promise}\ \frac{\Gamma \vdash V : X}{\Gamma \vdash \langle V \rangle : \langle X \rangle} \qquad \textsc{TyVal-Box}\ \frac{\Gamma, \blacksquare \vdash V : X}{\Gamma \vdash [V] : [X]}$$

Admissible strengthening rule needed for safely unboxing boxes (unit of ■)

$$\frac{\Gamma, \blacksquare, \Gamma' \vdash V : X}{\Gamma, \Gamma' \vdash V : X}$$

Other admissible rules (making ■ an idempotent functor/monad)

$$\frac{\Gamma, \blacksquare, \Gamma' \vdash V : X}{\Gamma, \blacksquare, \blacksquare, \Gamma' \vdash V : X} \qquad \frac{\Gamma, \blacksquare, \blacksquare, \Gamma' \vdash V : X}{\Gamma, \blacksquare, \Gamma' \vdash V : X}$$

Fig. 4. Value typing rules.

$$\textsc{TyComp-Return}\quad \frac{\Gamma \vdash V : X}{\Gamma \vdash \mathsf{return}\ V : X \,!\, (o, \iota) \lhd C}$$

$$\textsc{TyComp-Let}\quad \frac{\Gamma \vdash M : X \,!\, (o, \iota) \lhd C \qquad \Gamma, x \!:\! X \vdash N : Y \,!\, (o, \iota) \lhd C}{\Gamma \vdash \mathsf{let}\ x = M\ \mathsf{in}\ N : Y \,!\, (o, \iota) \lhd C}$$

$$\textsc{TyComp-LetRec}\quad \frac{\Gamma, f \!:\! X \to Y \,!\, (o, \iota) \lhd C, x \!:\! X \vdash M : Y \,!\, (o, \iota) \lhd C \qquad \Gamma, f \!:\! X \to Y \,!\, (o, \iota) \lhd C \vdash N : Z \,!\, (o', \iota') \lhd C}{\Gamma \vdash \mathsf{let\ rec}\ f\ x : X \to Y \,!\, (o, \iota) \lhd C = M\ \mathsf{in}\ N : Z \,!\, (o', \iota') \lhd C}$$

$$\textsc{TyComp-Apply}\quad \frac{\Gamma \vdash V : X \to Y \,!\, (o, \iota) \lhd C \qquad \Gamma \vdash W : X}{\Gamma \vdash V\, W : Y \,!\, (o, \iota) \lhd C}$$

$$\textsc{TyComp-MatchPair}\quad \frac{\Gamma \vdash V : X \times Y \qquad \Gamma, x \!:\! X, y \!:\! Y \vdash M : Z \,!\, (o, \iota) \lhd C}{\Gamma \vdash \mathsf{match}\ V\ \mathsf{with}\ \{(x, y) \mapsto M\} : Z \,!\, (o, \iota) \lhd C}$$

$$\textsc{TyComp-MatchEmpty}\quad \frac{\Gamma \vdash V : 0}{\Gamma \vdash \mathsf{match}\ V\ \mathsf{with}\ \{\}_{Z!(o,\iota)\lhd C} : Z \,!\, (o, \iota) \lhd C}$$

$$\textsc{TyComp-MatchSum}\quad \frac{\Gamma \vdash V : X + Y \qquad \Gamma, x \!:\! X \vdash M : Z \,!\, (o, \iota) \lhd C \qquad \Gamma, y \!:\! Y \vdash N : Z \,!\, (o, \iota) \lhd C}{\Gamma \vdash \mathsf{match}\ V\ \mathsf{with}\ \{\mathsf{inl}\ x \mapsto M, \mathsf{inr}\ y \mapsto N\} : Z \,!\, (o, \iota) \lhd C}$$

$$\textsc{TyComp-Signal}\quad \frac{\mathsf{op} \in o \qquad \Gamma \vdash V : A_{\mathsf{op}} \qquad \Gamma \vdash M : X \,!\, (o, \iota) \lhd C}{\Gamma \vdash \uparrow \mathsf{op}\, (V, M) : X \,!\, (o, \iota) \lhd C}$$

$$\textsc{TyComp-Interrupt}\quad \frac{\Gamma \vdash V : A_{\mathsf{op}} \qquad \Gamma \vdash M : X \,!\, (o, \iota) \lhd C}{\Gamma \vdash \downarrow \mathsf{op}\, (V, M) : X \,!\, \mathsf{op} \downarrow ((o, \iota) \lhd C)}$$

$$\textsc{TyComp-Promise}\quad \frac{\iota\, (\mathsf{op}) = (o', \iota') \lhd C \qquad \Gamma, x \!:\! A_{\mathsf{op}} \vdash M : \langle X \rangle \,!\, (o', \iota') \lhd C \qquad \Gamma, p \!:\! \langle X \rangle \vdash N : Y \,!\, (o, \iota) \lhd \mathcal{D}}{\Gamma \vdash \mathsf{promise}\ (\mathsf{op}\ x \mapsto M)\ \mathsf{as}\ p\ \mathsf{in}\ N : Y \,!\, (o, \iota) \lhd \mathcal{D}}$$

$$\textsc{TyComp-Await}\quad \frac{\Gamma \vdash V : \langle X \rangle \qquad \Gamma, x \!:\! X \vdash M : Y \,!\, (o, \iota) \lhd C}{\Gamma \vdash \mathsf{await}\ V\ \mathsf{until}\ \langle x \rangle\ \mathsf{in}\ M : Y \,!\, (o, \iota) \lhd C}$$

$$\textsc{TyComp-Unbox}\quad \frac{\Gamma \vdash V : [X] \qquad \Gamma, x \!:\! X \vdash M : Y \,!\, (o', \iota') \lhd C}{\Gamma \vdash \mathsf{unbox}\ V\ \mathsf{as}\ [x]\ \mathsf{in}\ M : Y \,!\, (o', \iota') \lhd C}$$

$$\textsc{TyComp-Spawn}\quad \frac{\Gamma, \blacksquare \vdash M : X \,!\, (o, \iota) \lhd C \qquad (X \,!\, (o, \iota) \lhd C) \in \mathcal{D} \qquad \Gamma \vdash N : Y \,!\, (o', \iota') \lhd \mathcal{D}}{\Gamma \vdash \mathsf{spawn}\ (M, N) : Y \,!\, (o', \iota') \lhd \mathcal{D}}$$

$$\textsc{TyComp-Subsume}\quad \frac{\Gamma \vdash M : X \,!\, (o, \iota) \lhd C \qquad (o, \iota) \sqsubseteq (o', \iota') \qquad C \sqsubseteq \mathcal{D}}{\Gamma \vdash M : X \,!\, (o', \iota') \lhd \mathcal{D}}$$

Admissible strengthening rule needed for safely unboxing boxes (unit of ■)

$$\frac{\Gamma, \blacksquare, \Gamma' \vdash M : X \,!\, (o, \iota) \lhd C}{\Gamma, \Gamma' \vdash M : X \,!\, (o, \iota) \lhd C}$$

Other admissible rules (making ■ an idempotent functor/monad)

$$\frac{\Gamma, \blacksquare, \Gamma' \vdash M : X \,!\, (o, \iota) \lhd C}{\Gamma, \blacksquare, \blacksquare, \Gamma' \vdash M : X \,!\, (o, \iota) \lhd C} \qquad \frac{\Gamma, \blacksquare, \blacksquare, \Gamma' \vdash M : X \,!\, (o, \iota) \lhd C}{\Gamma, \blacksquare, \Gamma' \vdash M : X \,!\, (o, \iota) \lhd C}$$

Fig. 5. Computation typing rules.

Individual computations

$$\frac{M \leadsto N}{\mathsf{run}\ M \leadsto \mathsf{run}\ N}$$

Signal hoisting

$$\mathsf{run}\ (\uparrow \mathsf{op}\ (V, M)) \leadsto \uparrow \mathsf{op}\ (V, \mathsf{run}\ M)$$

Broadcasting

$$\uparrow \mathsf{op}\ (V, P) \mid\mid Q \leadsto \uparrow \mathsf{op}\ (V, P \mid\mid \downarrow \mathsf{op}\ (V, Q))$$

$$P \mid\mid \uparrow \mathsf{op}\ (V, Q) \leadsto \uparrow \mathsf{op}\ (V, \downarrow \mathsf{op}\ (V, P) \mid\mid Q)$$

Interrupt propagation

$$\downarrow \mathsf{op}\ (V, \mathsf{run}\ M) \leadsto \mathsf{run}\ (\downarrow \mathsf{op}\ (V, M))$$

$$\downarrow \mathsf{op}\ (V, P \mid\mid Q) \leadsto \downarrow \mathsf{op}\ (V, P) \mid\mid \downarrow \mathsf{op}\ (V, Q)$$

$$\downarrow \mathsf{op}\ (V, \uparrow \mathsf{op}'\ (W, P)) \leadsto \uparrow \mathsf{op}'\ (W, \downarrow \mathsf{op}\ (V, P))$$

Spawned process hoisting

$$\mathsf{run}\ (\mathsf{spawn}\ (M, N)) \leadsto \mathsf{run}\ M \mid\mid \mathsf{run}\ N$$

Evaluation context rule

$$\frac{P \leadsto Q}{\mathcal{F}[P] \leadsto \mathcal{F}[Q]}$$

where

$$\mathcal{F} ::= [\,] \ \big|\ \mathcal{F} \mid\mid Q \ \big|\ P \mid\mid \mathcal{F} \ \big|\ \uparrow \mathsf{op}\ (V, \mathcal{F}) \ \big|\ \downarrow \mathsf{op}\ (V, \mathcal{F})$$

Fig. 6. Small-step operational semantics of parallel processes.

$$\textsc{TyProc-Run}\ \frac{\Gamma \vdash M : X\ !\ (o, \iota) \lhd C}{\Gamma \vdash \mathsf{run}\ M : X\ !!\ (o, \iota) \lhd C} \qquad \textsc{TyProc-Par}\ \frac{\Gamma \vdash P : C \qquad \Gamma \vdash Q : D}{\Gamma \vdash P \mid\mid Q : C \mid\mid D}$$

$$\textsc{TyProc-Signal}\ \frac{\mathsf{op} \in \text{signals-of}(C) \quad \Gamma \vdash V : A_{\mathsf{op}} \qquad \Gamma \vdash P : C}{\Gamma \vdash \uparrow \mathsf{op}\ (V, P) : C} \qquad \textsc{TyProc-Interrupt}\ \frac{\Gamma \vdash V : A_{\mathsf{op}} \qquad \Gamma \vdash P : C}{\Gamma \vdash \downarrow \mathsf{op}\ (V, P) : \mathsf{op} \downarrow C}$$

Admissible strengthening rule needed for safely spawning processes (unit of ■)

$$\frac{\Gamma, \blacksquare, \Gamma' \vdash P : C}{\Gamma, \Gamma' \vdash P : C}$$

Other admissible rules (making ■ an idempotent functor/monad)

$$\frac{\Gamma, \blacksquare, \Gamma' \vdash P : C}{\Gamma, \blacksquare, \blacksquare, \Gamma' \vdash P : C} \qquad \frac{\Gamma, \blacksquare, \blacksquare, \Gamma' \vdash P : C}{\Gamma, \blacksquare, \Gamma' \vdash P : C}$$

Fig. 7. Process typing rules.