Adaptivity analysis

Abstract

An adaptive data analysis is based on multiple queries over a data set, in which some queries rely on the results of some other queries. The error of each query is usually controllable and bound independently, but the error can propagate through the chain of different queries and bring to high generalization error. To address this issue, data analysts are adopting different mechanisms in their algorithms, such as Gaussian mechanism, etc. To utilize these mechanisms in the best way one needs to understand the depth of chain of queries that one can generate in a data analysis. In this work, we define a programming language which can provide, through its type system, an upper bound on the adaptivity depth (the length of the longest chain of queries) of a program implementing an adaptive data analysis. We show how this language can help to analyze the generalization error of two data analyses with different adaptivity structures.

1 Everything Else

Adaptivity Adaptivity is a measure of the nesting depth of a mechanism. To represent this depth, we use extended natural numbers. Define $\mathbb{N}_\perp = \mathbb{N} \cup \{\perp\}$, where $\perp$ is a special symbol and $\mathbb{N}_\perp^\infty = \mathbb{N}_\perp \cup \{\infty\}$. We use Z, m to range over $\mathbb{N}$, s, t to range over $\mathbb{N}_\perp$, and q, r to range over $\mathbb{N}_\perp^\infty$.

The functions max and $+$, and the order $\leq$ on natural numbers extend to $\mathbb{N}_\perp^\infty$ in the natural way:

$$
\begin{array}{lcll}
\max(\perp, q) & = & q & \\
\max(q, \perp) & = & q & \\
\max(\infty, q) & = & \infty & \\
\max(q, \infty) & = & \infty & \\
 & & & \\
\perp + q & = & \perp & \\
q + \perp & = & \perp & \\
\infty + q & = & \infty & \text{if } q \neq \perp \\
q + \infty & = & \infty & \text{if } q \neq \perp \\
 & & & \\
\perp \leq q & & & \\
q \leq \infty & & &
\end{array}
$$

One can think of $\bot$ as $-\infty$, with the special proviso that, here, $-\infty + \infty$ is specifically defined to be $-\infty$.

Language Expressions are shown below. c denotes constants (of some base type $\mathtt{b}$, which may, for example, be reals or rational numbers). δ represents a primitive operation (such as a mechanism), which determines adaptivity. For simplicity, we assume that δ can only have type $\mathtt{b} \to \mathtt{bool}$. We make environments explicit in closures. This is needed for the tracing semantics later.

$$
\begin{array}{llll}
\text{Expr.} & e & ::= & x \mid e_1\ e_2 \mid \mathtt{fix}\, f(x:\tau).e \mid (e_1, e_2) \mid \mathtt{fst}(e) \mid \mathtt{snd}(e) \mid \\
 & & & \mathtt{true} \mid \mathtt{false} \mid \mathtt{if}(e_1, e_2, e_3) \mid c \mid \delta(e) \mid \Lambda.e \mid e\,[] \\
 & & & \mid \mathtt{let}\, x = e_1\, \mathtt{in}\, e_2 \mid \mathtt{nil} \mid \mathtt{cons}(e_1, e_2) \\
 & & & \mid \quad \mathtt{bernoulli}\ e \mid \mathtt{uniform}\ e_1\ e_2 \\
\text{Value} & v & ::= & \mathtt{true} \mid \mathtt{false} \mid c \mid (\mathtt{fix}\, f(x:\tau).e, \theta) \mid (v_1, v_2) \mid \mathtt{nil} \mid \mathtt{cons}(v_1, v_2) \mid \\
 & & & (\Lambda.e, \theta) \\
\text{Environment} & \theta & ::= & x_1 \mapsto (v_1, T_1), \ldots, x_n \mapsto (v_n, T_n)
\end{array}
$$

2 Tracing operational semantics and adaptivity

Traces A trace T is a representation of the big-step derivation of an expression's evaluation. Our big-step semantics output a trace. We use traces to define the adaptivity of a run. Our notion of traces and the tracing semantics is taken from [1, Section 4], but we omit their "holes" for which we have no need. The construct $T_1\ T_2 \triangleright \mathtt{fix}\, f(x).T_3$ records a trace of function application. T_1 is the trace of the head, T_2 the trace of the argument and T_3 is the trace of the function body. f and x are bound in T_3.

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

Big-step tracing semantics The big-step, tracing semantics $\theta, e \Downarrow^{\mathsf{n}} v, T$ computes a value v and a trace T from an expression e and an enviroment θ which maps the free variables of e to *closed* values. The rules, taken from [1], are shown in Figure 2. Some salient points:

- Erasing the traces from the semantics yields a standard big-step semantics.
- The trace of a primitive application $\delta(e)$ records that δ was applied to the trace of e. This enables us to define adaptivity from a trace later.
- The trace of a variable x is x. This way traces record where substitutions occur and, hence, variable dependencies. This is also needed for defining adaptivity.

$$\frac{}{\theta[x \to (v, R)], x \Downarrow^R v}\ \textsf{var1} \qquad \frac{x \notin dom(\theta)}{\theta, x \Downarrow^0 x}\ \textsf{var2} \qquad \frac{}{\theta, \texttt{true} \Downarrow^0 \texttt{true}}$$

$$\frac{}{\theta, \texttt{false} \Downarrow^0 \texttt{false}} \qquad \frac{\theta, e \Downarrow^K c}{\theta, \texttt{bernoulli}\ e \Downarrow^K c}\ \textsf{bernoulli}$$

$$\frac{\theta, e_1 \Downarrow^R c \qquad \theta, e_2 \Downarrow^S c}{\theta, \texttt{uniform}\ e_1\ e_2 \Downarrow^{R+S} c}\ \textsf{uniform} \qquad \frac{\theta, e \Downarrow^R v}{\theta, \texttt{fix}\, f(x).e \Downarrow^R \texttt{fix}\, f(x).v,}\ \textsf{fix}$$

$$\frac{\begin{array}{c}\theta, e_1 \Downarrow^K \texttt{fix}\, f(x).e \\ \theta, e_2 \Downarrow^R v_2 \qquad \theta[f \to (\texttt{fix}\, f(x).e, K), x \to (v_2, R)], e \Downarrow^S v\end{array}}{\theta, e_1\ e_2 \Downarrow^{K+S} v}\ \textsf{app1}$$

$$\frac{\theta, e_1 \Downarrow^K v_1 \not\equiv \texttt{fix}\, f(x).e \qquad \theta, e_2 \Downarrow^R v_2}{\theta, e_1\ e_2 \Downarrow^{K+R} v_1\ v_2}\ \textsf{app2}$$

$$\frac{\theta, e_1 \Downarrow^R v_1 \qquad \theta, e_2 \Downarrow^S v_2}{\theta, (e_1, e_2) \Downarrow^{(R,S)} (v_1, v_2)}\ \textsf{pair} \qquad \frac{\theta, e \Downarrow^{(R_1,R_2)} (v_1, v_2)}{\theta, \texttt{fst}(e) \Downarrow^{R_1} v_1}\ \textsf{fst}$$

$$\frac{\theta, e \Downarrow^{(R_1,R_2)} (v_1, v_2), T}{\theta, \texttt{snd}(e) \Downarrow^{(R,)} v_2, \texttt{snd}(T)}\ \textsf{snd} \qquad \frac{\theta, e \Downarrow^R \texttt{true} \qquad \theta, e_1 \Downarrow^S v, T_1}{\theta, \texttt{if}(e, e_1, e_2) \Downarrow^{R+S} v}\ \textsf{if-true}$$

$$\frac{\theta, e \Downarrow^R \texttt{false} \qquad \theta, e_2 \Downarrow^S v}{\theta, \texttt{if}(e, e_1, e_2) \Downarrow^{R+S} v}\ \textsf{if-false}$$

$$\frac{\theta, e \Downarrow^R v' \qquad \delta(v') = v \qquad v' \in \mathsf{Vec}}{\theta, \delta(e) \Downarrow^{R+1} v, \delta(T)}\ \textsf{delta1} \qquad \frac{\theta, e \Downarrow^R v' \qquad v' \notin \mathsf{Vec}}{\theta, \delta(e) \Downarrow^R \delta(v'), \delta(T)}\ \textsf{delta2}$$

$$\frac{}{\theta, \texttt{nil} \Downarrow^0 \texttt{nil}, \texttt{nil}}\ \textsf{nil} \qquad \frac{\theta, e_1 \Downarrow^R v_1 \qquad \theta, e_2 \Downarrow^S v_2}{\theta, \texttt{cons}(e_1, e_2) \Downarrow^{\max(R,S)} \texttt{cons}(v_1, v_2)}\ \textsf{cons}$$

$$\frac{\theta, e_1 \Downarrow^R v_1 \qquad \theta[x \mapsto (v_1, R)], e_2 \Downarrow^S v}{\theta, \texttt{let}\, x = e_1\, \texttt{in}\, e_2 \Downarrow^S v}\ \textsf{let}$$

$$\frac{\theta, e \Downarrow^R v}{\theta, \Lambda.e \Downarrow^0 \Lambda.v,}\ \textsf{eilam} \qquad \frac{\theta, e \Downarrow^K (\Lambda.e') \qquad \theta, e' \Downarrow^S v}{\theta, e[] \Downarrow^{K+S} v, \texttt{IApp}(T_1, T_2)}\ \textsf{eiapp1}$$

$$\frac{\theta, e \Downarrow^K v \not\equiv (\Lambda.e')}{\theta, e[] \Downarrow^K v[]}\ \textsf{eiapp2} \qquad \frac{\theta, e \Downarrow^R v \not\equiv \mathbb{B}}{\theta, \texttt{if}(e, e_1, e_2) \Downarrow^R \texttt{if}(v, e_1, e_2)}\ \textsf{if}$$

$$\frac{\theta, e \Downarrow^R v}{\theta, \texttt{fst}(e) \Downarrow^{fst(R)} \texttt{fst}(v)}\ \textsf{fst1}$$

Figure 1: Big-step semantics

$$\frac{}{\theta, x \Downarrow^0 \theta_1(x), (x, \theta)} \qquad \frac{}{\theta, c \Downarrow^0 c, c} \qquad \frac{}{\theta, \texttt{true} \Downarrow^0 \texttt{true}, \texttt{true}}$$

$$\frac{}{\theta, \texttt{false} \Downarrow^0 \texttt{false}, \texttt{false}} \qquad \frac{\theta, e \Downarrow^{\mathsf{n}} c, T}{\theta, \texttt{bernoulli}\ e \Downarrow^{\mathsf{n}} c, \texttt{bernoulli}(T)}$$

$$\frac{\theta, e_1 \Downarrow^{\mathsf{n}_1} c, T_1 \qquad \theta, e_2 \Downarrow^{\mathsf{n}_2} c, T_2}{\theta, \texttt{uniform}\ e_1\ e_2 \Downarrow^{\mathsf{n}_1+\mathsf{n}_2} c, \texttt{uniform}(T_1, T_2)}$$

$$\frac{}{\theta, \texttt{fix}\, f(x).e \Downarrow^0 (\texttt{fix}\, f(x).e, \theta), (\texttt{fix}\, f(x).e, \theta)}$$

$$\frac{\begin{array}{c}\theta, e_1 \Downarrow^{\mathsf{n}_1} v_1, T_1 \qquad v_1 = (\texttt{fix}\, f(x:\tau).e, \theta') \\ \theta, e_2 \Downarrow^{\mathsf{n}_2} v_2, T_2 \qquad \theta'[f \mapsto (v_1, (\texttt{fix}\, f(x:\tau).e, \theta')), x \mapsto (v_2, T_2)], e \Downarrow^{\mathsf{n}_3} v, T\end{array}}{\theta, e_1\ e_2 \Downarrow^{\mathsf{n}_1+\mathsf{n}_2+\mathsf{n}_3} v, T_1\ T_2 \triangleright \texttt{fix}\, f(x).T}$$

$$\frac{\theta, e_1 \Downarrow^{\mathsf{n}_1} v_1, T_1 \qquad \theta, e_2 \Downarrow^{\mathsf{n}_2} v_2, T_2}{\theta, (e_1, e_2) \Downarrow^{\mathsf{n}_1+\mathsf{n}_2} (v_1, v_2), (T_1, T_2)} \qquad \frac{\theta, e \Downarrow^{\mathsf{n}} (v_1, v_2), T}{\theta, \texttt{fst}(e) \Downarrow^{\mathsf{n}} v_1, \texttt{fst}(T)}$$

$$\frac{\theta, e \Downarrow^{\mathsf{n}} (v_1, v_2), T}{\theta, \texttt{snd}(e) \Downarrow^{\mathsf{n}} v_2, \texttt{snd}(T)} \qquad \frac{\theta, e \Downarrow^{\mathsf{n}} \texttt{true}, T \qquad \theta, e_1 \Downarrow^{\mathsf{n}_1} v, T_1}{\theta, \texttt{if}(e, e_1, e_2) \Downarrow^{\mathsf{n}+\mathsf{n}_1} v, \texttt{if}^{\mathsf{t}}(T, T_1)}$$

$$\frac{\theta, e \Downarrow^{\mathsf{n}} \texttt{false}, T \qquad \theta, e_2 \Downarrow^{\mathsf{n}_2} v, T_2}{\theta, \texttt{if}(e, e_1, e_2) \Downarrow^{\mathsf{n}+\mathsf{n}_2} v, \texttt{if}^{\mathsf{f}}(T, T_2)} \qquad \frac{\theta, e \Downarrow^{\mathsf{n}} v, T \qquad \delta(v) = v'}{\theta, \delta(e) \Downarrow^{\mathsf{n}+1} v', \delta(T)}$$

$$\frac{}{\theta, \texttt{nil} \Downarrow^0 \texttt{nil}, \texttt{nil}} \qquad \frac{\theta, e_1 \Downarrow^{\mathsf{n}_1} v_1, T_1 \qquad \theta, e_2 \Downarrow^{\mathsf{n}_2} v_2, T_2}{\theta, \texttt{cons}(e_1, e_2) \Downarrow^{\mathsf{n}_1+\mathsf{n}_2} \texttt{cons}(v_1, v_2), \texttt{cons}(T_1, T_2)}$$

$$\frac{\theta, e_1 \Downarrow^{\mathsf{n}_1} v_1, T_1 \qquad \theta[x \mapsto (v_1, T_1)], e_2 \Downarrow^{\mathsf{n}_2} v, T_2}{\theta, \texttt{let}\, x = e_1\, \texttt{in}\, e_2 \Downarrow^{\mathsf{n}_1+\mathsf{n}_2} v, \texttt{let}(x, T_1, T_2)}$$

$$\boxed{\frac{}{\theta, \Lambda.e \Downarrow^0 (\Lambda.e, \theta), (\Lambda.e, \theta)}} \qquad \boxed{\frac{\theta, e \Downarrow^{\mathsf{n}_1} (\Lambda.e', \theta'), T_1 \qquad \theta, e' \Downarrow^{\mathsf{n}_2} v, T_2}{\theta, e[] \Downarrow^{\mathsf{n}_1+\mathsf{n}_2} v, \texttt{IApp}(T_1, T_2)}}$$

Figure 2: Big-step semantics with provenance

Adaptivity of a trace We define the *adaptivity* of a trace T, $\mathtt{adap}(T)$, which means the maximum number of nested δs in T, taking variable and control dependencies into account. To define this, we need an auxiliary notion called 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), again taking variable and control dependencies into account. Technically, $\mathtt{adap} : \text{Traces} \to \mathbb{N}$ and $\mathtt{depth}_x : \text{Traces} \to \mathbb{N}_\perp$. If x does not appear free in T, $\mathtt{depth}_x(T)$ is $\perp$.

The functions $\mathtt{adap}$ and $\mathtt{depth}_x$ are defined by mutual induction in Figure 6.

Explanation of $\mathtt{adap}$ We explain the interesting cases of the definition of $\mathtt{adap}$. The case $T_1\ T_2 \triangleright \mathtt{fix}\, f(x).T_3$ corresponds to a function application with T_1, T_2, T_3 being the traces of the head, the argument and the body, respectively, and x being the argument. The adaptivity is defined to be $\mathtt{adap}(T_1) + \max(\mathtt{adap}(T_3), \mathtt{adap}(T_2) + \mathtt{depth}_x(T_3))$. The term $\mathtt{adap}(T_1)$ occurs additively since the entire computation is control-dependent on the function the head of the application evaluates to. The rest of the term $\max(\mathtt{adap}(T_3), \mathtt{adap}(T_2) + \mathtt{depth}_x(T_3))$ is simply the maximum nesting depth in the body, taking the data dependency on the argument into account. To see this, consider the following exhaustive cases:

- When x appears free in the trace T_3, $\mathtt{depth}_x(T_3)$ is the maximum δ-nesting depth of x in the body. Hence, $\max(\mathtt{adap}(T_3), \mathtt{adap}(T_2) + \mathtt{depth}_x(T_3))$ represents the maximum number of nested δs in the evaluation of $e[e'/x]$ where e' is the argument expression that generates the trace T_2 and e is the body of the function.

- When x does not appear free in the trace T_3 of the body (i.e., the body's evaluation does not depend on x), $\mathtt{depth}_x(T_3) = \perp$, so $\max(\mathtt{adap}(T_3), \mathtt{adap}(T_2) + \mathtt{depth}_x(T_3)) = \max(\mathtt{adap}(T_3), \mathtt{adap}(T_2) + \perp) = \max(\mathtt{adap}(T_3), \perp) = \mathtt{adap}(T_3)$, which is the adaptivity of the body in the absence of dependency from x.

The case $\mathtt{if}^{\mathtt{t}}(T_b, T_t)$ corresponds to the evaluation of $\mathtt{if}(e_b, e_t, \_)$ where e_b evaluates to $\mathtt{true}$ With trace T_b and T_t is the trace of e_t. In this case, since the entire evaluation of e_t is control dependent on e_b, the adaptivity is simply $\mathtt{adap}(T_b) + \mathtt{adap}(T_t)$.

Explanation of $\mathtt{depth}_x$ We explain interesting cases in the definition of $\mathtt{depth}_x$. For the trace $T_1\ T_2 \triangleright \mathtt{fix}\, f(y).T_3$, $\mathtt{depth}_x$ is defined as $\max(\mathtt{depth}_x(T_1), \mathtt{adap}(T_1) + \max(\mathtt{depth}_x(T_3), \mathtt{depth}_x(T_2) + \mathtt{depth}_y(T_3)))$. Here, $\max(\mathtt{depth}_x(T_3), \mathtt{depth}_x(T_2) + \mathtt{depth}_y(T_3))$ is the maximum depth of x in the body (T_3), taking the dependency on the argument into account. Specifically, when the argument variable y is not used in the body, $\mathtt{depth}_y(T_3) = \perp$, and this term is $\mathtt{depth}_x(T_3)$.

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

$$
\begin{array}{lcl}
\mathtt{adap}((x, \theta)) & = & \mathtt{adap}(\theta_2(x)) \\
\mathtt{adap}(T_1\ T_2 \triangleright \mathtt{fix}\, f(x).T_3) & = & \mathtt{adap}(T_1) + \max(\mathtt{adap}(T_3), \mathtt{adap}(T_2) + \mathtt{depth}_x(T_3)) \\
\mathtt{adap}((\mathtt{fix}\, f(x:\tau).e, \theta)) & = & 0 \\
\mathtt{adap}((T_1, T_2)) & = & \max(\mathtt{adap}(T_1), \mathtt{adap}(T_2)) \\
\mathtt{adap}(\mathtt{fst}(T)) & = & \mathtt{adap}(T) \\
\mathtt{adap}(\mathtt{snd}(T)) & = & \mathtt{adap}(T) \\
\mathtt{adap}(\mathtt{true}) & = & 0 \\
\mathtt{adap}(\mathtt{false}) & = & 0 \\
\mathtt{adap}(\mathtt{if}^{\mathtt{t}}(T_b, T_t)) & = & \mathtt{adap}(T_b) + \mathtt{adap}(T_t) \\
\mathtt{adap}(\mathtt{if}^{\mathtt{f}}(T_b, T_f)) & = & \mathtt{adap}(T_b) + \mathtt{adap}(T_f) \\
\mathtt{adap}(c) & = & 0 \\
\mathtt{adap}(\delta(T)) & = & 1 + \mathtt{adap}(T) \\
\mathtt{adap}(\mathtt{nil}) & = & 0 \\
\mathtt{adap}(\mathtt{cons}(T_1, T_2)) & = & \max(\mathtt{adap}(T_1), \mathtt{adap}(T_2)) \\
\mathtt{adap}(\mathtt{let}(x, T_1, T_2)) & = & \max(\mathtt{adap}(T_2), \mathtt{adap}(T_1) + \mathtt{depth}_x(T_2)) \\
\mathtt{adap}(\mathtt{IApp}(T_1, T_2)) & = & \mathtt{adap}(T_1) + \mathtt{adap}(T_2) \\
\mathtt{adap}((\Lambda.e, \theta)) & = & 0 \\
\mathtt{adap}(\mathtt{bernoulli}(T)) & = & \mathtt{adap}(T) \\
\mathtt{adap}(\mathtt{uniform}(T_1, T_2)) & = & \max(\mathtt{adap}(T_1), \mathtt{adap}(T_2)) \\
\mathtt{adap}(\mathtt{dict}({T_i}^{i \in 1 \ldots n})) & = & \max(\mathtt{adap}(T_i)^{i \in 1 \ldots n})
\end{array}
$$

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

$$
\begin{array}{lcl}
\mathtt{depth}_x((y, \theta)) & = & \begin{cases} 0 & \text{if } x = y \\ \perp & \text{if } x \neq y \end{cases} \\
\mathtt{depth}_x(T_1\ T_2 \triangleright \mathtt{fix}\, f(y).T_3) & = & \max(\mathtt{depth}_x(T_1), \\
 & & \mathtt{adap}(T_1) + \max(\mathtt{depth}_x(T_3), \mathtt{depth}_x(T_2) + \mathtt{depth}_y(T_3))) \\
\mathtt{depth}_x((\mathtt{fix}\, f(y:\tau).e, \theta)) & = & \perp \\
\mathtt{depth}_x((T_1, T_2)) & = & \max(\mathtt{depth}_x(T_1), \mathtt{depth}_x(T_2)) \\
\mathtt{depth}_x(\mathtt{fst}(T)) & = & \mathtt{depth}_x(T) \\
\mathtt{depth}_x(\mathtt{snd}(T)) & = & \mathtt{depth}_x(T) \\
\mathtt{depth}_x(\mathtt{true}) & = & \perp \\
\mathtt{depth}_x(\mathtt{false}) & = & \perp \\
\mathtt{depth}_x(\mathtt{if}^{\mathtt{t}}(T_b, T_t)) & = & \max(\mathtt{depth}_x(T_b), \mathtt{adap}(T_b) + \mathtt{depth}_x(T_t)) \\
\mathtt{depth}_x(\mathtt{if}^{\mathtt{t}}(T_b, T_f)) & = & \max(\mathtt{depth}_x(T_b), \mathtt{adap}(T_b) + \mathtt{depth}_x(T_f)) \\
\mathtt{depth}_x(c) & = & \perp \\
\mathtt{depth}_x(\delta(T)) & = & 1 + \mathtt{depth}_x(T) \\
\mathtt{depth}_x(\mathtt{nil}) & = & \perp \\
\mathtt{depth}_x(\mathtt{cons}(T_1, T_2)) & = & \max(\mathtt{depth}_x(T_1), \mathtt{depth}_x(T_2)) \\
\mathtt{depth}_x(\mathtt{let}(y, T_1, T_2)) & = & \max(\mathtt{depth}_x(T_2), \mathtt{depth}_x(T_1) + \mathtt{depth}_y(T_2)) \\
\mathtt{depth}_x(\mathtt{IApp}(T_1, T_2)) & = & \max(\mathtt{depth}_x(T_1), \mathtt{adap}(T_1) + \mathtt{depth}_x(T_2)) \\
\mathtt{depth}_x((\Lambda.e, \theta)) & = & \perp \\
\mathtt{depth}_x(\mathtt{uniform}(T_1, T_2)) & = & \max(\mathtt{depth}_x(T_1), \mathtt{depth}_x(T_2)) \\
\mathtt{depth}_x(\mathtt{bernoulli}(T)) & = & \mathtt{depth}_x(T) \\
\mathtt{depth}_x(\mathtt{dict}({T_i}^{i \in 1 \ldots n})) & = & \max\,(\mathtt{depth}_x(T_i))
\end{array}
$$

Figure 3: Adaptivity of a trace and depth of variable x in a trace

The term $\texttt{adap}(T_1)$ is added since the body's entire execution is control-flow dependent on the function that the head of the application evaluates to.

For the trace $\texttt{if}^{\texttt{t}}(T_b, T_t)$, $\texttt{depth}_x$ is defined as $\max(\texttt{depth}_x(T_b), \texttt{adap}(T_b) + \texttt{depth}_x(T_t))$. The term $\texttt{depth}_x(T_b)$ is simply the maximum depth of x in T_b. We take the max of this with $\texttt{adap}(T_b) + \texttt{depth}_x(T_t)$, the maximum depth of x in T_t, taking the control dependency on T_b into account. Note that when x is not used in T_t, then $\texttt{depth}_x(T_t) = \bot$ and $\texttt{depth}_x(\texttt{if}^{\texttt{t}}(T_b, T_t)) = \texttt{depth}_x(T_b)$.

Lemma 1. For all T and x, $\texttt{depth}_x(T) \leq \texttt{adap}(T)$ in $\mathbb{N}_\bot$.

Proof. By easy induction on T, following the definitions of $\texttt{depth}_x$ and $\texttt{adap}$. □

Remark At first glance it may seem that Lemma 1 can be used to simplify the definition of $\texttt{depth}_x(\texttt{if}^{\texttt{t}}(T_b, T_t))$ from $\max(\texttt{depth}_x(T_b), \texttt{adap}(T_b) + \texttt{depth}_x(T_t))$ to $\texttt{adap}(T_b) + \texttt{depth}_x(T_t)$ since $\texttt{depth}_x(T_b) \leq \texttt{adap}(T_b)$. However, this simplification is not correct, since $\texttt{depth}_x(T_t)$ may be $\bot$. In that case, $\max(\texttt{depth}_x(T_b), \texttt{adap}(T_b) + \texttt{depth}_x(T_t))$ equals $\texttt{depth}_x(T_b)$ while $\texttt{adap}(T_b) + \texttt{depth}_x(T_t)$ equals $\bot$.

More generally, since $\bot$ behaves like $-\infty$, we do not have the implication $a \leq b \Rightarrow a \leq b + c$ as c may be $-\infty$ ($\bot$). As a result, $a \leq b$ does not imply $\max(a, b + c) = b + c$.

13

$$\frac{}{\Gamma, x :!_Z A?_r \vdash_Z x :!_Z A?_r}\ \textbf{Ax} \qquad \frac{}{\Gamma \vdash_Z c :!_Z \mathtt{b}?_0}\ \textbf{b}$$

$$\frac{}{\Gamma \vdash_Z \mathtt{true} :!_Z \mathtt{bool}?_0}\ \textbf{true} \qquad \frac{\Gamma, x : \tau_1 \vdash_k e : \tau_2}{r + \Gamma \vdash_{k+r} \lambda x.e :!_r (\tau_1 \multimap \tau_2)?_0}\ \textbf{lambda}$$

$$\frac{\Gamma \vdash_Z (\lambda x.e) :!_r (\tau_1 \multimap \tau_2)?_0 \qquad \theta \vDash \Gamma}{\cdot \vdash_Z (\lambda x.e, \theta) :!_r (\tau_1 \multimap \tau_2)?_0}\ \textbf{closure}$$

$$\frac{\Gamma_1 \vdash_{Z_1} e_1 :!_0 (\tau_1 \multimap \tau_2)?_r \qquad \Gamma_2 \vdash_{Z_2} e_2 : \tau_1}{\max(\Gamma_1, \Gamma_2) \vdash_{\max(Z_1, r+Z_2)} e_1\ e_2 : \tau_2}\ \textbf{app}$$

$$\frac{\Gamma \vdash_Z e :!_k A?_r}{\Gamma', 1 + \Gamma \vdash_{1+Z} \delta(e) :!_k A?_{r+1}}\ \textbf{delta}$$

$$\frac{\Gamma_1 \vdash_Z e : \tau_1 \qquad \Gamma_2, x : \tau_1 \vdash_{Z'} e' : \tau}{\max(\Gamma_1, \Gamma_2) \vdash_{\max(Z, Z')} \mathtt{let}\, x = e\, \mathtt{in}\, e' : \tau}\ \textbf{let}$$

$$\frac{\Gamma_1 \vdash_{Z_1} e_1 :!_k \mathtt{bool}?_r \qquad \Gamma_2 \vdash_{Z_2} e_2 : \tau \qquad \Gamma_2 \vdash_{Z_2} e_3 : \tau}{\boxed{\max(\Gamma_1, Z_1 + \Gamma_2)} \vdash_{Z_1+Z_2} \mathtt{if}\ e_1\ e_2\ e_3 : \tau}\ \textbf{if}$$

Figure 4: Typing rules, part 1

3 Type system

Linear type	A	$::=$	$\tau \multimap \tau \mid \mathtt{b} \mid \mathtt{bool}$
Type	$Nonlinear\tau$	$::=$	$!_I A$

Theorem 2 (Context Raising)**.** If $\Gamma \vdash_Z e : \tau$ and $r \in \mathbb{N}$, then $r + \Gamma \vdash_{r+Z} e : r + \tau$.

Theorem 3 (Context Weaking)**.** If $\Gamma \vdash_Z e : \tau$, $\Gamma, x : \tau' \vdash_Z e : \tau$.

Theorem 4 (Context Strengthening)**.** If $\Gamma, x : \tau' \vdash_Z e : \tau$ and $x \notin \mathsf{FV}(e)$, then $\Gamma \vdash_Z e : \tau$.

Theorem 5 (Context Max)**.** If $\Gamma_1 \vdash_{Z_1} e : \tau$ and $\Gamma_2 \vdash_{Z_2} e : \tau$, then $\max(\Gamma_1, \Gamma_2) \vdash_{\max(Z_1, Z_2)} e : \tau$.

Theorem 6 (Adaptivity Monotonicity)**.** If $\Gamma \vdash_Z e : \tau$ and $Z' > Z$, then $\Gamma \vdash_{Z'} e : \tau$.

Theorem 7 (Substitution)**.** 1. If $\Gamma, x : \tau' \vdash_Z e : \tau$ and $\vdash_{Z'} v : \tau'$ and $x \in \mathsf{FV}(e)$, then $\Gamma \vdash_{\max(Z, Z')} e[v/x] : \tau$.

2. If $\Gamma, x : \tau' \vdash_Z e : \tau$ and $\vdash_{Z'} v : \tau'$ and $x \notin \mathsf{FV}(e)$, then $\Gamma \vdash_Z e[v/x] : \tau$.

Proof. By simultaneous induction on the typing derivation.
Proof of Statement (2).
Since we know $x \notin \mathsf{FV}(e)$ so that $e[v/x] = e$.
From the assumption $\Gamma, x : \tau' \vdash_Z e : \tau$, by context strengthening, we know that $\Gamma \vdash_Z e : \tau$.

Proof of Statement (1).

$$\frac{}{\Gamma, x :!_Z A?_r \vdash_Z x :!_Z A?_r}\ \textbf{Ax}$$

Assume $\vdash_{Z'} v :!_Z A?_r$ $(\star)$.
TS: $\Gamma \vdash_{\max(Z,Z')} x[v/x] :!_Z A?_r$.
It is proved from the assumption $(\star)$ by weaking from Z' to $\max(Z, Z')$.

$$\frac{\Gamma, y : \tau', x : \tau_1 \vdash_k e : \tau_2 (\diamond)}{r + \Gamma, y : (r + \tau') \vdash_{k+r} \lambda x.e :!_r(\tau_1 \multimap \tau_2)?_0}\ \textbf{lambda}$$

Assume $\vdash_{Z'} v : \tau'$ $(\star)$.
By Theorem 2 context raisig, we know: $\vdash_{Z'+r} v : r + \tau'$ $(\star\star)$
TS: $r + \Gamma \vdash_{\max(k+r,Z'+r)} \lambda x.e[v/y] :!_r(\tau_1 \multimap \tau_2)?_0$.
We know that $y \in \mathsf{FV}(e)$ from the assumption $y \in \mathsf{FV}(\lambda x.e)$.
By IH on $\diamond$ along with $(\star)$, we get: $\Gamma, x : \tau_1 \vdash_{\max(k,Z')} e[v/y] : \tau_2$ (1).
Using the lambda rule, we conclude:
$r + \Gamma \vdash_{\max(k+r,Z'+r)} \lambda x.e[v/y] :!_r(\tau_1 \multimap \tau_2)?_0$.

$$\frac{\Gamma_1, x : \tau' \vdash_{Z_1} e_1 :!_0(\tau_1 \multimap \tau_2)?_r\ (\diamond) \qquad \Gamma_2, x : \tau' \vdash_{Z_2} e_2 : \tau_1\ (\clubsuit)}{\max(\Gamma_1, \Gamma_2), x : \tau' \vdash_{\max(Z_1,r+Z_2)} e_1\ e_2 : \tau_2}\ \textbf{app}$$

Assume $\vdash_{Z'} v : \tau'$ (1).
So we also know: $\vdash_{Z'+r} v : \tau'$ (2).
TS: $\max(\Gamma_1, \Gamma_2) \vdash_{\max(\max(Z_1,r+Z_2),r+Z')} (e_1\ e_2)[v/x] : \tau_2$.
There are three situations:

1. $x \in \mathsf{FV}(e_1), x \in \mathsf{FV}(e_2)$,
   By IH1 on $(\diamond)$ with (1), we get: $\Gamma_1 \vdash_{\max(Z_1,Z')} e_1[v/x] :!_0(\tau_1 \multimap \tau_2)?_r$.
   By IH1 on $(\clubsuit)$ and (1), we get: $\Gamma_2 \vdash_{\max(Z_2,Z')} e_2[v/x] : \tau_1$.

2. $x \notin \mathsf{FV}(e_1), x \in \mathsf{FV}(e_2)$
   By IH2 on $(\diamond)$ with (1), we get: $\Gamma_1 \vdash_{Z_1} e_1[v/x] :!_0(\tau_1 \multimap \tau_2)?_r$, by Lemma Adaptivity Monotonicity, we know: $\Gamma_1 \vdash_{\max(Z_1,Z')} e_1[v/x] : !_0(\tau_1 \multimap \tau_2)?_r$.
   By IH1 on $(\clubsuit)$ and (2), we get: $\Gamma_2 \vdash_{\max(Z_2,Z')} e_2[v/x] : \tau_1$.

3. $x \in \mathsf{FV}(e_1), x \notin \mathsf{FV}(e_2)$
   By IH1 on ($\diamond$) with (1), we get: $\Gamma_1 \vdash_{\max(Z_1,Z')} e_1[v/x] :!_0(\tau_1 \multimap \tau_2)?_r$.
   By IH2 on ($\clubsuit$) and (2), by Lemma Adaptivity Monotonicty, we get: $\Gamma_2 \vdash_{\max(Z_2,Z')} e_2[v/x] : \tau_1$.

By using the rule app, we conclude:
$\max(\Gamma_1, \Gamma_2) \vdash_{\max(\max(Z_1,Z'),r+\max(Z_2,Z'))} (e_1\ e_2)[v/x] : \tau_2$.

$$\frac{\Gamma, x : \tau' \vdash_Z e :!_k A}{\Gamma', 1+\Gamma, x : 1+\tau' \vdash_{1+Z} \delta(e) :!_k A}\ \textbf{delta}$$

Assume $\vdash_{Z'} v : \tau'$ (1).
By Theorem 2 context raisig, we know: $\vdash_{Z'+1} v : 1+\tau'$ (2)
TS: $\Gamma', 1+\Gamma \vdash_{\max(1+Z,Z'+1)} \delta(e) :!_k A$.
We know :$x \in \mathsf{FV}(e)$ from the assumption $x \in \mathsf{FV}(\delta(e))$.
By IH2 on the premise, we know: $\Gamma \vdash_{\max(Z,Z')} e[v/x] :!_k A$.
By the rule delta, we get : $\Gamma', 1+\Gamma \vdash_{1+\max(Z,Z')} \delta(e[v/x]) :!_k A$.

$$\frac{\Gamma_1, y : \tau' \vdash_Z e : \tau_1 \qquad \Gamma_2, y : \tau', x : \tau_1 \vdash_{Z'} e' : \tau}{\max(\Gamma_1, \Gamma_2), y : \tau' \vdash_{\max(Z,Z')} \mathtt{let}\, x = e \,\mathtt{in}\, e' : \tau}\ \textbf{let}$$

Assume $\vdash_{Z_1'} v : \tau'$ (1).
TS: $\max(\Gamma_1, \Gamma_2) \vdash_{\max(Z_1',\max(Z',Z))} (\mathtt{let}\, x = e \,\mathtt{in}\, e')[v/x] : \tau_2$.
Similar to the application rule, we have 3 situations.
By IH on the first premise, we know :$\Gamma_1 \vdash_{\max(Z_1',Z)} e[v/y] : \tau_1$.
By IH on the second premise, we know : $\Gamma_2, x : \tau_1 \vdash_{\max(Z_1',Z')}$.
By the rule let, we conclude that:
$\max(\Gamma_1, \Gamma_2) \vdash_{\max(Z_1',\max(Z,Z'))} (\mathtt{let}\, x = e \,\mathtt{in}\, e')[v/y] : \tau_2$.

□

Theorem 8 (Adaptivity Monotonicity). If $\vdash_Z v :!_k A?_r$ and $r' \geq r$, then $\vdash_Z v :!_k A?_r$.

Theorem 9 (Adaptivity Soundness theorem). If $\Gamma \vdash_Z e : \tau$ and exists θ satisfies Γ and $\theta, e \Downarrow^{\mathsf{n}} v, T$, then $\vdash_{Z-A(T)} v : \tau[?0]$ and $Z - A(T) \geq 0$.
(θ satisfies Γ means $\mathtt{dom}(\theta) = \mathtt{dom}(\Gamma) \wedge \forall x_i \in \mathtt{dom}(\Gamma)$. so that $\vdash_{r_i} \theta_1(x_i) : \Gamma(x)[?0] \wedge r_i \geq \mathtt{adap}(\theta_2(x_i))$, and $\tau[?0]$ means that $\tau =!_k A?_r \wedge \tau[?0] =!_k A?_0$)

Proof. By indution on the typing derivation.

$$\frac{}{\Gamma, x :!_Z A \vdash_Z x :!_Z A?_r}\ \textbf{Ax}$$

Assume the enviroment $\theta \vDash \Gamma$, and $\vdash_Z v :!_Z A?_0$. We know that $\theta_2(x) = T$,

so that $\dfrac{}{\theta, x \Downarrow^v, (x,\theta)}$.
So we conclude that: $Z - A((x,\theta)) = Z-$ and $\vdash_{Z-A((x,\theta))} v :!_Z A?_0$.

$$\frac{\Gamma, x:\tau_1 \vdash_k e:\tau_2}{r+\Gamma \vdash_{k+r} \lambda x.e :!_r(\tau_1 \multimap \tau_2)?_0}\ \textbf{lambda}$$

Assume exists the enviroment θ so that
$\dfrac{}{\theta, \lambda x.e \Downarrow^(\lambda x.e, \theta), (\lambda x.e, \theta)}$.
So by the rule closure we conclude that:
$Z - A((\lambda x.e,\theta)) = Z$ and $\vdash_{Z-A((\lambda x.e,\theta))} (\lambda x.e,\theta) :!_r(\tau_1 \multimap \tau_2)?_0$.

$$\frac{\Gamma_1 \vdash_{Z_1} e_1 :!_0(\tau_1 \multimap \tau_2)?_r\ (\star) \qquad \Gamma_2 \vdash_{Z_2} e_2 : \tau_1\ (\diamond)}{\max(\Gamma_1,\Gamma_2) \vdash_{\max(Z_1, r+Z_2)} e_1\ e_2 : \tau_2}\ \textbf{app}$$

For all the variables x_i in $\mathtt{dom}(\max(\Gamma_1,\Gamma_2))$ and $x_i \notin \mathtt{dom}(\Gamma_1)$. We extend Γ_1 to $\Gamma_1, x_i : \tau_i \wedge \tau_i = \Gamma_2(x_i)$. With the Theorem 11, from $(\star)$, we know : $\Gamma_1, x_i : \tau_i \vdash_{Z_1} e_1 :!_0(\tau_1 \multimap \tau_2)?_r$ $(\star\star)$.

For all the variables x'_i in $\mathtt{dom}(\max(\Gamma_1,\Gamma_2))$ and $x_i \notin \mathtt{dom}(\Gamma_2)$. We extend Γ_2 to $\Gamma_2, x'_i : \tau'_i \wedge \tau' = \Gamma_1(x'_i)$. With the Theorem 11, from $(\diamond)$, we know : $\Gamma_2, x'_i : \tau'_i \vdash_{Z_2} e_2 : \tau_1$ $(\diamond\diamond)$.

Assume θ satisfies $\min(\Gamma_1,\Gamma_2)$.
$\min(\Gamma_1,\Gamma_2) = \{\Gamma | \mathtt{dom}(\Gamma) = \mathtt{dom}(\Gamma_1) \cup \mathtt{dom}(\Gamma_2) \wedge \Gamma(x) = \min(\Gamma_1(x), \Gamma_2(x))\}$.

From Theorem 10, we conclude that θ satisfies $\Gamma_1, x_i : \tau_i$, as well as θ satisfies $\Gamma_2, x'_i : \tau'_i$, also θ satisfies $\max(\Gamma_1,\Gamma_2)$.

By IH on $(\star\star)$, assume $\theta, e_1 \Downarrow^v{}_1, T_1$, we know: $\vdash_{Z_1 - A(T_1)} v_1 :!_0(\tau_1 \multimap \tau_2)?_0 \wedge A(T_1) \le r$ (e).
Because v_1 is a function, we assume: $v_1 = (\lambda x.e, \theta')$. So we have: $\vdash_{Z_1 - A(T_1)} (\lambda x.e, \theta') :!_0(\tau_1 \multimap \tau_2)?_0$ $(\clubsuit)$, from the closure rule, we know : θ' satisfies $\Gamma'_1 \wedge \Gamma'_1 \vdash_{Z_1 - A(T_1)} \lambda x.e :!_0(\tau_1 \multimap \tau_2)?_0$ $(\clubsuit\clubsuit)$.

By IH on $(\diamond\diamond)$, assume $\theta, e_2 \Downarrow^v{}_2, T_2$, we know $\vdash_{Z_2 - A(T_2)} v_2 : \tau_1[?0]$ (a).

From $(\clubsuit\clubsuit)$ and the rule lambda:

$$\frac{\Gamma'_1, x:\tau_1 \vdash_{Z_1 - A(T_1)} e : \tau_2 (\heartsuit)}{\Gamma'_1 \vdash_{Z_1 - A(T_1) + 0} \lambda x.e :!_0(\tau_1 \multimap \tau_2)}\ \textbf{lambda}$$

We know that : $\theta'[x \to v_2]$satisfies $\Gamma'_1, x : \tau_1$ and assume $\theta'[x \mapsto v_2], e \Downarrow^v$ $, T$.
By IH on $(\heartsuit)$, we know: $\vdash_{Z_1 - A(T_1) - A(T)} v : \tau_2[?0]$ (d) and $Z_1 - A(T_1) - A(T) \ge$

$0 \implies Z_1 \geq A(T_1) + A(T)$ (b).

From the theorem Depth Bound on $\heartsuit$, we assume that $\tau_1 = !_{k_1} A_1 ?_{r_1}$. So we know that $k_1 \geq \texttt{depth}_x(T)$.

By the Theorem 13 on (a), we know that: $Z_2 - A(T_2) \geq k_1 \implies Z_2 - A(T_2) \geq \texttt{depth}_x(T)$.
So we conclude that $Z_2 \geq A(T_2) + \texttt{depth}_x(T)$ (c).

From the application evaluation rule, we have:

$$\frac{\begin{array}{cc} \theta, e_1 \Downarrow^v{}_1, T_1 & v_1 = (\lambda x.e, \theta') \\ \theta, e_2 \Downarrow^v{}_2, T_2 & \theta'[x \mapsto v_2], e \Downarrow^v, T \end{array}}{\theta, e_1\ e_2 \Downarrow^v, T_1\ T_2 \triangleright \texttt{fix}\, f(x).T}$$

TS1: $\vdash_{\max(Z_1, r+Z_2) - A(T_1\ T_2 \triangleright \texttt{fix}\, f(x).T)} v : \tau_2[?0]$.
It is proved from (d) and Theorem Aaptivity Monotonicity.
TS2: $\max(Z_1, r + Z_2) \geq A(T_1\ T_2 \triangleright \texttt{fix}\, f(x).T$.
It is proved by (b),(c), (e).

□

Theorem 10. $\vdash_Z v : !_k A ?_r$ and $k \leq k'$ and $r \leq r'$, then exists Z' so that $\vdash_{Z'} v : !_{k'} A ?_{r'}$.

Theorem 11. $\Gamma \vdash_Z e : \tau$ and $x \notin \mathsf{FV}(e)$, then $\forall \tau'. \Gamma, x : \tau' \vdash_Z e : \tau$.

Theorem 12. If $\Gamma \vdash_Z e : !_k A ?_r$ and $\theta| = \Gamma$ and $\theta, e \Downarrow^v, T$, then $\texttt{adap}(T) \leq r$.

Theorem 13. If $\Gamma \vdash_Z e : !_k A ?_r$, then $Z \geq k$.

Theorem 14 (Depth Bound)**.** If $\Gamma, x : !_k A ?_r \vdash_Z e : \tau$ and $\theta| = \Gamma, x : !_k A ?_r$ and $\theta, e \Downarrow^v, T$, then $\texttt{depth}_x(T) \leq k$.

4 vectors

$$
\begin{array}{lcll}
\max(\bot, q) & = & q & \\
\max(q, \bot) & = & q & \\
\max(\infty, q) & = & \infty & \\
\max(q, \infty) & = & \infty & \\
 & & & \\
\bot + q & = & \bot & \\
q + \bot & = & \bot & \\
\infty + q & = & \infty & \text{if } q \neq \bot \\
q + \infty & = & \infty & \text{if } q \neq \bot \\
 & & & \\
\bot \leq q & & & \\
q \leq \infty & & &
\end{array}
$$

$$
\begin{array}{llll}
\text{Expr.} & e & ::= & x \mid e_1\ e_2 \mid \mathtt{fix}\, f(x).e \mid (e_1, e_2) \mid \mathtt{fst}(e) \mid \mathtt{snd}(e) \mid \\
 & & & \mathtt{true} \mid \mathtt{false} \mid \mathtt{if}(e_1, e_2, e_3) \mid c \mid \delta(e) \mid \Lambda.e \mid e\,[] \\
 & & & \mid \mathtt{let}\, x : q = e_1\, \mathtt{in}\, e_2 \mid \mathtt{nil} \mid \mathtt{cons}(e_1, e_2) \\
 & & & \mid \quad \mathtt{bernoulli}\ e \mid \mathtt{uniform}\ e_1\ e_2 \\
 & & & \mid \mathtt{dict}(\mathsf{attr}_i \to {e'_i}^{i \in 1 \ldots n}) \\
\text{Value} & v & ::= & \mathtt{true} \mid \mathtt{false} \mid c \mid (\mathtt{fix}\, f(x : \tau).e, \theta) \mid (v_1, v_2) \mid \mathtt{nil} \mid \mathtt{cons}(v_1, v_2) \mid \\
 & & & (\Lambda.e, \theta) \mid \mathtt{dict}(\mathsf{attr}_i \to {v'_i}^{i \in 1 \ldots n}) \\
\text{Environment} & \theta & ::= & x_1 \mapsto v_1, \ldots, x_n \mapsto v_n
\end{array}
$$

$$
\begin{array}{lll}
\text{Trace} \quad T & ::= & (x, \theta) \mid T_1\ T_2 \triangleright \mathtt{fix}\, f(x).T_3 \mid (\mathtt{fix}\, f(x : \tau).e, \theta) \mid (T_1, T_2) \mid \mathtt{fst}(T) \mid \\
 & & \mathtt{snd}(T) \mid \mathtt{true} \mid \mathtt{false} \mid \mathtt{if}^{\mathtt{t}}(T_b, T_t) \mid \mathtt{if}^{\mathtt{f}}(T_b, T_f) \mid c \mid \delta(T) \\
 & & \mathtt{nil} \mid \mathtt{cons}(T_1, T_2) \mid \mathtt{IApp}(T_1, T_2) \mid (\Lambda.e, \theta) \\
 & & \mid \mathtt{dict}({T_i}^{i \in 1 \ldots n})
\end{array}
$$

$$\frac{}{\theta, x \Downarrow^{\theta} (x), (x, \theta)} \qquad \frac{}{\theta, c \Downarrow^{c}, c} \qquad \frac{}{\theta, \texttt{true} \Downarrow^{\texttt{true}}, \texttt{true}}$$

$$\frac{}{\theta, \texttt{false} \Downarrow^{\texttt{false}}, \texttt{false}} \qquad \frac{\theta, e \Downarrow^{c}, T}{\theta, \texttt{bernoulli}\ e \Downarrow^{c}, \texttt{bernoulli}(T)}$$

$$\frac{\theta, e_1 \Downarrow^{c}, T_1 \qquad \theta, e_2 \Downarrow^{c}, T_2}{\theta, \texttt{uniform}\ e_1\ e_2 \Downarrow^{c}, \texttt{uniform}(T_1, T_2)}$$

$$\frac{}{\theta, \texttt{fix}\, f(x : \tau).e \Downarrow^{(}\ \texttt{fix}\, f(: \tau).e, \theta), (\texttt{fix}\, f(x : \tau).e, \theta)}$$

$$\frac{\begin{array}{cc}\theta, e_1 \Downarrow^{v}{}_1, T_1 & v_1 = (\texttt{fix}\, f(x : \tau).e, \theta') \\ \theta, e_2 \Downarrow^{v}{}_2, T_2 & \theta'[f \mapsto v_1, x \mapsto v_2], e \Downarrow^{v}, T\end{array}}{\theta, e_1\ e_2 \Downarrow^{v}, T_1\ T_2 \triangleright \texttt{fix}\, f(x).T} \qquad \frac{\theta, e_1 \Downarrow^{v}{}_1, T_1 \qquad \theta, e_2 \Downarrow^{v}{}_2, T_2}{\theta, (e_1, e_2) \Downarrow^{(}\ v_1, v_2), (T_1, T_2)}$$

$$\frac{\theta, e \Downarrow^{(}\ v_1, v_2), T}{\theta, \texttt{fst}(e) \Downarrow^{v}{}_1, \texttt{fst}(T)} \qquad \frac{\theta, e \Downarrow^{(}\ v_1, v_2), T}{\theta, \texttt{snd}(e) \Downarrow^{v}{}_2, \texttt{snd}(T)}$$

$$\frac{\theta, e \Downarrow^{\texttt{true}}, T \qquad \theta, e_1 \Downarrow^{v}, T_1}{\theta, \texttt{if}(e, e_1, e_2) \Downarrow^{v}, \texttt{if}^{\texttt{t}}(T, T_1)} \qquad \frac{\theta, e \Downarrow^{\texttt{false}}, T \qquad \theta, e_2 \Downarrow^{v}, T_2}{\theta, \texttt{if}(e, e_1, e_2) \Downarrow^{v}, \texttt{if}^{\texttt{f}}(T, T_2)}$$

$$\frac{\theta, e \Downarrow^{v}, T \qquad \delta(v) = v'}{\theta, \delta(e) \Downarrow^{v'}, \delta(T)} \qquad \frac{}{\theta, \texttt{nil} \Downarrow^{\texttt{nil}}, \texttt{nil}}$$

$$\frac{\theta, e_1 \Downarrow^{v}{}_1, T_1 \qquad \theta, e_2 \Downarrow^{v}{}_2, T_2}{\theta, \texttt{cons}(e_1, e_2) \Downarrow^{\texttt{cons}} (v_1, v_2), \texttt{cons}(T_1, T_2)}$$

$$\frac{\theta, e_1 \Downarrow^{v}{}_1, T_1 \qquad \theta[x \mapsto v_1], e_2 \Downarrow^{v}, T_2}{\theta, \texttt{let}\, x; q = e_1\, \texttt{in}\, e_2 \Downarrow^{v}, \texttt{let}(x, T_1, T_2)}$$

$$\frac{}{\theta, \Lambda.e \Downarrow^{(}\ \Lambda.e, \theta), (\Lambda.e, \theta)} \qquad \frac{\theta, e \Downarrow^{(}\ \Lambda.e', \theta'), T_1 \qquad \theta, e' \Downarrow^{v}, T_2}{\theta, e[] \Downarrow^{v}, \texttt{IApp}(T_1, T_2)}$$

$$\frac{\theta, e_i'^{\,i \in 1\ldots n} \Downarrow^{v_i'^{\,i \in 1\ldots n}}, T_i'^{\,i \in 1\ldots n}}{\theta, \mathsf{dict}(\mathsf{attr}_i \to e_i'^{\,i \in 1\ldots n}) \Downarrow^{\mathsf{dict}} (\mathsf{attr}_i \to v_i'^{\,i \in 1\ldots n}), \mathsf{dict}(T_i'^{\,i \in 1\ldots n})}$$

Figure 5: Big-step semantics with provenance, vector

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

$$
\begin{array}{lcl}
\texttt{adap}((x, \theta)) & = & 0 \\
\texttt{adap}(T_1\ T_2 \triangleright \texttt{fix}\, f(x).T_3) & = & \texttt{adap}(T_1) + \max(\texttt{adap}(T_3), \texttt{adap}(T_2) + \texttt{depth}_x(T_3)) \\
\texttt{adap}((\texttt{fix}\, f(x:\tau).e, \theta)) & = & 0 \\
\texttt{adap}((T_1, T_2)) & = & \max(\texttt{adap}(T_1), \texttt{adap}(T_2)) \\
\texttt{adap}(\texttt{fst}(T)) & = & \texttt{adap}(T) \\
\texttt{adap}(\texttt{snd}(T)) & = & \texttt{adap}(T) \\
\texttt{adap}(\texttt{true}) & = & 0 \\
\texttt{adap}(\texttt{false}) & = & 0 \\
\texttt{adap}(\texttt{if}^{\texttt{t}}(T_b, T_t)) & = & \texttt{adap}(T_b) + \texttt{adap}(T_t) \\
\texttt{adap}(\texttt{if}^{\texttt{f}}(T_b, T_f)) & = & \texttt{adap}(T_b) + \texttt{adap}(T_f) \\
\texttt{adap}(c) & = & 0 \\
\texttt{adap}(\delta(T)) & = & 1 + \texttt{adap}(T) \\
\texttt{adap}(\texttt{nil}) & = & 0 \\
\texttt{adap}(\texttt{cons}(T_1, T_2)) & = & \max(\texttt{adap}(T_1), \texttt{adap}(T_2)) \\
\texttt{adap}(\texttt{let}(x, T_1, T_2)) & = & \max(\texttt{adap}(T_2), \texttt{adap}(T_1) + \texttt{depth}_x(T_2)) \\
\texttt{adap}(\texttt{IApp}(T_1, T_2)) & = & \texttt{adap}(T_1) + \texttt{adap}(T_2) \\
\texttt{adap}((\Lambda.e, \theta)) & = & 0 \\
\texttt{adap}(\texttt{bernoulli}(T)) & = & \texttt{adap}(T) \\
\texttt{adap}(\texttt{uniform}(T_1, T_2)) & = & \max(\texttt{adap}(T_1), \texttt{adap}(T_2)) \\
\texttt{adap}(\texttt{dict}(T_i{}^{i \in 1 \ldots n})) & = & \max(\texttt{adap}(T_i)^{i \in 1 \ldots n})
\end{array}
$$

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

$$
\begin{array}{lcl}
\texttt{depth}_x((y, \theta)) & = & \begin{cases} 0 & \text{if } x = y \\ \perp & \text{if } x \neq y \end{cases} \\
\texttt{depth}_x(T_1\ T_2 \triangleright \texttt{fix}\, f(y).T_3) & = & \max(\texttt{depth}_x(T_1), \\
 & & \texttt{adap}(T_1) + \max(\texttt{depth}_x(T_3), \texttt{depth}_x(T_2) + \texttt{depth}_y(T_3))) \\
\texttt{depth}_x((\texttt{fix}\, f(y:\tau).e, \theta)) & = & \perp \\
\texttt{depth}_x((T_1, T_2)) & = & \max(\texttt{depth}_x(T_1), \texttt{depth}_x(T_2)) \\
\texttt{depth}_x(\texttt{fst}(T)) & = & \texttt{depth}_x(T) \\
\texttt{depth}_x(\texttt{snd}(T)) & = & \texttt{depth}_x(T) \\
\texttt{depth}_x(\texttt{true}) & = & \perp \\
\texttt{depth}_x(\texttt{false}) & = & \perp \\
\texttt{depth}_x(\texttt{if}^{\texttt{t}}(T_b, T_t)) & = & \max(\texttt{depth}_x(T_b), \texttt{adap}(T_b) + \texttt{depth}_x(T_t)) \\
\texttt{depth}_x(\texttt{if}^{\texttt{t}}(T_b, T_f)) & = & \max(\texttt{depth}_x(T_b), \texttt{adap}(T_b) + \texttt{depth}_x(T_f)) \\
\texttt{depth}_x(c) & = & \perp \\
\texttt{depth}_x(\delta(T)) & = & 1 + \texttt{depth}_x(T) \\
\texttt{depth}_x(\texttt{nil}) & = & \perp \\
\texttt{depth}_x(\texttt{cons}(T_1, T_2)) & = & \max(\texttt{depth}_x(T_1), \texttt{depth}_x(T_2)) \\
\texttt{depth}_x(\texttt{let}(y, T_1, T_2)) & = & \max(\texttt{depth}_x(T_2), \texttt{depth}_x(T_1) + \texttt{depth}_y(T_2)) \\
\texttt{depth}_x(\texttt{IApp}(T_1, T_2)) & = & \max(\texttt{depth}_x(T_1), \texttt{adap}(T_1) + \texttt{depth}_x(T_2)) \\
\texttt{depth}_x((\Lambda.e, \theta)) & = & \perp \\
\texttt{depth}_x(\texttt{uniform}(T_1, T_2)) & = & \max(\texttt{depth}_x(T_1), \texttt{depth}_x(T_2)) \\
\texttt{depth}_x(\texttt{bernoulli}(T)) & = & \texttt{depth}_x(T) \\
\texttt{depth}_x(\texttt{dict}(T_i{}^{i \in 1 \ldots n})) & = & \max\,(\texttt{depth}_x(T_i))
\end{array}
$$

Figure 6: Adaptivity of a trace and depth of variable x in a trace

Two-rounds:

$$
\begin{array}{l}
\texttt{let}\ g : \bot = \texttt{fix}\ f(j:\texttt{int}).\lambda k:\texttt{int}.\\
\quad \texttt{if}\big((j<k),\\
\qquad \texttt{let}\ a : 0 = \delta\left(\texttt{dict}\big(\mathsf{attr}_i \to (\mathsf{get}\ \mathsf{attr}_i\ j) * (\mathsf{get}\ \mathsf{attr}_i\ k)^{i\in 1\ldots 2^k}\big)\right)\ \texttt{in}\\
\qquad\quad (a,j) :: (f\ (j+1)\ k)\\
\qquad\quad ,\ []\big)\ \texttt{in}\\
\quad \texttt{fix}\ \mathsf{twoRound}(k:\texttt{int}).\\
\qquad \texttt{let}\ l : 1 = g\ 0\ k\ \texttt{in}\\
\qquad \texttt{let}\ q : 1 = \texttt{dict}\Big(\mathsf{attr}_i \to \mathsf{sign}\\
\qquad (\mathsf{foldl}\ (\lambda acc:\texttt{real}.\lambda ai:\texttt{real}\cdot\texttt{int}.\big(acc\ + (\mathsf{get}\ \mathsf{attr}_i\ (\texttt{snd}\ ai))\cdot \log(\frac{1+(\texttt{fst}\ ai)}{1-(\texttt{fst}\ ai)})\big)\ 0.0\ l)^{i\in 1\ldots 2^k}\Big)\ \texttt{in}\\
\qquad \delta(q)
\end{array}
$$

Algorithm 1 A two-round analyst strategy for random data (Algorithm 4 in ...)

Require: Mechanism $\mathcal{M}$ with a hidden state $X \in \{-1,+1\}^{n\times(k+1)}$.
  for $j \in [k]$ **do**.
    define $q_j(x) = x(j)\cdot x(k)$ where $x \in \{-1,+1\}^{k+1}$.
    let $a_j = \mathcal{M}(q_j)$
    {In the line above, $\mathcal{M}$ computes approx. the exp. value of q_j over X. So, $a_j \in [-1,+1]$.}
  define $q_{k+1}(x) = \mathrm{sign}\big(\sum_{i\in[k]} x(i) \times \ln\frac{1+a_i}{1-a_i}\big)$ where $x \in \{-1,+1\}^{k+1}$.
  {In the line above, $\mathrm{sign}(y) = \begin{cases} +1 & \text{if } y \geq 0 \\ -1 & \text{otherwise} \end{cases}$.}
  let $a_{k+1} = \mathcal{M}(q_{k+1})$
  {In the line above, $\mathcal{M}$ computes approx. the exp. value of q_{k+1} over X. So, $a_{k+1} \in [-1,+1]$.}
  return a_{k+1}.
Ensure: $a_{k+1} \in [-1,+1]$

Algorithm 2 A multi-round analyst strategy for random data (Algorithm 5 in ...)

Require: Mechanism $\mathcal{M}$ with a hidden state $X_0 \in [N]^n$ sampled u.a.r., control set size c
  Define control dataset $C = \{0, 1, \cdots, c-1\}$
  Initialize $Nscore(i) = 0$ for $i \in [N]$, $I = \emptyset$ and $Cscore(C(i)) = 0$ for $i \in [c]$
  for $j \in [k]$ **do**
    let $p = \mathtt{uniform}(0, 1)$
    define $q(x) = \mathtt{bernoulli}(p)$.
    define $qc(x) = \mathtt{bernoulli}(p)$.
    define $qj(x) = \mathsf{restrict}(q, X_j)$
    let $a = \mathcal{M}(qj)$
    for $i \in [N]$ **do**
      $Nscore(i) = Nscore(i) + (a - p) * (q(i) - p)$ if $i \notin I$
    for $i \in [c]$ **do**
      $Cscore(C(i)) = Cscore(C(i)) + (a - p) * (qc(i) - p)$
    let $I = \{i | i \in [N] \wedge Nscore(i) > \max(Cscore)\}$
    let $X_j = X_{j-1} \setminus I$
  return X_j.

$\texttt{let}\ \mathsf{updtSC} =$
$\texttt{fix}\ \mathsf{f}(z:\texttt{unit}).\lambda sc:\texttt{list}\,\texttt{real}.\lambda a:\texttt{real}.\lambda p:\texttt{real}.\lambda q.$
$\lambda I:\texttt{list}\,\texttt{int}.\lambda i:\texttt{int}.\lambda n:\texttt{int}.$
$\quad \texttt{if}\Big((i<n),$
$\quad\quad \texttt{if}\big((\mathsf{in}\ i\ I),$
$\quad\quad\quad \texttt{let}\ x:0 = (\mathsf{depth}\ sc\ i) + (a-p)*(q\{i\}-p)\ \texttt{in}$
$\quad\quad\quad \texttt{let}\ sc':0 = \mathsf{updt}\ sc\ i\ x\ \texttt{in}$
$\quad\quad\quad\quad \mathsf{f}\ ()\ sc'\ a\ p\ q\ I\ \ (i+1)\ n$
$\quad\quad ,\mathsf{f}\ ()\ sc\ a\ p\ q\ I\ \ (i+1)\ n\big)$
$\quad ,sc\Big)\ \texttt{in}$

$\texttt{let}\ \mathsf{updtSCC} =$
$\texttt{fix}\ \mathsf{f}(z:\texttt{unit}).\lambda scc:\texttt{list}\,\texttt{real}.\lambda a:\texttt{real}.\lambda p:\texttt{real}.\lambda qc$
$\lambda i:\texttt{int}.\lambda cr:\texttt{int}.$
$\quad \texttt{if}\Big((i<cr),$
$\quad\quad\quad \texttt{let}\ x:0 = (\mathsf{nth}\ scc\ i) + (a-p)*(qc\{i\}-p)\ \texttt{in}$
$\quad\quad\quad \texttt{let}\ scc':0 = \mathsf{updt}\ scc\ i\ x\ \texttt{in}$
$\quad\quad\quad\quad \mathsf{f}\ ()\ scc'\ a\ p\ qc\ (i+1)\ cr$
$\quad ,scc\Big)\ \texttt{in}$

$\texttt{let}\ \mathsf{updtI} =$
$\texttt{fix}\ \mathsf{f}(z:\texttt{unit}).\lambda maxScc:\texttt{real}.\lambda sc:\texttt{list}\,\texttt{real}.\lambda i:\texttt{int}.\lambda n:\texttt{int}.$
$\quad \texttt{if}\Big((i<n),$
$\quad\quad \texttt{if}\big(((\mathsf{nth}\ scc\ i) > maxScc),$
$\quad\quad\quad i :: (\mathsf{f}\ ()\ maxScc\ sc\ (i+1)\ n)$
$\quad\quad\quad \mathsf{f}\ ()\ maxScc\ sc\ (i+1)\ n\big)$
$\quad ,[]\Big)\ \texttt{in}$

$\texttt{fix}\ \mathsf{multiRound}(z:\texttt{unit}).\Lambda k.\Lambda j.\lambda k:\texttt{int}[k].\lambda j:\texttt{int}[j].\lambda sc:\texttt{list}\,\texttt{real}.$
$\lambda scc:\texttt{list}\,\texttt{real}.\lambda il:\texttt{list}\,\texttt{int}.\lambda n:\texttt{int}.\lambda cr:\texttt{int}.\lambda d:\texttt{list}\,\texttt{int}.$
$\quad \texttt{if}\big((j<k),$
$\quad \texttt{let}\ p:k-j = \texttt{uniform}\ 0\ 1\ \texttt{in}$
$\quad\quad \texttt{let}\ q:k-j = \texttt{dict}(\mathsf{attr}_i \to \texttt{bernoulli}\ p^{i\in 1\ldots n})\ \texttt{in}$
$\quad\quad \texttt{let}\ qc:k-j = \texttt{dict}(\mathsf{attr}_i \to \texttt{bernoulli}\ p^{i\in 1\ldots n})\ \texttt{in}$
$\quad\quad \texttt{let}\ qj:k-j = \texttt{dict}(\mathsf{attr}_i \to if((\mathsf{in}\ e_i\ d)\ ,\ q\{e_i\}\ ,\mathsf{else}\ 0)^{i\in 1\ldots n})$
$\quad\quad \texttt{let}\ a:k-j-1 = \delta(q_j)\ \texttt{in}$
$\quad\quad\quad \texttt{let}\ sc':k-j-1 = \mathsf{updtSC}\ ()\ sc\ a\ p\ qj\ il\ \ 0\ n\ \ \texttt{in}$
$\quad\quad\quad \texttt{let}\ scc':k-j-1 = \mathsf{updtSCC}\ ()\ scc\ a\ p\ qc\ \ 0\ cr\ \texttt{in}$
$\quad\quad\quad \texttt{let}\ maxScc:k-j-1 = \mathsf{foldl}\ (\lambda acc:\texttt{real}.\lambda a:\texttt{real}.\texttt{if}(acc<a,a,acc))\ 0\ scc'\ \texttt{in}$
$\quad\quad\quad \texttt{let}\ il':k-j-1 = \mathsf{updtI}\ ()\ maxScc\ sc\ 0\ n\ \texttt{in}$
$\quad\quad\quad \texttt{let}\ d':k-j-1 = d \setminus il'\ \texttt{in}$
$\quad\quad\quad\quad a :: (\mathsf{multiRound}\ ()\ [k]\ [j+1]\ k\ (j+1)\ sc'\ scc'\ il'\ n\ cr\ d')$
$\quad ,[]\big)$

$$
\begin{array}{llcl}
\text{Index Term} & I, Z & ::= & i \mid n \mid I_1 + I_2 \mid I_1 - I_2 \mid \mathtt{max}(I_1, I_2) \\
\text{Sort} & S & ::= & \mathbb{N} \\
\text{Type} & \tau & ::= & \mathtt{b} \mid \mathtt{bool} \mid \tau_1 \times \tau_2 \mid \tau_1; q \xrightarrow{;Z} \tau_2 \mid \mathtt{list}\,\tau \mid \Box\,(\tau) \mid \\
 & & & \mathtt{real} \mid \mathtt{int} \mid \mathtt{int}[I] \mid \forall i \overset{;Z}{::} S.\, \tau \mid \mathsf{VC}(\tau_i \to {\tau'_i}^{\,i \in 1,2,\ldots n})
\end{array}
$$

$$\frac{\Gamma(x) = \tau \qquad \rho(x) \geq 0}{\Delta; \Gamma; \rho \vdash_0 x : \tau} \textbf{ var}$$

$$\frac{\Delta; \Gamma; \rho_1 \vdash_{Z_1} e_1 : \tau_1; \xrightarrow{q;\, Z} \tau_2 \qquad \Delta; \Gamma; \rho_2 \vdash_{Z_2} e_2 : \tau_1 \\ Z' = Z_1 + \max(Z, Z_2 + q) \qquad \rho' = \max(\rho_1, Z_1 + \rho_2)}{\Delta; \Gamma; \rho' \vdash_{Z'} e_1\, e_2 : \tau_2} \textbf{ app}$$

$$\frac{\Delta; \Gamma, f : (\tau_1; \xrightarrow{q;\, Z} \tau_2), x : \tau_1; \rho, [x : q] \vdash_Z e : \tau_2}{\Delta; \Gamma; \rho \vdash_0 \texttt{fix}\, f(x).e : \tau_1; \xrightarrow{q;\, Z} (\tau_2)} \textbf{ fix}$$

$$\frac{\Delta; \Gamma; \rho_1 \vdash_{Z_1} e_1 : \tau_1 \qquad \Delta; \Gamma; \rho_2 \vdash_{Z_2} e_2 : \tau_2 \\ Z' = \max(Z_1, Z_2) \qquad \rho' = \max(\rho_1, \rho_2)}{\Delta; \Gamma; \rho' \vdash_{Z'} (e_1, e_2) : \tau_1 \times \tau_2} \textbf{ pair}$$

$$\frac{\Delta; \Gamma; \rho \vdash_Z e : \tau_1 \times \tau_2}{\Delta; \Gamma; \rho \vdash_Z \texttt{fst}(e) : \tau_1} \textbf{ fst} \qquad \frac{\Delta; \Gamma; \rho \vdash_Z e : \tau_1 \times \tau_2}{\Delta; \Gamma; \rho \vdash_Z \texttt{snd}(e) : \tau_2} \textbf{ snd}$$

$$\frac{}{\Delta; \Gamma; \rho \vdash_Z \texttt{true} : \texttt{bool}} \textbf{ true} \qquad \frac{}{\Delta; \Gamma; \rho \vdash_Z \texttt{false} : \texttt{bool}} \textbf{ false}$$

$$\frac{\Delta; \Gamma; \rho' \vdash_{Z'} e : \tau' \qquad Z' < Z \qquad \rho' < \rho \qquad \Delta \models \tau' <: \tau}{\Delta; \Gamma; \rho \vdash_Z e : \tau} \textbf{ subtype}$$

$$\frac{\Delta; \Gamma; \rho \vdash_Z e : \texttt{real}}{\Delta; \Gamma; \rho \vdash_Z \texttt{bernoulli}\ e : \texttt{real}} \textbf{ bernoulli}$$

$$\frac{\Delta; \Gamma; \rho_1 \vdash_{Z_1} e_1 : \texttt{real} \\ \Delta; \Gamma; \rho_2 \vdash_{Z_2} e_2 : \texttt{real} \qquad Z = \max(Z_1, Z_2) \qquad \rho' = \max(\rho_1, \rho_2)}{\Delta; \Gamma; \rho' \vdash_{Z'} \texttt{uniform}\ e_1\ e_2 : \texttt{real}} \textbf{ uniform}$$

Figure 7: Typing rules, part 1

$$\frac{\Delta;\Gamma;\rho_1 \vdash_{Z_1} e_1 : \mathtt{bool} \qquad \Delta;\Gamma;\rho \vdash_Z e_2 : \tau \qquad \Delta;\Gamma;\rho \vdash_Z e_3 : \tau \qquad Z' = Z_1 + Z \qquad \rho' = \max(\rho_1, Z_1 + \rho)}{\Delta;\Gamma;\rho' \vdash_{Z'} \mathtt{if}(e_1, e_2, e_3) : \tau} \ \textbf{if}$$

$$\frac{}{\Delta;\Gamma;\rho \vdash_Z c : \mathtt{b}} \ \textbf{const} \qquad \frac{}{\Delta;\Gamma;\rho \vdash_Z n : \mathtt{int}[n]} \ \textbf{intI}$$

$$\frac{}{\Delta;\Gamma;\rho \vdash_Z n : \mathtt{int}} \ \textbf{int} \qquad \frac{\Delta \vdash \tau\, \mathtt{wf}}{\Delta;\Gamma;\rho \vdash_Z \mathtt{nil} : \mathtt{list}\,\tau} \ \textbf{nil}$$

$$\frac{\Delta;\Gamma;\rho \vdash_Z e : \mathsf{DICT}(\tau_i \to {\tau'_i}^{i \in 1,2 \ldots n}) \qquad Z' = 1 + Z}{\Delta;\Gamma;\rho \vdash_{Z'} \delta(e) : \mathtt{real}} \ \delta$$

$$\frac{\Delta;\Gamma;\rho_1 \vdash_{Z_1} e_1 : \tau \qquad \Delta;\Gamma;\rho_2 \vdash_{Z_2} e_2 : \mathtt{list}\,\tau \qquad Z' = \max(Z_1, Z_2) \qquad \rho' = \max(\rho_1, \rho_2)}{\Delta;\Gamma;\rho' \vdash_{Z'} \mathtt{cons}(e_1, e_2) : \mathtt{list}\,\tau} \ \textbf{cons}$$

$$\frac{\Delta;\Gamma;\rho_1 \vdash_{Z_1} e_1 : \tau_1 \qquad \Delta;\Gamma, x : \tau_1;\rho_2 \vdash_{Z_2} e_2 : \tau \qquad Z' = \max(Z_2, Z_1 + q) \qquad \rho' = \max(\rho_2, \rho_1 + q)}{\Delta;\Gamma \vdash_{Z'} \mathtt{let}\, x; q = e_1\, \mathtt{in}\, e_2 : \tau} \ \textbf{let}$$

$$\frac{i, \Delta;\Gamma;\rho \vdash_Z e : \tau \qquad i \notin \mathsf{FIV}(\Gamma)}{\Delta;\Gamma;\rho \vdash_{Z'} \Lambda.e : \forall i \overset{,Z}{::} S.\,\tau} \ \textbf{ilam}$$

$$\frac{\Delta;\Gamma;\rho \vdash_Z e : \forall i \overset{,Z_1}{::} S.\,\tau \qquad \Delta \vdash I :: S \qquad Z' = Z_1[I/i] + Z}{\Delta;\Gamma;\rho \vdash_{Z'} e\,[] : \tau[I/i]} \ \textbf{iapp}$$

$$\frac{\Delta;\Gamma;\rho \vdash_{Z'_i} e'_i : \tau'_i \qquad Z' = \max({Z'_i}^{1,2 \ldots n})}{\Delta;\Gamma;\rho \vdash_{Z'} \mathtt{dict}(\mathsf{attr}_i \to {e'_i}^{i \in 1,2 \ldots n}) : \mathsf{DICT}(\tau_i \to {\tau'_i}^{i \in 1,2 \ldots n})} \ \textbf{dict}$$

Figure 8: Typing rules, part 2

$$
\begin{array}{lcl}
[\![\texttt{bool}]\!]_{\mathrm{V}} & = & \{(k, \texttt{true}) \mid k \in \mathbb{N}\} \cup \{(k, \texttt{false}) \mid k \in \mathbb{N}\} \\
[\![\texttt{b}]\!]_{\mathrm{V}} & = & \{(k, c) \mid k \in \mathbb{N} \wedge c : \texttt{b}\} \\
[\![\tau_1 \times \tau_2]\!]_{\mathrm{V}} & = & \{(k, (v_1, v_2)) \mid (k, v_1) \in [\![\tau_1]\!]_{\mathrm{V}} \wedge (k, v_2) \in [\![\tau_2]\!]_{\mathrm{V}}\} \\
[\![\tau_1; q \xrightarrow{\rho; Z} \tau_2]\!]_{\mathrm{V}} & = & \{(k, (\texttt{fix}\, f(x).e, \theta)) \mid \forall j < k.\ \forall (j, v) \in [\![\tau_1]\!]_{\mathrm{V}}. \\
 & & \quad (j, (\theta[x \mapsto v, f \mapsto (\texttt{fix}\, f(x).e, \theta)], e)) \in [\![\tau_2]\!]_{\mathrm{E}}^{\rho[x:q, f:\infty], Z}\} \\
[\![\tau_1; \xrightarrow{q; Z} \tau_2]\!]_{\mathrm{V}} & = & \{(k, (\texttt{fix}\, f(x).e, \theta)) \mid \forall j < k.\ \forall (j, v) \in [\![\tau_1]\!]_{\mathrm{V}}. \\
 & & \quad (\theta[x \mapsto v, f \mapsto (\texttt{fix}\, f(x).e, \theta)], e \Downarrow^{v}, T) \wedge |T| = j' \leq j \\
 & & \wedge\ \texttt{depth}_x(T) \leq q \wedge \texttt{adap}(T) \leq Z \wedge v \in (j - j', v) \in [\![\tau_2]\!]_{\mathrm{V}}\} \\
[\![\texttt{list}\,\tau]\!]_{\mathrm{V}} & = & \{(k, \texttt{nil}) \mid k \in \mathbb{N}\} \cup \{(k, \texttt{cons}(v_1, v_2)) \mid (k, v_1) \in [\![\tau]\!]_{\mathrm{V}} \wedge (k, v_2) \in [\![\texttt{list}\,\tau]\!]_{\mathrm{V}}\} \\
 & & \\
[\![\tau]\!]_{\mathrm{E}}^{\rho, Z} & = & \{(k, (\theta, e)) \mid \forall v\ T\ j.\ (\theta, e \Downarrow^{v}, T) \wedge (|T| = j) \wedge (j \leq k) \\
 & & \qquad \Rightarrow (\texttt{adap}(T) \leq Z\ \wedge \\
 & & \qquad\quad \forall x \in \text{Vars}.\ \texttt{depth}_x(T) \leq \rho(x)\ \wedge \\
 & & \qquad\quad ((k - j, v) \in [\![\tau]\!]_{\mathrm{V}})\} \\
[\![\texttt{int}]\!]_{\mathrm{V}} & = & \{(k, i) \mid k \in \mathbb{N} \wedge i : \texttt{int}\} \\
[\![\texttt{real}]\!]_{\mathrm{V}} & = & \{(k, r) \mid k \in \mathbb{N} \wedge r : \texttt{real}\} \\
[\![\forall i \overset{\rho, Z}{::} S.\, \tau]\!]_{\mathrm{V}} & = & \{(k, (\Lambda.e, \theta)) \mid k \in \mathbb{N} \wedge \forall I.\ \vdash I :: S, (k, e) \in [\![\tau[I/i]]\!]_{\mathrm{E}}^{\rho, Z[I/i]}\} \\
[\![\texttt{int}[I]]\!]_{\mathrm{V}} & = & \{(k, n) \mid k \in \mathbb{N} \wedge n = I\}
\end{array}
$$

Figure 9: Logical relation with step-indexing

Theorem 15 (Fundamental theorem)**.** If $\Delta; \Gamma; \rho \vdash_Z e : \tau$ and $\sigma \in [\![\Delta]\!]_{\mathrm{V}}$ and $(k, \theta) \in [\![\sigma\Gamma]\!]_{\mathrm{V}}$, then $(k, (\theta, \sigma e)) \in [\![\sigma\tau]\!]_{\mathrm{E}}^{\rho, \sigma Z}$.

Proof. By induction on the given typing derivation. For the case of `fix`, we subinduct on the step index.

Case $\boxed{\dfrac{\Delta; \Gamma, f : (\tau_1; \xrightarrow{q; Z} \tau_2), x : \tau_1; \rho, [x : q] \vdash_Z e : \tau_2\ \ (1)}{\Delta; \Gamma; \rho \vdash_0 \texttt{fix}\, f(x).e : \tau_1; \xrightarrow{q; Z} (\tau_2)}\ \textbf{fix}}$

Assume $\sigma \in [\![\Delta]\!]_{\mathrm{V}}$, $(k, \theta) \in [\![\sigma\Gamma]\!]_{\mathrm{V}}$.
TS: $(k, (\theta, \sigma\, \texttt{fix}\, f(x).e)) \in [\![\sigma(\tau_1; \xrightarrow{q; Z} \tau_2)]\!]_{\mathrm{E}}^{\rho, 0}$.
By inversion, STS: $\forall v, T, j\ (\theta, \sigma\, \texttt{fix}\, f(x).e \Downarrow^{v}, T) \wedge (|T| = j) \wedge (j \leq k)$
1. $(\texttt{adap}(T) \leq 0)$;
2. $(\forall x \in \text{Vars}.\texttt{depth}_x(T) \leq \rho(x))$;
3. $((k - j), v) \in [\![\sigma(\tau_1; \xrightarrow{q; Z} \tau_2)]\!]_{\mathrm{V}}$.
By E-FIX, let $v = (\sigma\, \texttt{fix}\, f(x).e, \theta)$, $T = (\sigma\, \texttt{fix}\, f(x).e, \theta)$ we know:
(2). $(\theta, \sigma\, \texttt{fix}\, f(x).e) \Downarrow^{(} (\sigma\, \texttt{fix}\, f(x).e, \theta), \sigma\, \texttt{fix}\, f(x).e)$;
(3). $|(\sigma\, \texttt{fix}\, f(x).e, \theta)| = j \wedge j < k$.
Suppose $(2), (3)$, STS:

1. $\texttt{adap}(\sigma\, \texttt{fix}\, f(x).e) = 0 \leq 0$;
2. $\forall x \in \text{Vars}.\texttt{depth}_x((\sigma\, \texttt{fix}\, f(x).e, \theta)) = \bot \leq \rho(x)$;
3. $((k-j), (\sigma\, \texttt{fix}\, f(x).e, \theta)) \in [\![\sigma(\tau_1; \xrightarrow{q;Z} \tau_2)]\!]_{\text{V}}$.

1. and 2. are proved by definition.
The third statement is proved by a general theorem:
Set $k - j = k'$, $\forall m \leq k', (m, (\sigma\, \texttt{fix}\, f(x).e, \theta)) \in [\![\sigma(\tau_1; \xrightarrow{q;Z} \tau_2)]\!]_{\text{V}}$.
Induction on m:

Subcase 1: $m = 0$,

TS: $\forall j' < 0.(j', v_m) \in [\![\sigma\tau_1]\!]_{\text{V}}$, $(\theta[x \mapsto v_m, f \mapsto (\sigma\, \texttt{fix}\, f(x).e, \sigma\theta)], e \Downarrow^v, T) \ldots$

it is obviously true because $j' < 0 \notin \mathbb{N}$.

Subcase 2: $m = m' + 1 \leq k'$,

TS: $(m, (\sigma\, \texttt{fix}\, f(x).e, \theta)) \in [\![\sigma(\tau_1; \xrightarrow{q;Z} \tau_2)]\!]_{\text{V}}$.
Pick $\forall j' < m' + 1$, $\forall (j', v_m) \in [\![\tau_1]\!]_{\text{V}}$,

STS: $(\theta[x \mapsto v_m, f \mapsto (\sigma\, \texttt{fix}\, f(x).e, \theta)], \sigma e) \Downarrow^v, T \wedge |T| = j'' \wedge \texttt{adap}(T) \leq \sigma Z \wedge \texttt{depth}_x(T) \leq q \wedge (j' - j'', v) \in [\![\sigma\tau_2]\!]_{\text{V}}$ (4).

By sub ih, we have: (5). $(m', \sigma(\texttt{fix}\, f(x).e, \theta)) \in [\![\sigma(\tau_1; q \xrightarrow{\rho;Z} \tau_2)]\!]_{\text{V}}$.
Pick $\theta' = \theta[x \mapsto v_m, f \mapsto \sigma(\texttt{fix}\, f(x).e, \theta)]$,

So we know:

$$(j', \theta') \in [\![\Gamma, f : \sigma(\tau_1; q \xrightarrow{\rho;Z} \tau_2), x : \sigma\tau_1]\!]_{\text{V}} \text{ (6)}$$

proved by:

a) $(k, \theta) \in [\![\Gamma]\!]_{\text{V}}$, applying Lemma **??** on assumption, we get: $(j', \theta) \in [\![\Gamma]\!]_{\text{V}}$.
b) $(j', v_m) \in [\![\sigma\tau_1]\!]_{\text{V}}$, from the assumption.
c) $(j', (\texttt{fix}\, f(x).e, \theta)) \in [\![\sigma(\tau_1; q \xrightarrow{\rho;Z} \tau_2)]\!]_{\text{V}}$ from (5).

by induction hypothesis on (1) and (6), we conclude that:

$$(j', (\theta[x \mapsto v_m, f \mapsto \sigma(\texttt{fix}\, f(x).e, \theta)], e)) \in [\![\sigma\tau_2]\!]_{\text{E}}^{\rho[x:q], \sigma Z}$$

Unfold the conclusion, we get: $(\theta[x \mapsto v_m, f \mapsto (\sigma\, \texttt{fix}\, f(x).e, \theta)], \sigma e) \Downarrow^v, T \wedge |T| = j'' \leq j' \wedge \texttt{adap}(T) \leq \sigma Z \wedge \forall x \in \text{Vars}.\ \texttt{depth}_x(T) \leq \rho[x : q](x) \wedge (j' - j'', v) \in [\![\sigma\tau_2]\!]_{\text{V}}$. (4) is proved by the above statements.

Case
$$\boxed{\frac{\Delta; \Gamma; \rho_1 \vdash_{Z_1} e_1 : \tau_1; \xrightarrow{q;Z} \tau_2 \ (\star) \qquad \Delta; \Gamma; \rho_2 \vdash_{Z_2} e_2 : \tau_1 \ (\diamond) \qquad Z' = Z_1 + \max(Z, Z_2 + q) \qquad \rho' = \max(\rho_1, Z_1 + \rho_2)}{\Delta; \Gamma; \rho' \vdash_{Z'} e_1\ e_2 : \tau_2}\ \textbf{app}}$$

Assume $\sigma \in [\![\Delta]\!]_{\mathrm{V}}$, $(k, \theta) \in [\![\sigma\Gamma]\!]_{\mathrm{V}}$ $(\triangle)$.
TS: $(k, (\theta, \sigma(e_1\ e_2))) \in [\![\sigma\tau_2]\!]_{\mathrm{E}}^{\rho', \sigma Z'}$.
By inversion, pick any v, T, j s.t. $((\theta, e_1\ e_2) \Downarrow^{(} v, T)) \wedge (|T| = j) \wedge (j \leq k)$,
STS:
1. $(\texttt{adap}(T) \leq \sigma Z')$;
2. $(\forall x \in \mathrm{Vars}.\texttt{depth}_x(T) \leq \rho'(x))$;
3. $(((k - j), v) \in [\![\sigma\tau_2]\!]_{\mathrm{V}})$.
By ih on $\star$ and $\triangle$, we get: (1) $(k, (\theta, \sigma e_1)) \in [\![\sigma(\tau_1; \xrightarrow{q;\, Z} \tau_2)]\!]_{\mathrm{E}}^{\rho_1, \sigma Z_1}$.
Inversion on (1), we get:
Pick any v_1, T_1, j_1, s.t. $((\theta, \sigma e_1) \Downarrow^{(} v_1, T_1))$ (a) $\wedge (|T_1| = j_1) \wedge (j_1 < k)$,
we know:
(2) $(\texttt{adap}(T_1) \leq \sigma Z_1)$;
(3) $(\forall x \in \mathrm{Vars}.\texttt{depth}_x(T_1) \leq \rho_1(x))$;
(4) $((k - j_1), v_1) \in [\![\sigma(\tau_1; \xrightarrow{q;\, Z} \tau_2)]\!]_{\mathrm{V}}$.
v_1 is a function by definition.
Let $v_1 = (\texttt{fix}\, f(x).e, \theta')$ (b) s.t. $T_1 = (\texttt{fix}\, f(x).e, \theta)$
By inversion on (4), we know: $\forall j' < (k - j_1) \wedge (j', v') \in [\![\sigma\tau_1]\!]_{\mathrm{V}}$,
$(\theta'[x \mapsto v', f \mapsto (\texttt{fix}\, f(x).e, \theta')], e) \Downarrow^{v''}, T''$ (d) $\wedge\ |T''| = j'' \leq j' \wedge \texttt{adap}(T') \leq \sigma Z \wedge \texttt{depth}_x(T') \leq \sigma q \wedge (j' - j'', v'') \in [\![\sigma\tau_2]\!]_{\mathrm{V}}$ (5).

By ih on $\diamond$ and $\square$, we get: $(k, (\theta, \sigma e_2)) \in [\![\sigma\tau_1]\!]_{\mathrm{E}}^{\rho_2, \sigma Z_2}$ (6).
Inversion on (6), we get:
Pick any v_2, T_2, j_2, s.t. $((\theta, \sigma e_2) \Downarrow^{(} v_2, T_2))$ (c) $\wedge (|T_2| = j_2) \wedge (j_2 \leq k)$,
we know:
(7). $(\texttt{adap}(T_2) \leq \sigma Z_2)$;
(8). $(\forall x \in \mathrm{Vars}.\texttt{depth}_x(T_2) \leq \rho_2(x))$;
(9). $((k - j_2), v_2) \in [\![\sigma\tau_1]\!]_{\mathrm{V}}$
Apply Lemma **??** on (9), we have $((k - j_2 - j_1 - 1), v_2) \in [\![\sigma\tau_1]\!]_{\mathrm{V}}$
Pick $j' = k - j_1 - j_2 - 1, v' = v_2$, from (5), we have:
(11). $\texttt{adap}(T'') \leq \sigma Z$;
(12). $\texttt{depth}_x(T'') \leq q$;
(13). $(k - j_1 - j_2 - j'' - 1, v'') \in [\![\sigma\tau_2]\!]_{\mathrm{V}}$.
Apply E-APP rule on $(a)(b)(c)(d)$ we have:

$$\frac{\begin{array}{cc} \theta, \sigma e_1 \Downarrow^{v}{}_1, T_1\ (a) & v_1 = (\texttt{fix}\, f(x).e, \theta')\ (b) \\ \theta, \sigma e_2 \Downarrow^{v}{}_2, T_2\ (c) & \theta'[f \mapsto v_1, x \mapsto v_2], e \Downarrow^{v''}, T''\ (d) \end{array}}{\theta, \sigma(e_1\ e_2) \Downarrow^{v''}, T_1\ T_2 \triangleright \texttt{fix}\, f(x).T''}$$

Pick $v = v'', j = j_1 + j_2 + j'' + 1, T = T_1\ T_2 \triangleright \texttt{fix}\, f(x).T''$ s.t. $\theta, e_1\ e_2 \Downarrow^{v}, T \wedge |T| = j \wedge j \leq k$.
Suffice to show the follwing three:
1. $\texttt{adap}(T) = \texttt{adap}(T_1\ T_2 \triangleright \texttt{fix}\, f(x).T'') = \texttt{adap}(T_1) + \max(\texttt{adap}(T''), \texttt{adap}(T_2) + \texttt{depth}_x(T'')) \leq \sigma Z_1 + \max(\sigma Z, \sigma Z_2 + q) = \sigma Z'$ proved by $(2), (7), (11), (12)$.

2. $\forall y \in \text{Vars}.\mathtt{depth}_y(T) = \max(\mathtt{depth}_y(T_1), \mathtt{adap}(T_1) + \max(\mathtt{depth}_y(T''), \mathtt{depth}_y(T_2) + \mathtt{depth}_x(T''))) \leq \max(\rho_1, Z_1 + \max(\rho, \rho_2 + q)) = \rho'(y)$ proved by (2), (3), (8), (12).
3. $(k - j, v) = (k - j_1 - j_2 - j'' - 1, v'') \in [\![\sigma\tau_2]\!]_\mathrm{V}$ proved by (13).

Case $\boxed{\dfrac{\Delta; \Gamma; \rho \vdash_Z e : \mathsf{DICT}(\tau_i \to \tau_i'^{\,i \in 1,2\ldots n}) \qquad Z' = 1 + Z}{\Delta; \Gamma; \rho \vdash_{Z'} \delta(e) : \mathtt{real}}\ \delta}$

Assume $\sigma \in [\![\Delta]\!]_\mathrm{V}$, $(k, \theta) \in [\![\sigma\Gamma]\!]_\mathrm{V}$, TS: $(k, \sigma(\delta(e), \theta)) \in [\![\mathtt{real}]\!]_\mathrm{E}^{\rho', \sigma Z'}$.
Unfold, pick v, T, assume $(\sigma\theta, \sigma\delta(e) \Downarrow^v, T) \wedge (|T| = j) \wedge (j \leq k)$.
STS: 1. $(\mathtt{adap}(T) \leq \sigma Z')$
2. $(\forall x \in \text{Vars}.\mathtt{depth}_x(T) \leq \rho'(x))$
3. $((k - j, v) \in [\![\mathtt{real}]\!]_\mathrm{V})$.
From the evaluation rules, we assume that :

$$\frac{\sigma\theta, \sigma e \Downarrow^{v'}, T' \qquad \delta(v') = v}{\theta, \delta(e) \Downarrow^{v'}, \delta(T')}$$

By induction hypothesis on $\star$, we get:

$$(k, (\sigma e, \sigma\theta)) \in [\![\square\,((\tau_1; 0 \xrightarrow{\rho''; 0} \tau_2))]\!]_\mathrm{E}^{\rho, \sigma Z}\ (1)$$

.
Assume $\sigma\theta, \sigma e \Downarrow^{v'}, T' \wedge |T'| = j' \wedge j' < k$,
we know: $(\mathtt{adap}(T') \leq \sigma Z)\ (a)$
$(\forall x \in \mathtt{depth}_x(T') \leq \rho(x))\ (b)$
$((k - j', v') \in [\![\square\,((\tau_1; 0 \xrightarrow{\rho''; 0} \tau_2))]\!]_\mathrm{V})\ (c)$

STS1: $\mathtt{adap}(T) = \mathtt{adap}(\delta(T')) \leq \sigma Z'$
Unfold $\mathtt{adap}(\delta(T'))$, STS:
$1 + \mathtt{adap}(T') + \mathsf{MAX}_{v \in \tau_1}\Big(\max\big(\mathtt{adap}(T_3(v)), \mathtt{depth}_x(T_3(v))\big)\Big) \leq \sigma Z'$.
where $v_1 = (\mathtt{fix}\, f(x : \tau_1).e_1, \theta_1) = \mathsf{extract}(T')$ and $\theta_1[f \mapsto v_1, x \mapsto v], e_1 \Downarrow^{v''}, T_3(v)$
By Lemma**??** based on our assumption $\sigma\theta, \sigma e \Downarrow^{v'}, T'$, we know that $v_1 = \mathsf{extract}(T) = v'$.
Unfold (c), we know : $(k - j', (\mathtt{fix}\, f(x : \tau_1).e_1, \theta_1)) \in [\![(\tau_1; 0 \xrightarrow{\rho''; 0} \tau_2)]\!]_\mathrm{V}$ (d) and $\delta \notin (\mathtt{fix}\, f(x : \tau).e_1, \theta_1)$
Unfold (d), we get : $\forall j_1 < (k - j').(j_1, v_a) \in [\![\tau_1]\!]_\mathrm{V}$, $(j_1, (e_1, \theta_1[f \mapsto v_1, x \mapsto v_a])) \in [\![\tau_2]\!]_\mathrm{E}^{\rho''[x:0, f:\infty], 0}$ (e).
Pick v_a, unfold (e), we assume: $\theta_1[f \mapsto v_1, x \mapsto v_a], e_1 \Downarrow^{v''}, T_3(v_a) \wedge |T_3(v_a)| = j_2 \wedge j_2 \leq j_1$.
we get: $\mathtt{adap}(T_3(v_a)) \leq 0 (f)$

$\forall x \in \mathtt{depth}_x(T_3(v_a)) \leq \rho''[x:0, f:\infty](x)$ (h)

From (f), we know $\forall v_a \in [\![\tau_1]\!]_{\mathrm{V}}.\mathtt{adap}(T_3(v_a)) = 0$.
By Lemma **??**, we conclude that $\mathsf{MAX}_{v\in\tau_1}\Big(\max\big(\mathtt{adap}(T_3(v)), \mathtt{depth}_x(T_3(v))\big)\Big) \leq 0$ (g).
This property is proved by $(a), (g)$.

STS2: $\mathtt{depth}_x(T) = \mathtt{depth}_x(\delta(T')) = 1+\max(\mathtt{depth}_x(T'), \mathtt{adap}(T')+\mathsf{MAX}_{v\in\tau_1}\Big(\max(\mathtt{depth}_x(T_3(v)), \perp)\Big)$
$\leq \rho'(x)$.
It is proved by $(a), (b), (h)$.

STS3: $((k-j, v) \in [\![\mathtt{real}]\!]_{\mathrm{V}})$
It is proved by the property of the constuct δ whose codomain is real number.

$\square$

References

[1] Roly Perera, Umut A. Acar, James Cheney, and Paul Blain Levy. Functional programs that explain their work. In *Proc. ICFP*, 2012.