Revisit of Adaptivity analysis

1 Attempt 1: linear-type based

Types $\tau ::= b \mid \tau \multimap \tau' \mid !_n \tau \mid \tau \times \tau \mid \forall i{::}\mathbb{N}.\tau \mid \mathrm{Q}$

Term $t ::= c \mid \mathsf{fix}\, f(x).t \mid \ t\ t \ \mid !t \mid (t_1, t_2) \mid \mathsf{let}\ !x = t_1\ \mathsf{in}\ t_2 \ \mid \Lambda.t \mid t[] \mid \ \lambda x.t \ \mid M(t) \mid x \mid q \mid$
$\quad \mathsf{case}\ t\ \mathsf{of}\ \{c_i \Rightarrow t_i\}_{c_i \in b} \mid \mathsf{let}\ (x_1, x_2) = t_1\ \mathsf{in}\ t_2$

Normal Form $v ::= c \mid \mathsf{fix}\, f(x).t \mid !t \mid (v_1, v_2) \mid \Lambda.t \mid \ \lambda x.t \ \mid x \mid q \mid \mathsf{case}\ v\ \mathsf{of}\ \{c_i \Rightarrow v_i\}_{c_i \in b_i} \mid$
$\mathtt{nil} \mid \mathtt{cons}(v_1, v_2)$

Mechanisms $M ::= \mathtt{gauss} \mid \mathtt{thdt}$

Tree $T_b ::= c \mid M(T_{query}) \mid \mathsf{case}\ T_b\ \mathsf{of}\ \{c_i \Rightarrow T_{b_i}\}_{c_i \in b}$
$T_{query} ::= q \mid \mathsf{case}\ T_b\ \mathsf{of}\ \{c_i \Rightarrow T_{query_i}\}_{c_i \in b}$

Depth
$\mathsf{depth}(c) = 0$
$\mathsf{depth}(!t) = \mathsf{depth}(t)$
$\mathsf{depth}(t_1\ t_2) = \max(\mathsf{depth}(t_1), \mathsf{depth}(t_2))$
$\mathsf{depth}(M(t)) = 1 + \mathsf{depth}(t)$
$\mathsf{depth}(\lambda x.t) = \mathsf{depth}(t)$
$\mathsf{depth}(x) = 0$
$\mathsf{depth}(q) = 0$
$\mathsf{depth}((t_1, t_2)) = \max(\mathsf{depth}(t_1), \mathsf{depth}(t_2))$
$\mathsf{depth}(\mathsf{let}\ (x_1, x_2) = t\ \mathsf{in}\ t') = \max(\mathsf{depth}(t), \mathsf{depth}(t'))$
$\mathsf{depth}(\mathsf{let}\ !x = t\ \mathsf{in}\ t') = \max(\mathsf{depth}(t), \mathsf{depth}(t'))$
$\mathsf{depth}(\mathsf{case}\ t\ \mathsf{of}\ \{c_i \Rightarrow t_i\}_{c_i \in b}) = \max(\mathsf{depth}(t), \mathsf{depth}(t_i))$
$\mathsf{depth}(\Lambda.t) = \mathsf{depth}(t)$
$\mathsf{depth}(t\,[]) = \mathsf{depth}(t)$

FAILURE: [[this semantics doesn't work in multi-Round case.

1. **depth in the multi-round case is variable.**
2. **unable to count depth if output of δ doesn't explicitly affect input of next δ.**
   In the multi-round case, the δ result will affect some arguments. These arguments will then affect Database d which will be used in next δ nested in recursion.

]]

$$\boxed{\Gamma \vdash_{n,m} t : \tau}$$

$$\boxed{\Gamma ::= \emptyset \mid \Gamma, x : \tau \mid \Gamma, x : [\tau]_p}$$

$$\frac{}{\Gamma \vdash_{n,m} c : b}\ \textsc{const} \qquad \frac{\Gamma, x : \tau_1 \vdash_n t : \tau_2}{\Gamma \vdash_{n,m} \lambda x.t : \tau_1 \multimap \tau_2}\ \textsc{abs} \qquad \frac{[\Gamma] \vdash_n t : \tau}{\Delta, p + [\Gamma] \vdash_{n+p} !t : !_p \tau}\ \textsc{pr}$$

$$\frac{}{\Gamma, x : \tau \vdash_n x : \tau}\ \textsc{var} \qquad \frac{[\Gamma] \vdash_n t : query}{\Delta, 1 + [\Gamma] \vdash_{n+1} M(t) : b}\ \text{MT} \qquad \frac{}{\Gamma \vdash_n q : query}\ \textsc{query}$$

$$\frac{\Gamma_1 \vdash_{n_1} t_1 : \tau_1 \multimap \tau_2 \qquad \Gamma_2 \vdash_{n_2} t_2 : \tau_1}{\max(\Gamma_1, \Gamma_2) \vdash_{\max(n_1,n_2)} t_1\, t_2 : \tau_2}\ \textsc{app} \qquad \boxed{\frac{\Gamma, x : \tau \vdash_n t : \tau}{\Gamma, x : [\tau]_0 \vdash_n t : \tau}\ \textsc{der}}$$

$$\frac{\Gamma_1 \vdash_{n_1} t : !_p \tau \qquad \Gamma_2, x : [\tau]_p \vdash_{n_2} t' : \tau'}{\max(\Gamma_1, \Gamma_2) \vdash_{\max(n_1,n_2)} \mathsf{let}\ !x = t\ \mathsf{in}\ t' : \tau'}\ \textsc{let-b}$$

$$\frac{\Gamma_1 \vdash_{n_1} t : \tau_1 \times \tau_2 \qquad \Gamma_2, x_1 : \tau_1, x_2 : \tau_2 \vdash_{n_2} t' : \tau'}{\max(\Gamma_1, \Gamma_2) \vdash_{\max(n_1,n_2)} \mathsf{let}\ (x_1, x_2) = t\ \mathsf{in}\ t' : \tau'}\ \textsc{let-p}$$

$$\frac{\Gamma_1 \vdash_{n_1} t_1 : \tau_1 \qquad \Gamma_2 \vdash_{n_2} t_2 : \tau_2}{\max(\Gamma_1, \Gamma_2) \vdash_{\max(n_1,n_2)} (t_1, t_2) : \tau_1 \times \tau_2}\ \textsc{pair}$$

$$\frac{\Gamma_1 \vdash_{n_1} t : b \qquad \Gamma_2 \vdash_{n_2} t_i : b}{\max(n_2 + \Gamma_1, \Gamma_2) \vdash_{(n_1+n_2)} \mathsf{case}\ t\ \mathsf{of}\ \{c_i \Rightarrow t_i\}_{c_i \in b} : b}\ \textsc{case-const}$$

$$\frac{\Gamma_1 \vdash_{n_1} t : b \qquad \Gamma_2 \vdash_{n_2} t_i : query}{\max(\Gamma_1, \Gamma_2) \vdash_{(n_1+n_2)} \mathsf{case}\ t\ \mathsf{of}\ \{c_i \Rightarrow t_i\}_{c_i \in b} : query}\ \textsc{case-query}$$

$$\frac{i :: \mathbb{N}; \Gamma \vdash_n t : \tau \qquad i \notin \mathsf{FIV}(\Gamma)}{\Gamma \vdash_n \Lambda.t : \forall i{::}\mathbb{N}.\tau}\ \textsc{iabs} \qquad \frac{\Gamma \vdash_n t : \forall i{::}\mathbb{N}.\tau \qquad \vdash I :: \mathbb{N}}{\Gamma \vdash_n t\,[] : \tau\{I/i\}}\ \textsc{iapp}$$

$$\frac{\Gamma \vdash_n t : \tau \qquad \Gamma' \sqsubseteq \Gamma \qquad \vDash n \le n' \qquad \tau \sqsubseteq \tau'}{\Gamma' \vdash_{n'} t : \tau'}\ \textsc{sub}$$

Figure 1: Typing judgment

$\boxed{t \Downarrow^m v}$

$$\frac{}{c \Downarrow^0 c}\ \text{E-CONST} \qquad \frac{}{q \Downarrow^0 q}\ \text{E-QUERY} \qquad \frac{}{\lambda x.t \Downarrow^0 \lambda x.t}\ \text{E-ABS}$$

$$\frac{}{!t \Downarrow^0 !t}\ \text{E-BANG} \qquad \frac{t_1 \Downarrow^{m_1} v_1 \qquad t_2 \Downarrow^{m_2} v_2}{(t_1, t_2) \Downarrow^{\max(m_1, m_2)} (v_1, v_2)}\ \text{E-PAIR}$$

$$\frac{t_1 \Downarrow^{m_1} \lambda x.t \qquad t_2 \Downarrow^{m_2} v \qquad t[v/x] \Downarrow^{m_3} v'}{t_1\ t_2 \Downarrow^{\max(m_1, m_2)+m_3} v'}\ \text{E-APP}$$

$$\boxed{\frac{t_1 \Downarrow^{m_1} !t_3 \qquad t_3 \Downarrow^{m_2} v' \qquad t_2[v'/x] \Downarrow^{m_3} v}{\mathsf{let}\ !x = t_1\ \mathsf{in}\ t_2 \Downarrow^{\max(m_1+m_2, m_3)} v}\ \text{E-LET-BANG}}$$

$$\frac{t \Downarrow^{m_1} (v_1, v_2) \qquad t'[v_1/x_1][v_2/x_2] \Downarrow^{m2} v}{\mathsf{let}\ (x_1, x_2) = t\ \mathsf{in}\ t' \Downarrow^{\max(m_1, m_2)} v}\ \text{E-LET-P}$$

$$\frac{t \Downarrow^m v \qquad t_i \Downarrow^{m_i} v_i}{\mathsf{case}\ t\ \mathsf{of}\ \{c_i \Rightarrow t_i\}_{c_i \in b} \Downarrow^{m+\max(m_i)} \mathsf{case}\ v\ \mathsf{of}\ \{c_i \Rightarrow v_i\}_{c_i \in b}}\ \text{E-CASE} \qquad \frac{}{\mathsf{fix}\, f(x).t \Downarrow^0 \mathsf{fix}\, f(x).t}\ \text{E-FIX}$$

$$\frac{}{x \Downarrow^0 x}\ \text{E-X} \qquad \frac{}{\Lambda.t \Downarrow^0 \Lambda.t}\ \text{E-ILAM} \qquad \frac{t \Downarrow^m \Lambda.t'}{t[] \Downarrow^m t'}\ \text{E-IAPP} \qquad \frac{t \Downarrow^m v \qquad M(v) \Downarrow^1 v'}{M(t) \Downarrow^{m+1} v'}\ \text{E-MECH}$$

Figure 2: Evaluation Rules

$$\begin{aligned}
\llbracket \tau \rrbracket_\epsilon &= \{e \mid \exists v. e \Downarrow v \wedge v \in \llbracket \tau \rrbracket_v\} \\
\llbracket b \rrbracket_v &= \{v \mid v = T_b\} \\
\llbracket query \rrbracket_v &= \{v \mid v = T_{query}\} \\
\llbracket \tau_1 \to \tau_2 \rrbracket_v &= \{\lambda x.t \mid \forall v \in \llbracket \tau \rrbracket_v . t[v/x] \in \llbracket \tau_2 \rrbracket_\epsilon\} \\
\llbracket\, !_n \tau \rrbracket_v &= \{!t \mid t \in \llbracket \tau \rrbracket_\epsilon\} \\
\llbracket \forall i{::}\mathbb{N}.\tau \rrbracket_v &= \{\Lambda.t \mid \forall I. \vdash i :: \mathbb{N}. t[I/i] \in \llbracket \tau \rrbracket_\epsilon\} \\
\llbracket \tau_1 * \tau_2 \rrbracket_v &= \{(v_1, v_2) \mid v_1 \in \llbracket \tau_1 \rrbracket_v \wedge v_2 \in \llbracket \tau_2 \rrbracket_v\} \\
\llbracket \cdot \rrbracket &= \{\emptyset\} \\
\llbracket \Gamma, x : [\tau]_p \rrbracket &= \{\gamma[x \to v] \mid v \in \llbracket \tau \rrbracket_v \wedge \gamma \in \llbracket \Gamma \rrbracket\} \\
\boxed{\llbracket \Gamma, x : [\tau]_p \rrbracket} &= \{\gamma[x \to v] \mid v \in \llbracket !_p \tau \rrbracket_v \wedge \gamma \in \llbracket \Gamma \rrbracket\} \\
\llbracket \Gamma, x : \tau \rrbracket &= \{\gamma[x \to v] \mid v \in \llbracket \tau \rrbracket_v \wedge \gamma \in \llbracket \Gamma \rrbracket\} \\
\gamma \vDash \Gamma &\triangleq dom(\gamma) = dom(\Gamma) \wedge \forall x \in dom(\Gamma). \gamma(x) \in \llbracket \Gamma(x) \rrbracket_v
\end{aligned}$$

Figure 3: denotations

2 Attempt 2: Trace-based effect system

Traces A trace T is a representation of the big-step derivation of an expression's evaluation.

The *adaptivity* of a trace T, $\mathtt{adap}(T)$, which means the maximum number of nested δs in T.

The *depth of variable* x in trace T, written $\mathtt{depth}_x(T)$, which is the maximum number of δs in any path leading from the root of T to an occurence of x (at a leaf),.

$$
\begin{array}{llcl}
\text{Expr.} & e & ::= & x \mid e_1\, e_2 \mid \mathtt{fix}\, f(x:\tau).e \mid (e_1, e_2) \mid \mathtt{fst}(e) \mid \mathtt{snd}(e) \mid \\
& & & \mathtt{if}(e_1, e_2, e_3) \mid c \mid \delta(e) \mid \mathtt{let}\, x : q = e_1\, \mathtt{in}\, e_2 \mid \mathtt{nil} \mid \mathtt{cons}(e_1, e_2) \\
\text{Value} & v & ::= & c \mid (\mathtt{fix}\, f(x:\tau).e, \theta) \mid (v_1, v_2) \mid \mathtt{nil} \mid \mathtt{cons}(v_1, v_2) \mid \\
\text{Environment} & \theta & ::= & x_1 \mapsto v_1, \ldots, x_n \mapsto v_n
\end{array}
$$

$$
\begin{array}{lcl}
\text{Trace} \quad T & ::= & (x, \theta) \mid T_1\, T_2 \rhd \mathtt{fix}\, f(x).T_3 \mid (\mathtt{fix}\, f(x:\tau).e, \theta) \mid (T_1, T_2) \mid \mathtt{fst}(T) \mid \\
& & \mathtt{snd}(T) \mid \mathtt{true} \mid \mathtt{false} \mid \mathtt{if}^{\mathtt{t}}(T_b, T_t) \mid \mathtt{if}^{\mathtt{f}}(T_b, T_f) \mid c \mid \delta(T)
\end{array}
$$

2.1 Challenge (Couterexample) in this setting

1 adaptivity is not precise. The definition of the max number of nested δs is not very accurate, especially the way it handles the application. Another way of understanding adaptivity is not only the occurence of δ, but the times the program accesses the database (δ). For instance, $\lambda x.(\mathtt{if}\,(x < 10)$ then x else $x)\delta(v)$ and $(\mathtt{if}\,(\delta(v) < 10)$ then $\delta(v)$ else $\delta(v))$. should distinguish the two definition of adaptivity.

2 This operational semantics has trouble to give the reasonable trace to nested lambda. Consider $\lambda x.\lambda y.(x y)$. and its corresponding application. consider $(\lambda x.\lambda y.(x y))\,\delta(v)\,\delta(v)$, its trace equals to 0

3 The typing system is not consistent for α renaming.

$$\frac{}{\theta, x \to \theta(x), (x,\theta)} \qquad \frac{}{\theta, c \to c, c} \qquad \frac{}{\theta, \mathtt{fix}\, f(x:\tau).e \to (\mathtt{fix}\, f(:\tau).e,\theta), (\mathtt{fix}\, f(x:\tau).e,\theta)}$$

$$\frac{\begin{array}{cc}\theta, e_1 \to v_1, T_1 & v_1 = (\mathtt{fix}\, f(x:\tau).e,\theta') \\ \theta, e_2 \to v_2, T_2 & \theta'[f \mapsto v_1, x \mapsto v_2], e \to v, T\end{array}}{\theta, e_1\ e_2 \to v, T_1\ T_2 \rhd \mathtt{fix}\, f(x).T} \qquad \frac{\theta, e_1 \to v_1, T_1 \qquad \theta, e_2 \to v_2, T_2}{\theta, (e_1, e_2) \to (v_1, v_2), (T_1, T_2)}$$

$$\frac{\theta, e \to (v_1, v_2), T}{\theta, \mathtt{fst}(e) \to v_1, \mathtt{fst}(T)} \qquad \frac{\theta, e \to (v_1, v_2), T}{\theta, \mathtt{snd}(e) \to v_2, \mathtt{snd}(T)} \qquad \frac{\theta, e \to \mathtt{true}, T \qquad \theta, e_1 \to v, T_1}{\theta, \mathtt{if}(e, e_1, e_2) \to v, \mathtt{if}^{\mathtt{t}}(T, T_1)}$$

$$\frac{\theta, e \to \mathtt{false}, T \qquad \theta, e_2 \to v, T_2}{\theta, \mathtt{if}(e, e_1, e_2) \to v, \mathtt{if}^{\mathtt{f}}(T, T_2)} \qquad \frac{\theta, e \to v, T \qquad \delta(v) = v'}{\theta, \delta(e) \to v', \delta(T)}$$

Figure 4: Big-step semantics with provenance

$\boxed{\texttt{adap} : \text{Traces} \to \mathbb{N}}$

$$
\begin{array}{lcl}
\texttt{adap}((x,\theta)) & = & 0 \\
\texttt{adap}(T_1\ T_2 \rhd \texttt{fix}\, f(x).T_3) & = & \texttt{adap}(T_1) + \max(\texttt{adap}(T_3), \texttt{adap}(T_2) + \texttt{depth}_x(T_3)) \\
\texttt{adap}((\texttt{fix}\, f(x:\tau).e,\theta)) & = & 0 \\
\texttt{adap}((T_1, T_2)) & = & \max(\texttt{adap}(T_1), \texttt{adap}(T_2)) \\
\texttt{adap}(\texttt{fst}(T)) & = & \texttt{adap}(T) \\
\texttt{adap}(\texttt{snd}(T)) & = & \texttt{adap}(T) \\
\texttt{adap}(\texttt{true}) & = & 0 \\
\texttt{adap}(\texttt{false}) & = & 0 \\
\texttt{adap}(\texttt{if}^{\texttt{t}}(T_b, T_t)) & = & \texttt{adap}(T_b) + \texttt{adap}(T_t) \\
\texttt{adap}(\texttt{if}^{\texttt{f}}(T_b, T_f)) & = & \texttt{adap}(T_b) + \texttt{adap}(T_f) \\
\texttt{adap}(c) & = & 0 \\
\texttt{adap}(\delta(T)) & = & 1 + \texttt{adap}(T) \\
 & & +\mathsf{MAX}_{v\in\tau}\Big(\max\big(\texttt{adap}(T_3(v)), \texttt{depth}_x(T_3(v))\big)\Big) \\
 & \text{where} & v_1 = (\texttt{fix}\, f(x:\tau).e,\theta) = \mathsf{extract}(T) \\
 & & \wedge\, \theta[f \mapsto v_1, x \mapsto v], e \Downarrow^{v'}, T_3(v)
\end{array}
$$

$\boxed{\texttt{depth}_x : \text{Traces} \to \mathbb{N}_\perp}$

$$
\begin{array}{lcl}
\texttt{depth}_x((y,\theta)) & = & \begin{cases} 0 & \text{if } x = y \\ \perp & \text{if } x \neq y \end{cases} \\
\texttt{depth}_x(T_1\ T_2 \rhd \texttt{fix}\, f(y).T_3) & = & \max(\texttt{depth}_x(T_1), \\
 & & \texttt{adap}(T_1) + \max(\texttt{depth}_x(T_3), \texttt{depth}_x(T_2) + \texttt{depth}_y(T_3))) \\
\texttt{depth}_x((\texttt{fix}\, f(y:\tau).e,\theta)) & = & \perp \\
\texttt{depth}_x((T_1, T_2)) & = & \max(\texttt{depth}_x(T_1), \texttt{depth}_x(T_2)) \\
\texttt{depth}_x(\texttt{fst}(T)) & = & \texttt{depth}_x(T) \\
\texttt{depth}_x(\texttt{snd}(T)) & = & \texttt{depth}_x(T) \\
\texttt{depth}_x(\texttt{true}) & = & \perp \\
\texttt{depth}_x(\texttt{false}) & = & \perp \\
\texttt{depth}_x(\texttt{if}^{\texttt{t}}(T_b, T_t)) & = & \max(\texttt{depth}_x(T_b), \texttt{adap}(T_b) + \texttt{depth}_x(T_t)) \\
\texttt{depth}_x(\texttt{if}^{\texttt{t}}(T_b, T_f)) & = & \max(\texttt{depth}_x(T_b), \texttt{adap}(T_b) + \texttt{depth}_x(T_f)) \\
\texttt{depth}_x(c) & = & \perp \\
\texttt{depth}_x(\delta(T)) & = & 1 + \max(\texttt{depth}_x(T), \\
 & & \texttt{adap}(T) + \mathsf{MAX}_{v\in\tau}\Big(\max(\texttt{depth}_x(T_3(v)), \perp)\Big)) \\
 & \text{where} & v_1 = (\texttt{fix}\, f(x:\tau).e,\theta) = \mathsf{extract}(T) \\
 & & \wedge\, \theta[f \mapsto v_1, x \mapsto v], e \Downarrow^{v'}, T_3(v)
\end{array}
$$

Figure 5: Adaptivity of a trace and depth of variable x in a trace

3 Attempt 3: Linear type based 2

$$\begin{array}{lcll} \text{Expr.} & e & ::= & x \mid e_1\ e_2 \mid \lambda x.e \\ & & & \texttt{true} \mid \texttt{false} \mid \texttt{if} e \text{ then } e_2 \text{ else } e_3 \mid c \mid \delta(e) \\ \text{Environment} & \theta & ::= & x_1 \mapsto (v_1, R_1), \ldots, x_n \mapsto (v_n, R_n) \end{array}$$

$$\begin{array}{lcll} \text{Index Term} & I, Z & ::= & i \mid n \\ \text{Linear type} & \tau & ::= & A \multimap Z\tau \mid \mathsf{b} \mid \texttt{bool} \\ \text{Nonlinear Type} & A & ::= & !_I \tau \\ \text{Typing context} & \Gamma & ::= & x_1 : A_1, \ldots, x_n : A_n \end{array}$$

$$\frac{}{v, \theta \Downarrow^0 v, \theta}\ \textsf{val} \qquad \frac{\theta(x) = (v, \theta_1, R)}{x, \theta \Downarrow^R v, \theta_1}\ \textsf{var} \qquad \frac{}{c, \theta \Downarrow^0 c, \theta}\ \textsf{const} \qquad \frac{}{\lambda x.e, \theta \Downarrow^0 \lambda x.e, \theta}\ \textsf{lambda}$$

$$\frac{e_1, \theta_1 \Downarrow^{R_1} \lambda x.e, \theta_1' \qquad e_2, \theta_2 \Downarrow^{R_2} v_2, \theta_2' \qquad \text{fresh } x' \qquad e[x'/x], \theta_1'[x' \to (v_2, \theta_2', R_2)] \Downarrow^{R_3} v, \theta_3}{e_1\ e_2, (\theta_1 \uplus \theta_2) \Downarrow^{R_1 + R_3} v, \theta_3}\ \textsf{app}$$

$$\frac{e, \theta \Downarrow^R v, \theta_1 \qquad \delta(v\theta) = v' \qquad FV(v') = \emptyset}{\delta(e), \theta \Downarrow^{R+1} v, \theta_1}\ \textsf{delta} \qquad \frac{e, \theta \Downarrow^R \texttt{false}, \theta' \qquad e_2, \theta \Downarrow^{R_2} v_2, \theta_2}{\texttt{if} e \text{ then } e_1 \text{ else } e_2, \theta \Downarrow^{R+R_2} v_2, \theta_2}\ \textsf{if-f}$$

$$\frac{e, \theta \Downarrow^R \texttt{true}, \theta' \qquad e_1, \theta \Downarrow^{R_1} v_1, \theta_1}{\texttt{if} e \text{ then } e_1 \text{ else } e_2, \theta \Downarrow^{R+R_1} v_1, \theta_1}\ \textsf{if-t}$$

$$\begin{array}{rcl} \theta_1 \uplus \emptyset & \triangleq & \theta_1 \\ \emptyset \uplus \theta_2 & \triangleq & \theta_2 \end{array}$$

Figure 6: Big-step semantics

Typable Approach By Weihao

$$\begin{array}{ll} F(e, \phi) & where \quad \phi(x_i) = (I_i, R_i, Z_i) \\ F(x, \phi) & = \sum_{x_i \in \mathsf{FV}(x)} I_i \times (R_i + Z_i) \\ F(\lambda x.e, \phi) & = \sum_{x_i \in \mathsf{FV}(\lambda x.e)} I_i \times (R_i + Z_i) \\ F(\delta(e), \phi) & = \sum_{x_i \in \mathsf{FV}(\delta(e))} I_i \times (R_i + Z_i) \\ F(c, \phi) & = 0 \\ F(e_1\ e_2, \phi) & = F(e_1, \phi) + F(e_2, \phi) \\ F(\texttt{if} e \text{ then } e_1 \text{ else } e_2, \phi) & = F(e, \phi) + \max(F(e_1, \phi), F(e_2, \phi)) \end{array}$$

Definition 1 (Typable). *A closure $(e, [x_1 \to (v_1, \theta_1, R_1), \ldots, x_i \to (v_i, \theta_i, R_i)])$ is typable with type τ and adaptivity J if exists k_i*

$$x_1 :!_{I_1} \tau_1, \ldots, !_{I_i} \tau_i \vdash_Z e : \tau$$

and each closure (v_i, θ_i) is also typable with type $!_{I_i} \tau_i$ and adaptivity Z_i, $\phi = [x_1 \to (I_1, R_1, Z_1), \ldots, x_i \to (I_i, R_i, Z_i)]$, $J = Z + F(e, \phi)$.

$$\frac{\theta(x) = (v,\theta',R) \qquad \vdash_Z (v,\theta') : \tau}{\vdash_{R+Z} (x,\theta) : \tau}\ \textbf{C-Ax} \qquad\qquad \frac{}{\vdash_0 (c,\theta) : \mathtt{b}}\ \textbf{C-const}$$

$$\frac{\vdash_{Z'} (v',\theta') : \tau_1 \qquad \text{fresh } x' \quad \forall R' \qquad \vdash_{S+I\times(R'+Z')+Z} (e[x'/x], \theta[x' \to (v',\theta',R')]) : \tau_2}{\vdash_S (\lambda x.e,\theta) :!_I \tau_1 \multimap Z\tau_2}\ \textbf{C-lambda}$$

$$\frac{\vdash_{Z_1} (e_1,\theta_1) :!_I \tau_1 \multimap Z\tau_2 \qquad \vdash_{Z_2} (e_2,\theta_2) : \tau_1}{\vdash_{Z_1+I\times Z_2+Z} (e_1\ e_2, \theta_1 \uplus \theta_2) : \tau_2}\ \textbf{C-app} \qquad\qquad \frac{\vdash_Z (e,\theta) : \mathtt{b}}{\vdash_{1+Z} (\delta(e),\theta) : \mathtt{b}}\ \textbf{C-delta}$$

$$\begin{array}{lll} \theta & \triangleq (x_i \to (v_i,\theta_i,R_i)) & i \in \mathbb{N} \\ (x_i :!_I \tau_i), \Gamma \vDash (x_i \to (v_i,\theta_i,R_i)) \uplus \theta & \triangleq\ \vdash_{\_} (v_i,\theta_i) : \tau_i & \wedge\, \Gamma \vDash \theta \end{array}$$

Figure 7: Typing rules, configure

$$\begin{array}{lcl} [[\mathtt{b}]]_V & = & \{(c,\theta,Z)\} \\ [[!_k\tau]]_V & = & \{(v,\theta,Z) | (v,\theta,Z) \in [[\tau]]_V\} \\ [[!_k\tau_1 \multimap Z\tau_2]]_V & = & \{(\lambda x.e,\theta,Z_1) \mid \forall v',\theta',Z'.(v',\theta',Z') \in [[!_k\tau_1]]_V. \\ & & \implies \text{fresh } x' \wedge \\ & & \forall R.(e[x'/x],\theta[x' \mapsto (v',\theta',R)]) \in [[\tau_2]]_E^{Z_1+Z+I\times(R+Z')}\} \\ & & \\ [[\tau]]_E^Z & = & \{(e,\theta) \mid (e,\theta \Downarrow^R v,\theta') \\ & & \qquad \implies R \le Z \wedge (v,\theta',Z-R) \in [[\tau]]_V\} \end{array}$$

Figure 8: Logical relation without step-indexing

Definition 2 (ClosedClosure)**.** *A closure* (e,θ) *is closed if* $\mathsf{FV}(e) \subseteq \mathtt{dom}(\theta)$.

Lemma 1 (programTypable)**.** *If* $\vdash_Z e : \tau$*, then* $(e,\emptyset)$ *is typable with* τ *and adaptivity* Z*.*

Lemma 2 (TypableMono)**.** *If a closure is* D *is typable with* τ *and* Z*, and* $Z \le Z'$*, then* D *is typable with* τ *and* Z'*.*

Lemma 3 (TypableSoundness)**.** *If a closure* D *is typable with* τ *and* J*, and* $D \Downarrow^R E$*, then closure* E *is typable with* τ *and adaptivity* $J - R$*.*

Typable Approach By Marco

Definition 3 (Typable Closures)**.** *Let* $\theta = [x_1 \to (v_1,\theta_1,R_1),\ldots,x_n \to (v_n,\theta_n,R_n)]$*. The closure* (e,θ) *is typable with type* τ *and adaptivity* J *if:*

1. $x_1 :!_{k_1}\tau_1,\ldots,!_{k_i}\tau_i \vdash_Z e : \tau$*, for some types* $!_{k_i}\tau_i$ *for* $(1 \le i \le n)$*,*
2. *each closure* (v_i,θ_i) *for* $(1 \le i \le n)$ *is typable with type* $!_{k_i}\tau_i$ *and adaptivity* Z_i*,*
3. $J = Z + \sum_{(v_i,\theta_i,S_i)\in\theta} k_i \times (R_i + Z_i)$*.*

To justify why we chose $\sum$ in the third clause above it is worth to consider the following configuration:

$$[x \mapsto (\lambda u.\lambda w.\delta(u) + \delta(w), [\,], 0), y :\mapsto (v, [\,], 2)], x\, y\, y$$

Lemma 4 (Soundness)**.** *If a closure D is typable with type τ and adaptivity J, and $D \Downarrow^R E$, then the closure E is typable with type τ and adaptivity I, where $I + R \le J$.*

FAILURE: [[Soundness is unable to be proven:
In the semantics app rule: $R_1 + R_3$ **adaptivity is numerically added.**
In the typing-configuration (typable closure): $J = Z + \sum_{(v_i, \theta_i, S_i) \in \theta} k_i \times (R_i + Z_i)$**, adaptivity adding is by** $\max$**.**
The adpativity in typing rule cannot bound the adaptivity in semantics. We need to have a better understanding on the adptivity flow]]

4 Attemp4: Call by need style

$$
\begin{array}{llll}
\text{Expr.} & e & ::= & c \mid x \mid e_1\ e_2 \mid \lambda x.e \mid \ \mathsf{let} x = e_1 \ \mathsf{in}\ e_2 \ \mid (e_1, e_2) \mid \mathtt{fst} e \mid \mathtt{snd} e \\
 & & & \mathtt{true} \mid \mathtt{false} \mid \mathtt{if}(e, e_2, e_3) \mid \delta(e) \\
\text{Value.} & v & ::= & c \mid \lambda x.e \mid \ (v_1, v_2) \\
\text{Environment} & \theta & ::= & \epsilon \mid \theta, [x \mapsto (v, R)]
\end{array}
$$

$$
\frac{}{\theta, v \Downarrow^0 \theta, v}\ \textsf{val} \qquad \frac{}{\theta[x->(v,R)], x \Downarrow^R \theta[x->(v,0)], v}\ \textsf{var} \qquad \frac{}{\theta, c \Downarrow^0 \theta, c}\ \textsf{const}
$$

$$
\frac{}{\theta, \lambda x.e \Downarrow^0 \theta, \lambda x.e}\ \textsf{lambda}
$$

$$
\frac{\theta, e_1 \Downarrow^{R_1} \theta_1, \lambda x.e \qquad \theta_1, e_2 \Downarrow^{R_2} \theta_2, v_2 \qquad \text{fresh } x' \qquad \theta_2[x' \to (v_2, R_2)], e[x'/x], \Downarrow^{R_3} \theta_3, v_3}{\theta, e_1\ e_2 \Downarrow^{R_1+R_3} \theta_3, v_3}\ \textsf{app}
$$

$$
\frac{\theta, e_1 \Downarrow^{R_1} \theta_1, v_1 \qquad \text{fresh } x' \qquad \theta_1[x' \to (v_1, R_1)], e_2[x'/x], \Downarrow^{R_2} \theta_2, v_2}{\theta, \mathsf{let} x = e_1 \ \mathsf{in}\ e_2 \Downarrow^{R_2} \theta_2, v_2}\ \textsf{let}
$$

$$
\frac{\theta, e \Downarrow^R \theta_1, \lambda x.e_1 \qquad \delta(\lambda x.e') = v_1}{\theta, \delta(e) \Downarrow^{R+1} \theta_1, v_1}\ \textsf{delta} \qquad \frac{\theta, e_1 \Downarrow^{R_1} \theta_1, \mathtt{false} \qquad \theta_1, e_2 \Downarrow^{R_2} \theta_2, v_2}{\theta, \mathtt{if}(e_1, e_2, e_3) \Downarrow^{R_1+R_2} \theta_2, v_2}\ \textsf{if-f}
$$

$$
\frac{\theta, e_1 \Downarrow^{R_1} \theta_1, \mathtt{true} \qquad \theta_1, e_3 \Downarrow^{R_3} \theta_3, v_3}{\theta, \mathtt{if}(e_1, e_2, e_3) \Downarrow^{R_1+R_2} \theta_3, v_3}\ \textsf{if-t} \qquad \frac{\theta, e_1 \Downarrow^{R_1} \theta_1, v_1 \qquad \theta, e_2 \Downarrow^{R_2} \theta_2, v_2}{\theta, (e_1, e_2) \Downarrow^{\max(R_1, R_2)} \theta_1 \uplus \theta_2, (v_1, v_2)}\ \textsf{prod}
$$

$$
\frac{\theta, e \Downarrow^R \theta_1, (v_1, v_2)}{\theta, \mathtt{fst}(e) \Downarrow^R \theta_1, v_1}\ \textsf{projl} \qquad \frac{\theta, e \Downarrow^R \theta_1, (v_1, v_2)}{\theta, \mathtt{snd}(e) \Downarrow^R \theta_1, v_2}\ \textsf{projr} \qquad \frac{\theta, e_1 \Downarrow^{R_1} \theta_1, v_1 \qquad \theta, e_2 \Downarrow^{R_2} \theta_2, v_2}{\theta, e_1 < e_2 \Downarrow^{\max(R_1, R_2)} \theta_1 \uplus \theta_2, v_1 < v_2}\ \textsf{bop}
$$

$$
\begin{array}{lll}
\theta_1 \uplus \epsilon & \triangleq & \theta_1 \\
\epsilon \uplus \theta_2 & \triangleq & \theta_2 \\
\theta_1[x \to (v, R)] \uplus \theta_2[x \to (v, 0)] & \triangleq & (\theta_1 \uplus \theta_2)[x \to (v, 0)] \\
\theta_1[x \to (v, 0)] \uplus \theta_2[x \to (v, R)] & \triangleq & (\theta_1 \uplus \theta_2)[x \to (v, 0)] \\
\theta_1[x \to (v, R)] \uplus \theta_2 & \triangleq & (\theta_1 \uplus \theta_2)[x \to (v, R)] \quad x \notin dom(\theta_2) \\
\theta_1 \uplus \theta_2[x \to (v, R)] & \triangleq & (\theta_1 \uplus \theta_2)[x \to (v, R)] \quad x \notin dom(\theta_1)
\end{array}
$$

Figure 9: Big-step semantics

4.1 Example

1. [], $\lambda x.(x, x)\ \delta(\lambda z.e) \Downarrow^{0+1} [x' \to (v', 0)], v'$ where $v' = \delta(\lambda z.e)$.
2. [], $\lambda x.\lambda y.(x, y)\ \delta(\lambda z.e)\ v$
3. $[x \to (v_x, 1), y \to (v_y, 2)], \big((x, \delta(y)), (\delta(x), y)\big)$

4. $[x \to (v_x, 1), y \to (v_y, 2)], \texttt{if}\big(\delta(x), \delta(y), y\big)$
5. $[x \to (v_x, 1), y \to (v_y, 2)], \texttt{if}\big(\delta(y), \delta(y), x\big)$
6. $[], \lambda x.\lambda y.\texttt{if}\big(x < y, x + y, \delta(\lambda m.m + x)\big)\, \delta(\lambda z_1.e_1)\, \delta(\lambda z_2.e_2)$
7. $[x \to (v_x, 5), y \to (v_y, 6)], \mathsf{let} z = \lambda m.x + y + m\ \mathsf{in}\ \delta(z)$