Clairvoyant Semantics for Adaptivity Analysis

Pure Expr.	e	$::=$	$c \mid x \mid e\, e \mid \lambda x.e \mid \texttt{fix}\, f(x)\, e \mid (e,e) \mid \texttt{fst}\, e \mid \texttt{snd}\, e$
			$\texttt{true} \mid \texttt{false} \mid \texttt{if}(e,e,e) \mid \texttt{nil} \mid e\, \texttt{cons}\, e \mid \{m\}$
			$\texttt{let}\, x = e_1\, \texttt{in}\, e_2$
Monadic Expr.	m	$::=$	$\texttt{return}(e) \mid \texttt{mlet}\, x = e\, \texttt{in}\, m \mid (m,m)_m \mid \delta(m)$
Value.	v	$::=$	$c \mid \lambda x.e \mid (v_1, v_2) \mid \texttt{true} \mid \texttt{false} \mid \{m\}$
Types	τ	$::=$	$\texttt{b} \mid \tau \to \tau \mid \tau \times \tau \mid \texttt{M}(\tau)$

Definition 1. *Let* $\vdash m : \texttt{M}(\tau)$*, its* adaptivity *is defined as:*

$$\min\{p \mid \exists e', v. m \Downarrow_f^p e' \wedge e' \Downarrow v\}$$

Definition 2. *Let* $\vdash e : \tau$*, its* adaptivity *is defined as:*

$$\min\{p \mid \exists m, e', v.\ e \Downarrow \{m\} \wedge m \Downarrow_f^p e' \wedge e' \Downarrow v\}$$

$$\Gamma, x : \tau \vdash x : \tau \ \textbf{ST-AX} \qquad \frac{}{\Gamma \vdash c : \texttt{b}}\ \textbf{ST-CST} \qquad \frac{\Gamma, x : \tau_1 \vdash e : \tau_2}{\Gamma \vdash \lambda x.e : \tau_1 \to \tau_2}\ \textbf{ST-LAM}$$

$$\frac{\Gamma \vdash e_1 : \tau_1 \to \tau_2 \qquad \Gamma \vdash e_2 : \tau_1}{\Gamma \vdash (e_1\ e_2) : \tau_2}\ \textbf{ST-app}$$

....

Figure 1: Simple Types - pure rules

$$\frac{\Gamma \vdash e : \tau}{\Gamma \vdash \texttt{return}(e) : \texttt{M}(\tau)}\ \textbf{ST-RET} \qquad \frac{\Gamma \vdash e : \texttt{M}(\tau_1) \qquad \Gamma, x : \tau_1 \vdash m : \texttt{M}(\tau_2)}{\Gamma \vdash \texttt{mlet}\ e = x\ \texttt{in}\ m : \texttt{M}(\tau_2)}\ \textbf{ST-LET}$$

$$\frac{\Gamma \vdash e : \texttt{M}(\texttt{row}) \qquad \hat{\delta} : \texttt{row} \to [0,1]}{\Gamma \vdash \delta(e) : \texttt{M}([0,1])}\ \textbf{ST-DELTA}$$

Figure 2: Simple Types - monadic rules

$$\frac{}{v \Downarrow v}\ \text{S-VAL} \qquad \frac{e_1 \Downarrow \lambda x.e \qquad e_2 \Downarrow v_2 \qquad e[v_2/x] \Downarrow v_3}{e_1\ e_2 \Downarrow v_3}\ \text{S-APP}$$

$$\frac{e_1 \Downarrow \texttt{true} \qquad e_2 \Downarrow v_2}{\theta, \texttt{if}(e_1, e_2, e_3) \Downarrow v_2}\ \text{S-IFT} \qquad \frac{e_1 \Downarrow \texttt{true} \qquad e_3 \Downarrow v_3}{\texttt{if}(e_1, e_2, e_3) \Downarrow v_3}\ \text{S-IFF} \qquad \frac{e_1 \Downarrow v_1 \qquad e_2 \Downarrow v_2}{(e_1, e_2) \Downarrow (v_1, v_2)}\ \text{S-PROD}$$

$$\frac{e \Downarrow (v_1, v_2)}{\texttt{fst}\ e \Downarrow v_1}\ \text{S-PL} \qquad \frac{e \Downarrow (v_1, v_2)}{\texttt{snd}(e) \Downarrow v_2}\ \text{S-PR}$$

Figure 3: Big-step semantics, pure rules.

$$\frac{}{\texttt{return}(e) \Downarrow_f^0 e}\ \text{FS-RET} \qquad \frac{e \Downarrow \{m_1\} \qquad m_1 \Downarrow_f^{p_1} e_1 \qquad m[e_1/x] \Downarrow_f^{p_2} e_2}{\texttt{mlet}\ x = e\ \texttt{in}\ m \Downarrow_f^{p_1+p_2} e_2}\ \text{FS-BIND}$$

$$\frac{m \Downarrow_f^{p_2} e_2}{\texttt{mlet}\ x = e_1\ \texttt{in}\ m \Downarrow_f^{p_2} e_2}\ \text{FS-SKIP} \qquad \frac{m \Downarrow_f^{p} e \qquad e \Downarrow v \qquad \hat{\delta}(v) = v'}{\delta(m) \Downarrow_f^{p+1} v'}\ \text{FS-MECH}$$

$$\frac{m_1 \Downarrow_f^{p_1} e_1 \qquad m_2 \Downarrow_f^{p_2} e_2}{(m_1, m_2)_m \Downarrow (e_1, e_2)}\ \text{FS-PROD}$$

Figure 4: Big-step semantics, forcing rules.

$$\Gamma, x :!_1 \tau \vdash x : \tau\ \textbf{ST-AX} \qquad \frac{}{\Gamma \vdash c : \texttt{b}}\ \textbf{ST-CST} \qquad \frac{\Gamma, x :!_i \tau_1 \vdash e : \tau_2}{\Gamma \vdash \lambda x.e :!_i \tau_1 \to \tau_2}\ \textbf{ST-LAM}$$

$$\frac{\Gamma_1 \vdash e_1 :!_i \tau_1 \to \tau_2 \qquad \Gamma_2 \vdash e_2 : \tau_1}{\Gamma_1 + i * \Gamma_2 \vdash (e_1\ e_2) : \tau_2}\ \textbf{ST-app}$$

....

Figure 5: Simple Types - pure rules

$$\frac{\Gamma \vdash e : \tau}{\Gamma \vdash \texttt{return}(e) : \texttt{M}(\tau)}\ \textbf{ST-RET} \qquad \frac{\Gamma \vdash e : \texttt{M}(\tau_1) \qquad \Gamma, x : \tau_1 \vdash m : \texttt{M}(\tau_2)}{\Gamma \vdash \texttt{mlet}\ e = x\ \texttt{in}\ m : \texttt{M}(\tau_2)}\ \textbf{ST-LET}$$

$$\frac{\Gamma \vdash e : \texttt{M}(\texttt{row}) \qquad \hat{\delta} : \texttt{row} \to [0,1]}{\Gamma \vdash \delta(e) : \texttt{M}([0,1])}\ \textbf{ST-DELTA}$$

Figure 6: Simple Types - monadic rules

1 Example

1. A mechnism with the simple input query : $\lambda x.\{\delta(\texttt{return}\,(x))\}\; c \Downarrow \{\delta(\texttt{return}\,(c))\}$.
2. A mechnism with a complex input query : $\lambda x.\{\delta(\texttt{return}\,(x))\}\; \{\delta(\texttt{return}\,(c))\} \Downarrow \{\delta(\texttt{return}\,(\delta(\texttt{return}\,(c))))\}$.
3. A mechnism with more arguments :
$\lambda x.\lambda y.\{\texttt{mlet}\; z = \{\delta(\texttt{return}\,(x))\}\;\texttt{in}\;\delta(\texttt{return}\,(\texttt{fst}\,y + z))\}\; c\; (c_1, c_2)$
4. A mechnism with impure arguments :
$\lambda x.\lambda y.\{\texttt{mlet}\; z = x\;\texttt{in}\;\delta(\texttt{return}\,(\texttt{fst}\,y + z))\}\; \{\delta(\texttt{return}\,(c))\}\; (c_1, c_2)$.
5. store the mechnism results into a list, the adaptivity?
$\mathsf{mlet}\, l =$
$\big\{\texttt{mlet}\; a = \{\delta(\texttt{return}\,(c))\}\;\texttt{in}\,\texttt{mlet}\; b = \{\delta(\texttt{return}\,(c))\}\;\texttt{in}\,\texttt{return}\,(a :: b :: \texttt{nil})\big\}$
$\texttt{in}\;\delta(\texttt{return}\,(l[1] + l[2]))$.
6. two rounds
$\texttt{let}\, g = \texttt{fix}\, f(j)\,\lambda\, k.$
$\texttt{if}(j < k,$
$\{\texttt{mlet}\; a = \{\delta(\texttt{return}\,(j + k))\}\;\texttt{in}\,\texttt{mlet}\; t' = f\;(j+1)\; k\;\texttt{in}\,\texttt{return}\,(a\,\mathsf{cons}\, t')\},$
$\{\texttt{return}\,(\texttt{nil})\})$
$\texttt{in}\,\{\texttt{mlet}\; l = g\, 0\, K\;\texttt{in}\;\delta(\texttt{sign}(\texttt{foldl}\;(\lambda\, acc.\lambda\, a.(acc + \texttt{lg}(\frac{1+a}{1-a})))\; 0\; l))\}$