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 Background: Adaptive Data Analysis

1.1 Motivation

Consider a dataset X consisting of n independent samples from some unknown population P. How can we ensure that the conclusions drawn from X *generalize* to the population P? Despite decades of research in statistics and machine learning on methods for ensuring generalization, there is an increased recognition that many scientific findings generalize poorly (e.g. [**?**]). While there are many reasons a conclusion might fail to generalize, one that is receiving increasing attention is *adaptivity*, which occurs when the choice of method for analyzing the dataset depends on previous interactions with the same dataset [**?**].

Adaptivity can arise from many common practices, such as exploratory data analysis, using the same data set for feature selection and regression, and the re-use of datasets across research projects. Unfortunately, adaptivity invalidates traditional methods for ensuring generalization and statistical validity, which assume that the method is selected independently of the data. The misinterpretation of adaptively selected results has even been blamed for a "statistical crisis" in empirical science [**?**].

A recent line of work initiated by Dwork *et al.* [**?**] and Hardt and Ullman [**?**] posed the question: Can we design *general-purpose* methods that ensure generalization in the presence of adaptivity, together with guarantees on their accuracy? This line of work has identified many new algorithmic techniques for ensuring generalization in adaptive data analysis, leading to algorithms with greater statistical power than all previous approaches. It has also identified problematic strategies for adaptive analysis, showing limitations on the statistical power one can hope to achieve.

A key development in this line of work is that the best method for ensuring generalization in an adaptive data analysis depends to a large extent on the number of *rounds of adaptivity*. That is, not only does the analysis of the generalization error depend on the number of rounds, but knowing the number of rounds actually allows one to choose methods that lead to the smallest possible generalization error.

1.2 Review of Prior Work

To explain the key concepts and motivate our work, we review the model of adaptive data analysis of [**?**, **?**]. We remark that many of the details of the model are immaterial for our work, and are included for illustrative purposes and concreteness.

In the model there is some distribution P over a domain $\mathcal{X}$ that an analyst would like to study. For our purposes, the analyst wishes to answer *statistical queries* (also known as *linear queries* or *linear functionals*) on the distribution. A statistical query is defined by some function $f : \mathcal{X} \to [-1, 1]$. The analyst wants to learn the *population mean*, which (abusing notation) is defined as

$$f(P) = \mathop{\mathbb{E}}_{X \sim P}[f(X)].$$

However, the distribution P can only be accessed via a set of *samples* $X_1, \ldots, X_n$ drawn identically and independently from P. These samples are held by a mechanism $M(X_1, \ldots, X_n)$ who receives the query f and computes an answer $a \approx f(P)$.

In this work we consider analysts that ask a sequence of k queries $f_1, \ldots, f_k$. If the queries are all chosen in advance, independently of the answers then we say they are *non-adaptive*. If the choice of each query f_j may depend on the prefix $f_1, a_1, \ldots, f_{j-1}, a_{j-1}$ then they are *fully adaptive*. An important intermediate notion is *r-round adaptive*, where the sequence can be partitioned into r batches of non-adaptive queries. Note that non-interactive queries are 1-round and fully adaptive queries are k rounds.

We now review what is known about the problem of answering r-round adaptive queries. Given the samples, the naïve way to approximate the population mean is to use the *empirical mean*, which (abusing notation) is

defined as

$$f(X_1, \ldots, X_n) = \frac{1}{n} \sum_{i=1}^{n} f(X_i).$$

Theorem 1. For any distribution P, and any k *non-adaptive* statistical queries, the naïve mechanism satisfies

$$\max_{j=1,\ldots,k} |a_j - f_j(P)| = O\left(\sqrt{\frac{\log k}{n}}\right)$$

For any $r \geq 2$ and any r*-round adaptive* statistical queries, it satisfies

$$\max_{j=1,\ldots,k} |a_j - f_j(P)| = O\left(\sqrt{\frac{k}{n}}\right)$$

And there exists analysts that make these bounds tight (up to constant factors).

Note that even allowing one extra round of adaptivity leads to an exponential increase in the generalization error from $\log k$ to k.

Perhaps surprisingly, Dwork *et al.* [**?**] and Bassily *et al.* [**?**] showed that an alternative mechanism can actually achieve much stronger generalization error as a function of the number of queries, specifically,

Theorem 2 ([**?**, **?**])**.** For any k, there exists a mechanism such that for any distribution P, and any $r \geq 2$ any r*-round adaptive* statistical queries, it satisfies

$$\max_{j=1,\ldots,k} |a_j - f_j(P)| = O\left(\frac{\sqrt[4]{k}}{\sqrt{n}}\right)$$

And there is an analyst that make this bound tight (up to constant factors).

Notice that Theorem 3 has different quantification in that the optimal choice of mechanism depends on the number of queries. Thus, we need to know the number of queries *a priori* to choose the best mechanism.[1]

Later work by Dwork *et al.* [**?**] gave more refined bounds in terms of the number of rounds of adaptivity. Similarly, if one knows a good *a priori* upper bound on the number of rounds of adaptivity, one can get a much better guarantee of generalization error, but only by using an appropriate choice of mechanism.[1] Specifically,

[1] One can, in principle, avoid knowing the number of queries and rounds *a priori* using a "guess-and-double" strategy, however this would weaken the bound on generalization error considerably.

Theorem 3 ([?])**.** For any r and k, there exists a mechanism such that for any distribution P, and any $r \geq 2$ any r-*round adaptive* statistical queries, it satisfies

$$\max_{j=1,\ldots,k} |a_j - f_j(P)| = O\left(\frac{r\sqrt{\log k}}{\sqrt{n}}\right)$$

And there is an analyst that make this bound tight (up to constant factors).

2 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 $\perp$ 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 `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 $\texttt{b} \to \texttt{bool}$. We make environments explicit in closures. This is needed for the tracing semantics

later.

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

3 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 [2, Section 4], but we omit their "holes" for which we have no need. The construct $T_1\ T_2 \triangleright \texttt{fix}\, f(x).T_3$ 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.

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

Big-step tracing semantics The big-step, tracing semantics $\theta, e \Downarrow v, T$ 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 [2], are shown in Figure 1. 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.

Adaptivity of a trace We define the *adaptivity* of a trace T, $\texttt{adap}(T)$, 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 $\texttt{depth}_x(T)$, 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

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

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

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

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

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

$$\frac{\theta, e_1 \Downarrow v_1, T_1 \qquad \theta, e_2 \Downarrow v_2, T_2}{\theta, (e_1, e_2) \Downarrow (v_1, v_2), (T_1, T_2)} \qquad \frac{\theta, e \Downarrow (v_1, v_2), T}{\theta, \texttt{fst}(e) \Downarrow v_1, \texttt{fst}(T)}$$

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

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

$$\frac{}{\theta, \texttt{nil} \Downarrow \texttt{nil}, \texttt{nil}} \qquad \frac{\theta, e_1 \Downarrow v_1, T_1 \qquad \theta, e_2 \Downarrow v_2, T_2}{\theta, \texttt{cons}(e_1, e_2) \Downarrow \texttt{cons}(v_1, v_2), \texttt{cons}(T_1, T_2)}$$

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

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

Figure 1: Big-step semantics with provenance

into account. Technically, $\texttt{adap} : \text{Traces} \to \mathbb{N}$ and $\texttt{depth}_x : \text{Traces} \to \mathbb{N}_\perp$. If x does not appear free in T, $\texttt{depth}_x(T)$ is $\perp$.

The functions $\texttt{adap}$ and $\texttt{depth}_x$ are defined by mutual induction in Figure 23.

Explanation of `adap` We explain the interesting cases of the definition of $\texttt{adap}$. The case $T_1\ T_2 \triangleright \texttt{fix}\, f(x).T_3$ 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 $\texttt{adap}(T_1) + \max(\texttt{adap}(T_3), \texttt{adap}(T_2) + \texttt{depth}_x(T_3))$. The term $\texttt{adap}(T_1)$ occurs additively since the entire computation is control-dependent on the function the head of the application evaluates to. The rest of the term $\max(\texttt{adap}(T_3), \texttt{adap}(T_2) + \texttt{depth}_x(T_3))$ 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, $\texttt{depth}_x(T_3)$ is the maximum δ-nesting depth of x in the body. Hence, $\max(\texttt{adap}(T_3), \texttt{adap}(T_2) + \texttt{depth}_x(T_3))$ 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), $\texttt{depth}_x(T_3) = \perp$, so $\max(\texttt{adap}(T_3), \texttt{adap}(T_2) + \texttt{depth}_x(T_3)) = \max(\texttt{adap}(T_3), \texttt{adap}(T_2) + \perp) = \max(\texttt{adap}(T_3), \perp) = \texttt{adap}(T_3)$, which is the adaptivity of the body in the absence of dependency from x.

The case $\texttt{if}^\texttt{t}(T_b, T_t)$ corresponds to the evaluation of $\texttt{if}(e_b, e_t, \_)$ where e_b evaluates to $\texttt{true}$ 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 $\texttt{adap}(T_b) + \texttt{adap}(T_t)$.

Explanation of $\texttt{depth}_x$ We explain interesting cases in the definition of $\texttt{depth}_x$. For the trace $T_1\ T_2 \triangleright \texttt{fix}\, f(y).T_3$, $\texttt{depth}_x$ is defined as $\max(\texttt{depth}_x(T_1), \texttt{adap}(T_1) + \max(\texttt{depth}_x(T_3), \texttt{depth}_x(T_2) + \texttt{depth}_y(T_3)))$. Here, $\max(\texttt{depth}_x(T_3), \texttt{depth}_x(T_2) + \texttt{depth}_y(T_3))$ 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, $\texttt{depth}_y(T_3) = \perp$, and this term is $\texttt{depth}_x(T_3)$. 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

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

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

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

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

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 4. 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 4 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 $\max(a, b + c) = b + c$.

Lemma 5. Exists a function $\mathsf{extract}$, if $\theta, e \Downarrow v, T$, then $\mathsf{extract}(T) = v$.

Proof. By induction on the derivation of the operational semantics.
Case

$$\frac{}{\theta, x \Downarrow \theta(x), (x, \theta)}$$

$\mathsf{extract}((x, \theta)) = \theta(x) \Rightarrow \mathsf{extract} = \lambda x.\mathsf{fst}(x)\ \mathsf{snd}(x)$.

Case

$$\frac{\theta, e_1 \Downarrow v_1, T_1 \qquad v_1 = (\texttt{fix}\, f(x).e, \theta') \qquad \theta, e_2 \Downarrow v_2, T_2 \qquad \theta'[f \mapsto v_1, x \mapsto v_2], e \Downarrow v, T}{\theta, e_1\ e_2 \Downarrow v, T_1\ T_2 \triangleright \texttt{fix}\, f(x).T}$$

By IH on the first premise, we get : exists $\mathsf{extract}_1(T_1) = v_1$ and because we know v_1 is a function, we know from the second premise : $\mathsf{extract}_1(T_1) = v_1 = (\texttt{fix}\, f(x).e, \theta')$.
By IH on the thrid premise, we get: exists $\mathsf{extract}_2(T_2) = v_2$.
By IH on the fourth premise, we know that : exists $\mathsf{extract}_3(T) = v$.
So we know : exists $\mathsf{extract}$ so that $\mathsf{extract}(T_1\ T_2 \triangleright \texttt{fix}\, f(x).T) = \mathsf{extract}_3(T) = v$.

Case

$$\frac{\theta, e_1 \Downarrow v_1, T_1 \qquad \theta, e_2 \Downarrow v_2, T_2}{\theta, (e_1, e_2) \Downarrow (v_1, v_2), (T_1, T_2)}$$

By IH on the first premise, we get : exists $\mathsf{extract}_1(T_1) = v_1$. By IH on the second premise, we get : exists $\mathsf{extract}_2(T_2) = v_2$.
So we know : exists $\mathsf{extract}((T_1, T_2)) = (\mathsf{extract}_1(T_1), \mathsf{extract}_2(T_2))$.

Case

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

By IH on the first premise, exists $\mathsf{extract}_1(T) = v$.
We know : exists $\mathsf{extract}(\delta(T)) = \delta(\mathsf{extract}_1(T)) = v'$.

Case

$$\frac{\theta, e \Downarrow \mathtt{true}, T \qquad \theta, e_1 \Downarrow v, T_1}{\theta, \mathtt{if}(e, e_1, e_2) \Downarrow v, \mathtt{if}^{\mathsf{t}}(T, T_1)}$$

By IH on the second premise, exists $\mathsf{extract}_1(T_1) = v$.
We know : exists $\mathsf{extract}(\mathtt{if}^{\mathsf{t}}(T, T_1)) = \mathsf{extract}_1(T_1) = v$.
Case

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

By IH on the second premise, exists $\mathsf{extract}_2(T_2) = v$.
exists $\mathsf{extract}(\mathtt{let}(x, T_1, T_2)) = \mathsf{extract}_2(T_2) = v$.

□

4 Type system

We build a type system that statically approximates both $\texttt{adap}$ and $\texttt{depth}_x$ for every x in the context.

A *depth map*, denoted ρ, is a map from variables to $\mathbb{N}^\infty_\bot$. Our typing judgment takes the form $\Gamma; \rho \vdash_Z e : \tau$, where Γ is a typing context, ρ is a depth map, and $Z \in \mathbb{N}$. The idea is that ρ gives an upper bound on the depth of each free variable of e. Obviously, we only care about the values of ρ on the domain of Γ (at the remaining points, ρ can be $\bot$; such a ρ can always be finitely represented as $x_1 : q_1, \ldots, x_k : q_k$, where $x_1, \ldots, x_k$ are bound by Γ). Z is an upper bound on the adaptivity of e.

Types Types τ are simple, except that the function type is annotated with the adaptivity Z of the function body, a depth-map ρ which gives the depths of all free variables of the body (i.e., variables bound in outer scopes), and a depth $q \in \mathbb{N}^\infty_\bot$ of the argument variable in the body. The function name, although free in the body, always has depth ∞, so we don't write its depth explicitly.[2]

Index Term	I, Z	$::=$	$i \mid n \mid I_1 + I_2 \mid I_1 - I_2 \mid \texttt{max}(I_1, I_2)$
Sort	S	$::=$	$\mathbb{N}$
Type	τ	$::=$	$\texttt{b} \mid \texttt{bool} \mid \tau_1 \times \tau_2 \mid \tau_1; q \xrightarrow{\rho; Z} \tau_2 \mid \texttt{list}\,\tau \mid \square(\tau) \mid$ $\texttt{real} \mid \texttt{int} \mid \texttt{int}[I] \mid \forall i \overset{\rho, Z}{::} S.\, \tau$

Typing rules The typing rules are shown in Figure 24. In these rules, we write $q + \rho$ or $\rho + q$ for the depth map $\lambda x.\ (q + \rho(x))$ defined on the same domain as ρ. We also lift max and $+$ pointwise to depth maps. These are also formally defined in Figure 24.

[2]The function name is never substituted by a value with a nontrivial trace, so we can choose any depth for it. ∞ is the most permissive depth, as will become clear from the typing rules shortly.

$$\frac{\Gamma(x) = \tau \qquad 0 \leq \rho(x) \text{ or equiv. } \rho(x) \neq \bot}{\Delta; \Gamma; \rho \vdash_Z x : \tau} \textbf{ var}$$

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

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

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

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

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

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

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

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

$(\tau_1; q \xrightarrow{\rho; Z} \tau_2) \setminus x == (\tau_1 \setminus x); q \xrightarrow{(\rho \setminus x); Z} (\tau_2 \setminus x)$
$\tau \setminus x$ means removing x in all the ρ of arrow types containing in the type τ.

Figure 3: Typing rules, part 1

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

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

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

$$\frac{\Delta;\Gamma;\rho \vdash_Z e : \Box\,((\tau_1; 0 \xrightarrow{\rho'';0} \tau_2)) \qquad Z' = 1 + Z \qquad \rho' = 1 + \max(\rho, \rho'' + Z)}{\Delta;\Gamma;\rho' \vdash_{Z'} \delta(e) : \texttt{real}}\ \delta$$

$$\frac{\Delta;\Gamma;\rho \vdash_Z e : \tau \qquad \forall x \in \texttt{dom}(\Gamma), \Gamma(x) <: \Box\,(\Gamma(x)) \qquad \delta \notin e \qquad \texttt{dom}(\Gamma') = \texttt{dom}(\rho')}{\Delta;\Gamma,\Gamma';\rho,\boxed{\rho'} \vdash_Z e : \Box\,(\tau)}\ \textbf{box}$$

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

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

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

$$\frac{\Delta;\Gamma;\rho \vdash_Z e : \forall i \overset{\rho_1;Z_1}{::} S.\,\tau \qquad \Delta \vdash I :: S \qquad \rho' = \max(\rho, Z + \rho_1) \qquad Z' = Z_1[I/i] + Z}{\Delta;\Gamma;\rho' \vdash_{Z'} e\ [] : \tau[I/i]}\ \textbf{iapp}\textit{domain?}$$

where:

$$q + \rho \doteq \rho + q \doteq \lambda x.\ (q + \rho(x))$$

$$\rho_1 + \rho_2 \doteq \lambda x.\ (\rho_1(x) + \rho_2(x))$$

$$\max(\rho_1, \rho_2) \doteq \lambda x.\ \max(\rho_1(x), \rho_2(x))$$

Figure 4: Typing rules, part 2

$\boxed{\Delta; \rho \vdash \tau\ \texttt{wf}}$

$$\frac{\Delta; \rho \vdash \tau\ \texttt{wf}}{\Delta; \rho \vdash \texttt{list}\,\tau\ \texttt{wf}} \qquad \frac{}{\Delta; \rho \vdash \texttt{bool}\ \texttt{wf}} \qquad \frac{\Delta; \rho \vdash \tau_1\ \texttt{wf} \qquad \Delta; \rho \vdash \tau_2\ \texttt{wf}}{\Delta; \rho \vdash \tau_1 \times \tau_2\ \texttt{wf}}$$

$$\frac{\Delta; \rho \vdash \tau_1\ \texttt{wf} \qquad \Delta; \rho' \vdash \tau_2\ \texttt{wf} \qquad \Delta \vdash Z :: S \qquad \Delta \vdash q :: S}{\Delta; \rho \vdash \tau_1; q \xrightarrow{\rho'; Z} \tau_2\ \texttt{wf}}$$

$$\frac{\Delta; \rho \vdash \tau\ \texttt{wf}}{\Delta; \rho \vdash \square\,(\tau)\ \texttt{wf}} \qquad \frac{}{\Delta; \rho \vdash \texttt{real}\ \texttt{wf}} \qquad \frac{}{\Delta; \rho \vdash \texttt{int}\ \texttt{wf}} \qquad \frac{\Delta \vdash I :: S}{\Delta; \rho \vdash \texttt{int}[I]\ \texttt{wf}}$$

$$\frac{i :: S, \Delta; \rho \vdash \tau\ \texttt{wf} \qquad i :: S, \Delta \vdash Z :: S}{\Delta; \rho \vdash \forall i \overset{\rho; Z}{::} S.\, \tau\ \texttt{wf}}$$

Figure 5: Well-formedness rules

$\boxed{\Delta \vdash I :: S}$

$$\frac{\Delta(i) = S}{\Delta \vdash i :: S} \qquad \frac{}{\Delta \vdash n :: \mathbb{N}}$$

$$\frac{\Delta \vdash I_1 :: S \qquad \Delta \vdash I_2 :: S \qquad \Delta \vdash \diamond \in \{+, -, \max\}}{\Delta \vdash I_1 \diamond I_2 :: S}$$

Figure 6: Sorting rules

The rules follow *exactly* the definitions of $\texttt{adap}()$ and $\texttt{depth}_x()$ to compute Z and ρ in the conclusion of the typing rule for every construct. For instance, the typing rule for $e_1\ e_2$ does exactly what $\texttt{adap}()$ and $\texttt{depth}_x()$ do for the trace $T_1\ T_2 \triangleright \texttt{fix}\, f(x).T_3$.

Remark The astute reader might note that our type system looks very much like a coeffect+effect system where the coeffect ρ estimates the depth of each free variable and the effect Z estimates the adaptivity. While this is true at a high-level, note there are two essential differences between our type system and a coeffect+effect system.

- In some of our rules (those for $e_1\ e_2$ and $\texttt{if}(e_1, e_2, e_3)$), the conclusion's coeffect depends on the effects of the premises. For instance, in the rule for $e_1\ e_2$, the final coeffect ρ' depends on the effect Z_1. This does not happen in standard coeffect+effect systems.

$$\frac{}{\Delta \models \mathtt{b} <: \Box(\mathtt{b})}\ \textbf{sb-box-base} \qquad \frac{}{\Delta \models \mathtt{bool} <: \Box(\mathtt{bool})}\ \textbf{sb-box-bool}$$

$$\frac{\Delta \models \tau_1 <: \tau_1' \qquad \Delta \models \tau_2 <: \tau_2'}{\Delta \models \tau_1 \times \tau_2 <: \tau_1' \times \tau_2'}\ \textbf{sb-pair}$$

$$\frac{}{\Delta \models \Box((\tau_1; q \xrightarrow{\rho'; Z} \tau_2)) <: (\Box(\tau_1)); 0 \xrightarrow{\rho'; 0} (\Box(\tau_2))}\ \textbf{sb-box-arrow}$$

$$\frac{}{\Delta \models \mathtt{int}[I] <: \mathtt{int}}\ \textbf{sb-intI}$$

$$\frac{\Delta \models \tau_1' <: \tau_1 \qquad \Delta \models \tau_2 <: \tau_2' \qquad \rho \leq \rho' \qquad Z \leq Z' \qquad q \leq q'}{\Delta \models \tau_1; q \xrightarrow{\rho; Z} \tau_2 <: \tau_1'; q' \xrightarrow{\rho'; Z'} \tau_2'}\ \textbf{sb-arrow}$$

$$\frac{}{\Delta \models \Box(\tau) <: \tau}\ \textbf{sb-T} \qquad \frac{}{\Delta \models \Box(\tau) <: \Box(\Box(\tau))}\ \textbf{sb-D}$$

$$\frac{\Delta \models \tau_1 <: \tau_2}{\Delta \models \Box(\tau_1) <: \Box(\tau_2)}\ \textbf{sb-box} \qquad \frac{}{\Delta \models \tau <: \tau}\ \textbf{sb-refl}$$

$$\frac{\Delta \models \tau_1 <: \tau_2 \qquad \Delta \models \tau_2 <: \tau_3}{\Delta \models \tau_1 <: \tau_3}\ \textbf{sb-tran}$$

$$\frac{\Delta \models \tau_1 <: \tau_2}{\Delta \models \mathtt{list}\,\tau_1 <: \mathtt{list}\,\tau_2}\ \textbf{sb-list}$$

$$\frac{}{\Delta \models \mathtt{list}\,\Box(\tau) <: \Box(\mathtt{list}\,\tau)}\ \textbf{sb-list-box}$$

$$\frac{i :: S, \Delta \models \tau <: \tau' \qquad Z \leq Z' \qquad \rho \leq \rho'}{\Delta \models \forall i \overset{\rho; Z}{::} S.\tau <: \forall i \overset{\rho'; Z'}{::} S.\tau'}\ \textbf{sb-}\forall$$

Figure 7: Subtyping rules

- Our type for functions internalizes the effect. In a standard coeffect + effect system, the effect is always on the typing judgment, and at the point of function construction (`fix`), one anticipatively adds the effects of future function applications to the effect in the conclusion.

It may be possible to do away with both these differences by tracking more information in the coeffects (e.g., what is the maximum adaptivity of a function's argument?), but the details need to be worked out and verified.

5 Algorithmic rules

$$\frac{\Gamma(x) = \tau \qquad \rho = [x : 0] \cup \bot}{\Delta; \Gamma; \rho \vdash_Z x \uparrow \tau \Rightarrow \top} \textbf{ alg-var-}\uparrow$$

$$\frac{\rho = \bot}{\Delta; \Gamma; \rho \vdash_0 c \uparrow \mathtt{b} \Rightarrow \top} \textbf{ alg-const-}\uparrow$$

$$\frac{\rho = \bot}{\Delta; \Gamma; \rho \vdash_0 \mathtt{true} \uparrow \mathtt{bool} \Rightarrow \top} \textbf{ alg-true-}\uparrow$$

$$\frac{\rho = \bot}{\Delta; \Gamma; \rho \vdash_0 \mathtt{false} \uparrow \mathtt{bool} \Rightarrow \top} \textbf{ alg-false-}\uparrow$$

$$\frac{\Delta; \Gamma; \rho \vdash_Z e \uparrow \tau_1 \times \tau_2 \Rightarrow \Phi}{\Delta; \Gamma; \rho \vdash_Z \mathtt{fst}(e) \uparrow \tau_1 \Rightarrow \Phi} \textbf{ alg-fst-}\uparrow$$

$$\frac{\Delta; \Gamma; \rho \vdash_Z e \uparrow \tau_1 \times \tau_2 \Rightarrow \Phi}{\Delta; \Gamma; \rho \vdash_Z \mathtt{snd}(e) \uparrow \tau_2 \Rightarrow \Phi} \textbf{ alg-snd-}\uparrow$$

$$\frac{\begin{array}{c}\Delta; \Gamma; \rho \vdash_Z e \uparrow \square\,(\tau_1; 0 \xrightarrow{\rho''; 0} \tau_2) \Rightarrow \Phi \\ Z' = Z + 1 \qquad \rho' = 1 + \max(\rho, \rho'' + Z)\end{array}}{\Delta; \Gamma; \rho' \vdash_{Z'} \delta e \uparrow \mathtt{real} \Rightarrow \Phi} \textbf{ alg-}\delta\textbf{-}\uparrow$$

$$\frac{\begin{array}{c}\Delta; \Gamma; \rho_1 \vdash_{Z_1} e_1 \uparrow \tau_1; q \xrightarrow{\boxed{\rho}; Z} \tau_2 \Rightarrow \Phi_1 \qquad \Delta; \Gamma; \rho_2 \vdash_{Z_2} e_2 \downarrow \tau_1 \Rightarrow \Phi_2 \\ Z' = Z_1 + \max(Z, Z_2 + q) \qquad \rho' = \max(\rho_1, Z_1 + \max(\rho, \rho_2 + q))\end{array}}{\Delta; \Gamma; \rho' \vdash_{Z'} e_1\ e_2 \uparrow \tau_2 \Rightarrow \Phi_1 \wedge \Phi_2} \textbf{ alg-app-}\uparrow$$

the domain of the boxed?

Figure 8: Algorithmic Typing Rules

$$\frac{\Delta;\Gamma, f:\tau_1; q \xrightarrow{\rho;Z} \tau_2, x:\tau; \rho'' \vdash_Z e \downarrow \tau_2 \Rightarrow \Phi}{\Delta;\Gamma;\bot \vdash_0 \mathtt{fix}\, f(x:\tau).e \downarrow \tau_1; q \xrightarrow{\rho;Z} \tau_2 \Rightarrow \Phi \wedge \rho''(x) \leq q \wedge \forall y.y \in \mathtt{dom}(\rho).\rho''(y) \leq \rho(y)}\ \textbf{alg-fix-}\downarrow$$

$$\frac{\begin{array}{c}\Delta;\Gamma;\rho_1 \vdash_{Z_1} e_1 \downarrow \tau_1 \Rightarrow \Phi_1 \\ \Delta;\Gamma;\rho_2 \vdash_{Z_2} e_2 \downarrow \tau_2 \Rightarrow \Phi_2 \qquad Z = \max(Z_1, Z_2) \qquad \rho = \max(\rho_1, \rho_2)\end{array}}{\Delta;\Gamma;\rho \vdash_Z (e_1, e_2) \downarrow \tau_1 \times \tau_2 \Rightarrow \Phi_1 \wedge \Phi_2}\ \textbf{alg-pair-}\downarrow$$

$$\frac{\begin{array}{c}\Delta;\Gamma;\rho_1 \vdash_{Z_1} e_1 \uparrow \mathtt{bool} \Rightarrow \Phi_1 \qquad \Delta;\Gamma;\rho_2 \vdash_{Z_2} e_2 \downarrow \tau \Rightarrow \Phi_2 \\ \Delta;\Gamma;\rho_3 \vdash_{Z_3} e_3 \downarrow \tau \Rightarrow \Phi_3 \\ Z' = Z_1 + \max(Z_2, Z_3) \qquad \rho' = \max(\rho_1, Z_1 + \max(\rho_2, \rho_3))\end{array}}{\Delta;\Gamma;\rho' \vdash_{Z'} \mathtt{if}(e_1, e_2, e_2) \downarrow \tau \Rightarrow \Phi_1 \wedge \Phi_2 \wedge \Phi_3}\ \textbf{alg-if-}\downarrow$$

$$\frac{\Delta \vdash \tau : wf \qquad \rho = \bot}{\Delta;\Gamma;\rho \vdash_0 \mathtt{nil} \downarrow \mathtt{list}\,\tau \Rightarrow \top.}\ \textbf{alg-nil-}\downarrow$$

$$\frac{\begin{array}{c}\Delta;\Box(\Gamma);\rho \vdash_Z e \downarrow \tau \Rightarrow \Phi \\ \delta \notin e \qquad \forall x \in \mathtt{dom}(\Gamma').\rho'(x) = \bot \qquad \mathtt{dom}(\rho') = \mathtt{dom}(\Gamma')\end{array}}{\Delta;\Gamma', \Box(\Gamma);\rho, \boxed{\rho'} \vdash_Z \mathsf{BOX}\, e \downarrow \Box(\tau) \Rightarrow \Phi}\ \textbf{alg-box-}\downarrow$$

$$\frac{\begin{array}{c}\Delta;\Gamma;\rho_1 \vdash_{Z_1} e_1 \downarrow \tau \Rightarrow \Phi_1 \\ \Delta;\Gamma;\rho_2 \vdash_{Z_2} e_2 \downarrow \mathtt{list}\,\tau \Rightarrow \Phi_2 \qquad Z = \max(Z_1, Z_2) \qquad \rho = \max(\rho_1, \rho_2)\end{array}}{\Delta;\Gamma;\rho \vdash_Z \mathtt{cons}(e_1, e_2) \downarrow \mathtt{list}\,\tau \Rightarrow \Phi_1 \wedge \Phi_2}\ \textbf{alg-cons-}\downarrow$$

$$\frac{\begin{array}{c}\Delta;\Gamma;\rho_1 \vdash_{Z_1} e_1 \uparrow \tau_1 \Rightarrow \Phi \qquad \Delta;\Gamma, x:\tau_1;\rho_2 \vdash_{Z_2} e_2 \downarrow \tau \Rightarrow \Phi_2 \\ Z = \max(Z_2, Z_1 + q) \qquad \rho = \max(\rho_2/x, \rho_1 + q)\end{array}}{\Delta;\Gamma;\rho \vdash_Z \mathtt{let}\, x : q = e_1\, \mathtt{in}\, e_2 \downarrow \tau \Rightarrow \Phi_1 \wedge \Phi_2 \wedge \rho_2(x) \leq q}\ \textbf{alg-let-}\downarrow$$

$$\frac{\Delta;\Gamma;\rho \vdash_Z e \uparrow \tau \Rightarrow \Phi \qquad \Delta \models \tau \equiv \tau' \Rightarrow \Phi'}{\Delta;\Gamma;\rho \vdash_Z e \downarrow \tau' \Rightarrow \Phi \wedge \Phi'}\ \textbf{alg-}\uparrow\textbf{-}\downarrow$$

$$\frac{\Delta;\Gamma;\rho \vdash_Z e \downarrow \tau \Rightarrow \Phi \qquad \Delta \vdash \tau\ \mathtt{wf} \qquad \mathsf{FIV}(\tau, \rho, Z) \in \Delta}{\Delta;\Gamma;\rho \vdash_Z (e : \tau, \rho, Z) \uparrow \tau \Rightarrow \Phi}\ \textbf{alg-anno-}\uparrow$$

$$\frac{i :: S, \Delta;\Gamma;\rho \vdash_Z e \downarrow \tau \Rightarrow \Phi}{\Delta;\Gamma;\rho \vdash_0 \Lambda.e \downarrow \forall i \overset{\rho;Z}{::} S.\tau \Rightarrow \forall i.\Phi}\ \textbf{alg-iabs-}\downarrow$$

$$\frac{\begin{array}{c}\Delta;\Gamma;\rho \vdash_Z e \uparrow \forall i \overset{\rho_1;Z_1}{::} S.\tau \Rightarrow \Phi \qquad \Delta \vdash I :: S \\ \rho' = \max(\rho, Z + \rho_1) \qquad Z' = Z_1[I/i] + Z\end{array}}{\Delta;\Gamma;\rho' \vdash_{Z'} e[I] \uparrow \tau[I/i] \Rightarrow \Phi}\ \textbf{alg-iapp}\uparrow$$

the domain of the boxed and rule alg-iapp ?

Figure 9: Algorithmic Typing Rules, part 2

$$\frac{}{\Delta \models \mathtt{b} \equiv \mathtt{b} \Rightarrow \top}\ \textbf{alg-eq-base} \qquad \frac{}{\Delta \models \mathtt{bool} \equiv \mathtt{bool} \Rightarrow \top}\ \textbf{alg-eq-bool}$$

$$\frac{\Delta \models \tau_1 \equiv \tau_1' \Rightarrow \Phi_1 \qquad \Delta \models \tau_2 \equiv \tau_2' \Rightarrow \Phi_2}{\Delta \models \tau_1 \times \tau_2 \equiv \tau_1' \times \tau_2' \Rightarrow \Phi_1 \wedge \Phi_2}\ \textbf{alg-eq-pair}$$

$$\frac{\Delta \models \tau_1 \equiv \tau_1' \Rightarrow \Phi_1 \qquad \Delta \models \tau_2 \equiv \tau_2' \Rightarrow \Phi_2}{\Delta \models \tau_1; q \xrightarrow{\rho; Z} \tau_2 \equiv \tau_1'; q' \xrightarrow{\rho'; Z'} \tau_2' \Rightarrow \Phi_1 \wedge \Phi_2 \wedge q = q' \wedge Z = Z' \wedge \rho = \rho'}\ \textbf{alg-eq-arrow}$$

$$\frac{}{\Delta \models \mathtt{int}[I] \equiv \mathtt{int}[I'] \Rightarrow I = I'}\ \textbf{alg-eq-int}$$

$$\frac{\Delta \models \tau \equiv \tau' \Rightarrow \Phi}{\Delta \models \Box(\tau) \equiv \Box(\tau') \Rightarrow \Phi}\ \textbf{alg-eq-box}$$

$$\frac{\Delta \models \tau \equiv \tau' \Rightarrow \Phi}{\Delta \models \mathtt{list}\,\tau \equiv \mathtt{list}\,\tau' \Rightarrow \Phi}\ \textbf{alg-eq-list}$$

$$\frac{i, \Delta \models \tau \equiv \tau' \Rightarrow \Phi}{\Delta \models \forall i \overset{\rho; Z}{::} S.\tau \equiv \forall i \overset{\rho'; Z'}{::} S.\tau' \Rightarrow \forall I :: S.\Phi \wedge \rho = \rho' \wedge Z = Z'}\ \textbf{alg-eq-}\forall$$

Figure 10: Algorithmic equivalence Rules

$$
\begin{array}{lcl}
[\![\texttt{bool}]\!]_{\mathrm{V}} & = & \{\texttt{true}, \texttt{false}\} \\
[\![\texttt{b}]\!]_{\mathrm{V}} & = & \{c \mid c : \texttt{b}\} \\
[\![\tau_1 \times \tau_2]\!]_{\mathrm{V}} & = & \{(v_1, v_2) \mid v_1 \in [\![\tau_1]\!]_{\mathrm{V}} \wedge v_2 \in [\![\tau_2]\!]_{\mathrm{V}}\} \\
[\![\tau_1; q \xrightarrow{\rho; Z} \tau_2]\!]_{\mathrm{V}} & = & \{(\texttt{fix}\, f(x).e, \theta) \mid \forall v \in [\![\tau_1]\!]_{\mathrm{V}}. \\
 & & \qquad (\theta[x \mapsto v, f \mapsto (\texttt{fix}\, f(x).e, \theta)], e) \in [\![\tau_2]\!]_{\mathrm{E}}^{\rho[x:q, f:\infty], Z}\} \\
 & & \\
[\![\tau]\!]_{\mathrm{E}}^{\rho, Z} & = & \{(\theta, e) \mid \forall v\ T.\ (\theta, e \Downarrow v, T) \\
 & & \qquad \Rightarrow (\texttt{adap}(T) \leq Z \wedge \\
 & & \qquad\quad \forall x \in \text{Vars}.\ \texttt{depth}_x(T) \leq \rho(x) \wedge \\
 & & \qquad\quad v \in [\![\tau]\!]_{\mathrm{V}})\}
\end{array}
$$

Figure 11: Logical relation without step-indexing

6 Logical relation and soundness

Our type system is sound in the following sense.

Theorem 6 (Soundness)**.** If $\vdash_Z e : \tau$ and $\cdot, e \Downarrow v, T$ then $\texttt{adap}(T) \leq Z$.

To prove this theorem, we build a logical relation on types. As usual, the relation consists of a value relation and an expression relation. We first show a non-step-indexed version of the relation in Figure 11. This relation is well-founded but cannot, as usual, be used directly to prove the soundness of the `fix` typing rule. We show this relation here purely for exposition without the clutter of step-indices. We will step-index the relation shortly.

The value relation $[\![\tau]\!]_{\mathrm{V}}$, as usual, defines which *closed* values are semantically in the type τ. The only interesting case of this relation is the case for the arrow type. The expression relation $[\![\tau]\!]_{\mathrm{E}}^{\rho, Z}$ is more interesting. It is indexed by a depth map and an adaptivity. Importantly, this relation contains sets of configurations (θ, e) where the evaluation of (θ, e) produces a value in the value relation at τ, the adaptivity of the trace is no more than Z and for any variable x, $\texttt{depth}_x$ of the trace is no more than $\rho(x)$.

The value relation can be lifted to contexts in the usual way. Say that $\theta \in [\![\Gamma]\!]_{\mathrm{V}}$ when $\texttt{dom}(\theta) \supseteq \texttt{dom}(\Gamma)$ and for every $x \in \texttt{dom}(\Gamma)$, $\theta(x) \in [\![\Gamma(x)]\!]_{\mathrm{V}}$.

The fundamental theorem we would *like* to prove is the following. Of course, this cannot be proven until we step-index the relation, but this should give an idea of the connection between the type system and the logical relation.

Proposition 7 (Fundamental theorem)**.** If $\Gamma; \rho \vdash_Z e : \tau$ and $\theta \in [\![\Gamma]\!]_{\mathrm{V}}$, then $(\theta, e) \in [\![\tau]\!]_{\mathrm{E}}^{\rho, Z}$.

If this proposition were provable, then Theorem 6 would follow from it immediately.

$\boxed{|T| : \text{Traces} \to \mathbb{N}}$

$$
\begin{array}{lcl}
|x| & = & 0 \\
|T_1\ T_2 \triangleright \mathtt{fix}\, f(x).T_3| & = & |T_1| + |T_2| + |T_3| + 1 \\
|\,\mathtt{fix}\, f(x).e| & = & 0 \\
|(T_1, T_2)| & = & |T_1| + |T_2| \\
|\mathtt{fst}(T)| & = & |T| + 1 \\
|\mathtt{snd}(T)| & = & |T| + 1 \\
|\mathtt{true}| & = & 0 \\
|\mathtt{false}| & = & 0 \\
|\mathtt{if}^{\mathtt{t}}(T_b, T_t)| & = & |T_b| + |T_t| + 1 \\
|\mathtt{if}^{\mathtt{f}}(T_b, T_f)| & = & |T_b| + |T_f| + 1 \\
|c| & = & 0 \\
|\delta(T)| & = & |T| + 1 \\
|\mathtt{nil}| & = & 0 \\
|\,\mathtt{cons}(T_1, T_2)| & = & |T_1| + |T_2| \\
|\,\mathtt{let}(x, T_1, T_2)| & = & |T_1| + |T_2| \\
\boxed{|\mathtt{IApp}(T_1, T_2)|} & = & |T_1| + |T_2| \\
\boxed{|\Lambda.e|} & = & 0
\end{array}
$$

Figure 12: Size of a trace

Step-indexed logical relation To step-index the relation we need to have a notion of the number of reduction steps. One way to do this is to change the operational semantics to count the number of reduction steps. However, this is unnecessary since the number of reduction steps in a derivation is exactly the size of the derivation's trace. So, we just "index" the relation on the size of the derivation's trace. The same technique has been used previously [1].

Definition 8 (Trace size)**.** The size of a trace is defined as the number of $T_1\ T_2 \triangleright \mathtt{fix}\, f(x).T_3$, $\mathtt{if}^{\mathtt{t}}(T_b, T_t)$, $\mathtt{if}^{\mathtt{f}}(T_b, T_f)$ and $\delta(T)$ constructors in it. Basically, we count all elimination constructors, but not introduction constructors.[3]

Definition 9 (Trace size)**.** We use $|T|$ to denote the size of the trace T.

The step-indexed relation is shown in Figure 26. Now, the value relation $[\![\tau]\!]_{\mathrm{V}}$ contains pairs of step-indices k and closed values. The expression relation contains pairs of step-indices and configurations (θ, e). The relation

[3]Deepak's note: The purpose of counting this way is to make sure that the trace of the evaluation of any *value* has size 0. This may be required in the proof of the fundamental theorem, but I am not sure.

$$
\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}}. \\
 & & \qquad (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}\} \\
\boxed{[\![\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 \mathrm{Vars}.\, \texttt{depth}_x(T) \leq \rho(x) \wedge \\
 & & \qquad\quad ((k - j, v) \in [\![\tau]\!]_{\mathrm{V}})\} \\
\boxed{[\![\texttt{int}]\!]_{\mathrm{V}}} & = & \{(k, i) \mid k \in \mathbb{N} \wedge i : \texttt{int}\} \\
\boxed{[\![\texttt{real}]\!]_{\mathrm{V}}} & = & \{(k, r) \mid k \in \mathbb{N} \wedge r : \texttt{real}\} \\
\boxed{[\![\Box\, (\tau)]\!]_{\mathrm{V}}} & = & \{(k, v) \mid k \in \mathbb{N} \wedge (k, v) \in [\![\tau]\!]_{\mathrm{V}} \wedge \delta \notin v\} \\
\boxed{[\![\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]}\} \\
\boxed{[\![\texttt{int}[I]]\!]_{\mathrm{V}}} & = & \{(k, n) \mid k \in \mathbb{N} \wedge n = I\}
\end{array}
$$

Figure 13: Logical relation with step-indexing

basically mirrors the non-step-indexed relation of Figure 11, with step indexes added in the completely standard way.

We say that $(k, \theta) \in [\![\Gamma]\!]_{\mathrm{V}}$ when $\texttt{dom}(\theta) \supseteq \texttt{dom}(\Gamma)$ and for every $x \in \texttt{dom}(\Gamma)$, $(k, \theta(x)) \in [\![\Gamma(x)]\!]_{\mathrm{V}}$.

We say that $\sigma \in [\![\Delta]\!]_{\mathrm{V}}$ when $\texttt{dom}(\sigma) \supseteq \texttt{dom}(\Delta)$ and for every $x \in \texttt{dom}(\Delta)$, $\cdot \vdash \sigma(x) :: \Delta(x)$

Lemma 10. if $\texttt{adap}(T) = 0$ then for all variable x, $\texttt{depth}_x(T) \leq 0$. ($\perp$ is less than any number in $\mathbb{N}$).

Proof. It is proved simply by induction on (T). □

Lemma 11. if $\theta, e \Downarrow v, T$ and $\delta \notin e$ and $\delta \notin \theta$, then $\texttt{adap}(T) = 0$ and $\delta \notin v$

Proof. By induction on the derivation of opertational semantics.
Case

$$\frac{}{\theta, x \Downarrow \theta(x), (x, \theta)}$$

We know $\delta \notin \theta \wedge \delta \notin x$,
so we know : $T = (x, \theta) \Rightarrow \texttt{adap}(T) = 0$ and $\delta \notin \theta(x)$.
Case

$$\frac{\theta, e_1 \Downarrow v_1, T_1 \qquad v_1 = (\texttt{fix}\, f(x).e, \theta') \qquad \theta, e_2 \Downarrow v_2, T_2 \qquad \theta'[f \mapsto v_1, x \mapsto v_2], e \Downarrow v, T}{\theta, e_1\ e_2 \Downarrow v, T_1\ T_2 \triangleright \texttt{fix}\, f(x).T}$$

We know that: $\delta \notin (e_1\ e_2)$ (1) $\wedge\ \delta \notin \theta$.
From (1), we get : $\delta \notin e_1$ (2) and $\delta \notin e_2$ (3).
IH on the first premise instantiated with (2), we know: $\texttt{adap}(T_1) = 0 \wedge \delta \notin v_1$ (4).
Because v_1 is a function, we know that $\delta \notin (\texttt{fix}\, f(x).e, \theta')$ (5) from the second premise.
IH on the third premise instantiated with (3), we know: $\texttt{adap}(T_2) = 0 \wedge \delta \notin v_2$ (6).

From (4), (5), (6), we know that: $\delta \notin \theta'[f \mapsto v_1, x \mapsto v_2]$ and $\delta \notin e$ (7).

By IH on the fourth premise instantiated with (7), we conclude that : $\delta \notin v$ and $\texttt{adap}(T) = 0$.

We want to show:
$\texttt{adap}(T_1\ T_2 \triangleright \texttt{fix}\, f(x).T_3) = \texttt{adap}(T_1) + \max(\texttt{adap}(T_3), \texttt{adap}(T_2) + \texttt{depth}_x(T_3)) = 0$.
We already know that $\texttt{adap}(T_1) = 0 \wedge \texttt{adap}(T_2) = 0 \wedge \texttt{adap}(T_1) = 0$.
By Lemma 10, we know: $\texttt{depth}_x(T_3) \leq 0$.
So we conclude that $\texttt{adap}(T_1\ T_2 \triangleright \texttt{fix}\, f(x).T_3) = 0$.
Case

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

We assume that $\delta \notin \theta$ and $\delta \notin \texttt{fix}\, f(x).e$.
STS1: $\texttt{adap}(T) = \texttt{adap}((\texttt{fix}\, f(x).e, \theta)) = 0$, which is proved by the defintion of adaptivity.

STS2: $\delta \notin (\texttt{fix}\, f(x).e, \theta)$, which is known by our assumption.
Case

$$\frac{\theta, e_1 \Downarrow v_1, T_1 \qquad \theta, e_2 \Downarrow v_2, T_2}{\theta, (e_1, e_2) \Downarrow (v_1, v_2), (T_1, T_2)}$$

We assume that $\delta \notin \theta$ and $\delta \notin (e_1, e_2)$ (1).
From (1), we can infer that : $\delta \notin e_1$ (2) and $\delta \notin e_2$ (3).
By IH on the first premise instantiated with (2), we know: $\texttt{adap}(T_1) = 0$ (4) and $\delta \notin v_1$ (5).
By IH on the first premise instantiated with (3), we know: $\texttt{adap}(T_2) = 0$ (6) and $\delta \notin v_2$ (7).
STS1: $\texttt{adap}((T_1, T_2)) = 0$, which is proved by (4), (6).
STS2: $\delta \notin (v_1, v_2)$, which is proved by (5), (7).

Case

$$\frac{\theta, e \Downarrow (v_1, v_2), T}{\theta, \texttt{fst}(e) \Downarrow v_1, \texttt{fst}(T)}$$

We assume that $\delta \notin \theta$ and $\delta \notin \texttt{fst}(e)$ (1).
From (1), we can infer that : $\delta \notin e$ (2)
By IH on the first premise instantiated with (2), we know: $\texttt{adap}(T) = 0$ (3) and $\delta \notin (v_1, v_2)$ (4).
STS1: $\texttt{adap}(\texttt{fst}(T)) = 0$, which is proved by (3).
STS2: $\delta \notin v_1$, which is proved by (4).

Case

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

We assume that $\delta \notin \theta$ and $\delta \notin \texttt{let}\, x; q = e_1\, \texttt{in}\, e_2$ (1).
From (1), we can infer that : $\delta \notin e_1$ (2) and $\delta \notin e_2$ (3).
By IH on the first premise instantiated with (2), we know: $\texttt{adap}(T_1) = 0$ (4) and $\delta \notin v_1$ (5).
From (5), We know that $\delta \notin \theta[x \mapsto v_1]$ (6).
By IH on the second premise instantiated with (3), (6) , we know: $\texttt{adap}(T_2) = 0$ (7) and $\delta \notin v$ (8).
STS1: $\texttt{adap}(\texttt{let}(x, T_1, T_2)) = 0$, which is proved by (4), (7) and Lemma 10.
STS2: $\delta \notin v_1$, which is proved by (8).

□

Lemma 12. Sutyping Soundness
1. If $\Delta \models \tau <: \tau'$ and $\sigma \in [\![\Delta]\!]_{\mathrm{V}}$ and $(k, v) \in [\![\sigma\tau]\!]_{\mathrm{V}}$, then $(k, v) \in [\![\sigma\tau']\!]_{\mathrm{V}}$.
2. If $\Delta \models \tau <: \tau'$ and $\sigma \in [\![\Delta]\!]_{\mathrm{V}}$ and $(k, (\theta, e)) \in [\![\sigma\tau]\!]_{\mathrm{E}}^{\rho, Z}$, and $\rho < \rho'$ and $Z < Z'$, then $(k, (\theta, e)) \in [\![\sigma\tau']\!]_{\mathrm{E}}^{\rho', Z'}$.

Proof. The proof of statement (1) is by induction on the subtying derivation.

Case

$$\frac{\Delta \models \tau_1 <: \tau_1' \qquad \Delta \models \tau_2 <: \tau_2'}{\Delta \models \tau_1 \times \tau_2 <: \tau_1' \times \tau_2'}$$

Suppose we know : $(k, (v_1, v_2)) \in [\![\sigma\tau_1 \times \tau_2]\!]_{\mathrm{V}}$.
TS: $(k, (v_1, v_2)) \in [\![\sigma\tau_1' \times \tau_2']\!]_{\mathrm{V}}$
Unfold its definition, we know: $(k, v_1) \in [\![\sigma\tau_1]\!]_{\mathrm{V}}$ and $(k, v_2) \in [\![\sigma\tau_2]\!]_{\mathrm{V}}$.
By IH on the first premise, we know: $(k, v_1) \in [\![\sigma\tau_1']\!]_{\mathrm{V}}$.
By IH on the second premise, we know: $(k, v_2) \in [\![\sigma\tau_2']\!]_{\mathrm{V}}$.
This case is proved.

Case

$$\frac{\Delta \models \tau_1' <: \tau_1 \qquad \Delta \models \tau_2 <: \tau_2' \qquad \rho \leq \rho' \qquad Z \leq Z' \qquad q \leq q'}{\Delta \models \tau_1; q \xrightarrow{\rho; Z} \tau_2 <: \tau_1'; q' \xrightarrow{\rho'; Z'} \tau_2'}$$

Suppose $(k, (\mathtt{fix}\, f(x).e, \theta)) \in [\![\sigma(\tau_1; q \xrightarrow{\rho; Z} \tau_2)]\!]_{\mathrm{V}}$ $(\star)$.
TS: $(k, (\mathtt{fix}\, f(x).e, \theta)) \in [\![\sigma(\tau_1'; q' \xrightarrow{\rho'; Z'} \tau_2')]\!]_{\mathrm{V}}$
From $(\star)$, we know:
$\forall j < k.\ \forall (j, v) \in [\![\tau_1]\!]_{\mathrm{V}}$.
$(j, (\theta[x \mapsto v, f \mapsto (\mathtt{fix}\, f(x).e, \theta)], e)) \in [\![\tau_2]\!]_{\mathrm{E}}^{\rho[x:q, f:\infty], Z}$
STS: $\forall j < k.\ \forall (j, v') \in [\![\tau_1']\!]_{\mathrm{V}}$.
$(j, (\theta[x \mapsto v', f \mapsto (\mathtt{fix}\, f(x).e, \theta)], e)) \in [\![\tau_2']\!]_{\mathrm{E}}^{\rho'[x:q, f:\infty], Z'}$ $(\diamond)$
Pick j, assume $(j, v') \in [\![\tau_1']\!]_{\mathrm{V}}$, we also know $(j, v') \in [\![\tau_1]\!]_{\mathrm{V}}$ by IH on the first premise.
From $(\star)$, we know:
$(j, (\theta[x \mapsto v', f \mapsto (\mathtt{fix}\, f(x).e, \theta)], e)) \in [\![\tau_2]\!]_{\mathrm{E}}^{\rho[x:q, f:\infty], Z}$.
Unfold its definition, we get:
$\forall v_t\ T\ j.\ \theta[x \mapsto v', f \mapsto (\mathtt{fix}\, f(x).e, \theta)], e \Downarrow v_t, T \wedge (|T| = j) \wedge (j \leq k)$
$\Rightarrow (\mathtt{adap}(T) \leq Z \wedge \forall x \in \text{Vars}.\ \mathtt{depth}_x(T) \leq \rho[x : q, f : \infty](x) \wedge ((k-j, v_t) \in [\![\tau_2]\!]_{\mathrm{V}})$
Unfold $(\diamond)$, STS:

1. $\mathtt{adap}(T) \leq Z'$, It is proved because we know $\mathtt{adap}(T) \leq Z$ and $Z \leq Z'$.

2. $\forall x \in \text{Vars}.\ \mathtt{depth}_x(T) \leq \rho'[x : q', f : \infty](x)$, because $\rho \leq \rho'$, It is proved using $\forall x \in \text{Vars}.\ \mathtt{depth}_x(T) \leq \rho[x : q, f : \infty](x)$.

3. $((k - j, v_t) \in [\![\tau_2']\!]_{\mathrm{V}})$ it is proved by IH on the second premise with $((k - j, v_t) \in [\![\tau_2]\!]_{\mathrm{V}})$.

Case

$$\frac{}{\Delta \models \Box\left((\tau_1; q \xrightarrow{\rho'; Z} \tau_2)\right) <: \left(\Box\left(\tau_1\right)\right); 0 \xrightarrow{\rho'; 0} \left(\Box\left(\tau_2\right)\right)}$$

Assume $\sigma \in [\![\Delta]\!]_{\mathrm{V}}$.
We have:

$$(k, (\mathtt{fix}\, f(x).e, \theta)) \in [\![\sigma\Box\left((\tau_1; q \xrightarrow{\rho'; Z} \tau_2)\right)]\!]_{\mathrm{V}}\ (\star)$$

TS: $(k, (\mathtt{fix}\, f(x).e, \theta)) \in [\![\sigma(\Box\left(\tau_1\right)); 0 \xrightarrow{\rho'; 0} \left(\Box\left(\tau_2\right)\right)]\!]_{\mathrm{V}}$
Assume $j < k \wedge (j, v) \in [\![\Box\left(\tau_1\right)]\!]_{\mathrm{V}}$, which implies $(j, v) \in [\![\tau_1]\!]_{\mathrm{V}} \wedge \delta \notin v$
STS: $(j, (\theta[x \mapsto v, f \mapsto (\mathtt{fix}\, f(x).e, \theta)], e)) \in [\![\Box\left(\tau_2\right)]\!]_{\mathrm{E}}^{\rho[x:0, f:\infty], 0}(\star\star)$

By unrolling $(\star)$, we know that:

$$(k, (\mathtt{fix}\, f(x).e, \theta)) \in [\![\sigma(\tau_1; q \xrightarrow{\rho'; Z} \tau_2)]\!]_{\mathrm{V}}\ (a) \wedge \delta \notin (\mathtt{fix}\, f(x).e, \theta)\ (b)$$

By unrolling (a) with $k < j \wedge (j, v) \in [\![\tau_1]\!]_{\mathrm{V}}$, we get:

$$(j, (\theta[x \mapsto v, f \mapsto (\mathtt{fix}\, f(x).e, \theta)], e)) \in [\![\tau_2]\!]_{\mathrm{E}}^{\rho'[x:q, f:\infty], Z}(\diamond)$$

Unrolling $(\diamond)$, assume

$$\theta[x \mapsto v, f \mapsto (\mathtt{fix}\, f(x).e, \theta)], e \Downarrow v_1, T_1 \wedge |T_1| = j_1 \wedge j_1 \leq k$$

. We get :

$$\mathtt{adap}(T_1) \leq Z\ (c)$$

$$\forall y \in \mathrm{Vars}.\ \mathtt{depth}_y(T_1) \leq \rho'[x : q, f : \infty](x)\ (d)$$

$$(k - j_1, v_1) \in [\![\tau_2]\!]_{\mathrm{V}}\ (e)$$

Unrolling $(\star\star)$ with our assumption $\theta[x \mapsto v, f \mapsto (\mathtt{fix}\, f(x).e, \theta)], e \Downarrow v_1, T_1 \wedge |T_1| = j_1 \wedge j_1 \leq k$.
We know that $\delta \notin v \wedge \delta \notin (\mathtt{fix}\, f(x).e, \theta) \wedge \delta \notin e$, by Lemma 11, we conclude that: $\mathtt{adap}(T_1) = 0$ (1) and $\delta \notin v_1$ (2). STS:

1. $\mathtt{adap}(T_1) \leq 0$, proved by (1)

2. $\forall y \in \mathrm{Vars}.\ \mathtt{depth}_y(T_1) \leq \rho'[x : 0, f : \infty](x)$. By Lemma 10 and (1), we can show that $\mathtt{depth}_x(T_1) = 0 leq \rho'[x : 0, f : \infty](x)$ when $y = x$. When $y \neq x$, it is proved by (d)

3. $(k - j_1, v_1) \in [\![\Box\left(\tau_2\right)]\!]_{\mathrm{V}}$
   It is proved by (e) and (2).

Case

$$\frac{\Delta \models \tau_1 <: \tau_2}{\Delta \models \mathtt{list}\,\tau_1 <: \mathtt{list}\,\tau_2}\ \textsc{list}$$

Assume $\sigma \in [\![\Delta]\!]_{\mathrm{V}}$.
There are two subcases.

1. $(k, \mathtt{nil}) \in [\![\sigma(\mathtt{list}\,\tau_1)]\!]_{\mathrm{V}}$
   It is proved that $(k, \mathtt{nil}) \in [\![\sigma(\mathtt{list}\,\tau_2)]\!]_{\mathrm{V}}$ by the definition.

2. $(k, \mathtt{cons}(v_1, v_2)) \in [\![\sigma(\mathtt{list}\,\tau_1)]\!]_{\mathrm{V}}$
   We know that $(k, v_1) \in [\![\sigma(\tau_1)]\!]_{\mathrm{V}}$ $(\star)$ and $(k, v_2) \in [\![\sigma(\mathtt{list}\,\tau_1)]\!]_{\mathrm{V}}$ $(\diamond)$.
   By IH on $(\star)$, we know that $(k, v_1) \in [\![\sigma\tau_2]\!]_{\mathrm{V}}$.
   STS: $(k, v_2) \in [\![\sigma(\mathtt{list}\,\tau_2)]\!]_{\mathrm{V}}$, which is proved by IH on $(\diamond)$.

Case

$$\frac{}{\Delta \models \mathtt{list}\,\Box\,(\tau) <: \Box\,(\mathtt{list}\,\tau)}\ \textsc{listbox}$$

Assume $\sigma \in [\![\Delta]\!]_{\mathrm{V}}$.
There are two subcases.

1. $(k, \mathtt{nil}) \in [\![\sigma(\mathtt{list}\,\Box\,(\tau))]\!]_{\mathrm{V}}$
   It is proved that $(k, \mathtt{nil}) \in [\![\sigma(\Box\,(\mathtt{list}\,\tau))]\!]_{\mathrm{V}}$ because $\delta \notin \mathtt{nil}$ and $(k, \mathtt{nil}) \in [\![\sigma\mathtt{list}\,\tau]\!]_{\mathrm{V}}$.

2. $(k, \mathtt{cons}(v_1, v_2)) \in [\![\sigma(\mathtt{list}\,\Box\,(\tau_1))]\!]_{\mathrm{V}}$
   We know that $(k, v_1) \in [\![\Box\,(\sigma\tau)]\!]_{\mathrm{V}}$ $(\star)$ and $(k, v_2) \in [\![\mathtt{list}\,\Box\,(\sigma\tau)]\!]_{\mathrm{V}}$ $(\diamond)$.
   Unfold $(\star)$, we know : $\delta \notin v_1$ (a) and $(k, v_1) \in [\![\sigma\tau]\!]_{\mathrm{V}}$ (b).
   By IH on $(\diamond)$, we know that : $(k, v_2) \in [\![\Box\,(\mathtt{list}\,\sigma\tau)]\!]_{\mathrm{V}}$.
   Unfolt its defintion, we know that $\delta \notin v_2$ (c) and $(k, v_2) \in [\![\mathtt{list}\,\sigma\tau]\!]_{\mathrm{V}}$ (d)
   .
   STS1: $(k, \mathtt{cons}(v_1, v_2)) \in [\![\sigma(\mathtt{list}\,\tau)]\!]_{\mathrm{V}}$.
   It is proved by (b),(d).

   STS2: $\delta \notin \mathtt{cons}(v_1, v_2)$.
   It is proved by (a), (c).
   By IH on $(\star)$, we know that $(k, v_1) \in [\![\tau_2]\!]_{\mathrm{V}}$.
   STS: $(k, v_2) \in [\![\mathtt{list}\,\tau_2]\!]_{\mathrm{V}}$, which is proved by IH on $(\diamond)$.

The proof of statement (2)
Assume $(k, (\theta, e)) \in [\![\sigma\tau]\!]_{\mathrm{E}}^{\rho,Z}$ $(\star)$.

TS: $(k, (\theta, e)) \in [\![\sigma\tau']\!]_{\mathrm{E}}^{\rho', Z'}$.
Unfold its definition, STS: $\forall j, \theta, T.\theta, e \Downarrow v, T \wedge (|T| = j) \wedge (j \leq k)$.
$\Rightarrow \mathtt{adap}(T) \leq Z' \wedge$
$\forall x \in \text{Vars}. \mathtt{depth}_x(T) \leq \rho'(x) \wedge$
$(k - j, v) \in [\![\tau']\!]_{\mathrm{V}}$
Pick j,v, T, s.t. $\theta, e \Downarrow v, T \wedge (|T| = j) \wedge (j \leq k)$.
STS1: $\mathtt{adap}(T) \leq Z'$
STS2: $\forall x \in \text{Vars}. \mathtt{depth}_x(T) \leq \rho'(x)$
STS3: $(k - j, v) \in [\![\tau']\!]_{\mathrm{V}}$.

By inversion on the assumption $(\star)$, Pick j,v, T. we know from assumption: $\theta, e \Downarrow v, T \wedge (|T| = j) \wedge (j \leq k)$.
We get: $\mathtt{adap}(T) \leq Z$ (a)
$\forall x \in \text{Vars}. \mathtt{depth}_x(T) \leq \rho(x)$ (b)
$(k - j, v) \in [\![\tau]\!]_{\mathrm{V}}$ (c)

STS1: $\mathtt{adap}(T) \leq Z'$ is proved by (a) and $Z \leq Z'$.
STS2: $\forall x \in \text{Vars}. \mathtt{depth}_x(T) \leq \rho'(x)$. It is proved by (b) and $\rho \leq \rho'$.
STS3: $(k - j, v) \in [\![\tau']\!]_{\mathrm{V}}$. It is proved by IH1 instantiated with (c) and $\Delta \models \tau <: \tau'$.

□

Lemma 13. 1. If $(k, v) \in [\![\tau]\!]_{\mathrm{V}}$ and $k' \leq k$, then $(k', v) \in [\![\tau]\!]_{\mathrm{V}}$.
2. 1. If $(k, e) \in [\![\tau]\!]_{\mathrm{E}}^{Z,\rho}$ and $k' \leq k$, then $(k', e) \in [\![\tau]\!]_{\mathrm{E}}^{Z,\rho}$
3.If $(k, \theta) \in [\![\Gamma]\!]_{\mathrm{V}}$ and $k' \leq k$, then $(k', \theta) \in [\![\Gamma]\!]_{\mathrm{V}}$.

Proof. (1,2) simultaneously proved by induction on τ, 3 followed by (1, 2). □

Theorem 14 (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
$$\frac{\Delta; \Gamma; \rho \vdash_Z e : \tau \qquad \forall x \in \mathtt{dom}(\Gamma), \Gamma(x) <: \Box(\Gamma(x)) \qquad \delta \notin e}{\Delta; \Gamma, \Gamma'; \rho \vdash_Z e : \Box(\tau)}$$

Assume $\sigma \in [\![\Delta]\!]_{\mathrm{V}}$, $(k, \theta) \in [\![\sigma(\Gamma, \Gamma')]\!]_{\mathrm{V}}$, TS: $(k, (\theta, \sigma e)) \in [\![\sigma\Box(\tau)]\!]_{\mathrm{E}}^{\rho,\sigma Z}$.
By inversion, STS :$\forall v, T, j$, s.t. $(\theta, \sigma e) \Downarrow (v, T) \wedge j = |T| \wedge j \leq k$.
(1) $\mathtt{adap}(T) \leq \sigma Z$, (2) $\forall x, \mathtt{depth}_x(T) \leq \rho(x)$, (3) $(k - j, v) \in [\![\sigma\Box(\tau)]\!]_{\mathrm{V}}$.
By IH instantiated with $\theta_1 \in [\![\sigma(\Gamma]\!]_{\mathrm{V}}$, we get: $(k, (\theta_1, \sigma e)) \in [\![\sigma\tau]\!]_{\mathrm{E}}^{\rho,\sigma Z}$ $(\star)$.
By inversion, we get $\forall v', T', j'$ s.t. $(\theta_1, \sigma e) \Downarrow (v', T') \wedge |T'| = j' \wedge j' \leq k$,
(1)' $\mathtt{adap}(T') \leq \sigma Z$, (2)' $\forall x, \mathtt{depth}_x(T') \leq \rho(x)$, (3)' $(k - j', v') \in [\![\sigma\tau]\!]_{\mathrm{V}}$.
Pick $v = v', T = T', j = j'$, (1) (2) is proved by (1)', (2)'.
STS (3) $(k - j, v) \in [\![\sigma\Box(\tau)]\!]_{\mathrm{V}}$.
Unfold the interpretation of box, STS:

1. $(k - j, v) \in [\![\sigma\tau]\!]_{\mathrm{V}}(a)$. It is proved from the inversion of $(\star)$.

2. $\delta \notin v$ (b). We know $\delta \notin e$.
   From the second premise, we know that $\forall x \in dom(\theta_1).\theta_1(x) \in [\![\Box(\Gamma(x))]\!]_{\mathrm{V}}$ from Subtyping soundness Lemma 12, which implies $\delta \notin \theta'$.
   By lemma 11, we know that $\delta \notin v$.

Case
$$\frac{\Delta, i; \Gamma; \rho \vdash_Z e : \tau}{\Delta; \Gamma; \rho \vdash_Z \Lambda.e : \forall i \overset{\rho,Z}{::} S.\tau}$$

Assume $(\sigma \in [\![\Delta]\!]_{\mathrm{V}}$ $(\star)$, $(k, \theta) \in [\![\sigma\Gamma]\!]_{\mathrm{V}}$ $(\diamond)$, TS $(k, (\theta, \Lambda.\sigma e)) \in [\![\forall i \overset{\rho,Z}{::} S.\tau]\!]_{\mathrm{E}}^{\rho,\sigma Z}$.
Since $(\theta, \Lambda.\sigma e)$ is value, STS (a) $(k, (\theta, \Lambda.\sigma e)) \in [\![\forall i \overset{\rho,Z}{::} S.\tau]\!]_{\mathrm{V}}$.
Unfolding (a), pick any I s.t. $\vdash I :: S$, STS: (b) $(k, (\theta, \sigma e)) \in [\![\sigma\tau[I/i]]\!]_{\mathrm{E}}^{\rho,(\sigma Z)[I/i]}$.
By $(\star)$, we know $\sigma[i \to I] \in [\![\Delta, i :: S]\!]_{\mathrm{V}}$ $(\triangle)$.
(b) is proved by ih on premise and $(\diamond)$ and $(\triangle)$.

Case

$$\frac{\Delta;\Gamma;\rho \vdash_Z e : \forall i \overset{\rho_1;Z_1}{::} S.\tau \ (\star) \qquad \Delta \vdash I :: S \ (\diamond) \quad \rho' = \max(\rho, Z + \rho_1) \qquad Z' = Z_1[I/i] + Z}{\Delta;\Gamma;\rho' \vdash_{Z'} e\ [] : \tau[I/i]}$$

Assume $\sigma \in [\![\Delta]\!]_{\mathrm{V}}$, $(k,\theta) \in [\![\sigma\Gamma]\!]_{\mathrm{V}}$, TS: $(k,(\theta,\sigma(e[]))) \in [\![\sigma(\tau[I/i])]\!]_{\mathrm{E}}^{\rho,\sigma(Z[I/i]+Z_1)}$.
By inversion, STS: $\forall v,T,j$ s.t. $(\theta,e[]) \Downarrow (v,T) \wedge |T| = j \wedge j \leq T$.
(1) $\mathtt{adap}(T) \leq \sigma(Z[I/i] + Z_1)$.
(2) $\forall x, \mathtt{depth}_x(T) \leq \rho'(x)$.
(3) $(k-j,v) \in [\![\sigma(\tau[I/i])]\!]_{\mathrm{V}}$
By induction hypothesis on $(\star)$, we get: $(k,(\theta,\sigma e) \in [\![\sigma\forall i \overset{\rho_1;Z_1}{::} S.\tau]\!]_{\mathrm{E}}^{\rho,\sigma Z}$.
By inversion, assume $(a)(\theta,\sigma e) \Downarrow ((\Lambda.e_1,\theta')T_1) \wedge |(|T_1) = j_1 \wedge j_1 \leq k$. We know:
(1)' $\mathtt{adap}(T_1) \leq \sigma Z$.
(2)' $\forall x, \mathtt{depth}_x(T_1) \leq \rho(x)$.
(3)' $(k-j_1,(\Lambda.e,\theta')) \in [\![\sigma\forall i \overset{\rho_1;Z_1}{::} S.\tau]\!]_{\mathrm{V}}$.
Unfold (3)', assume $\vdash \sigma I :: S$ from $(\diamond)$, we get:
(4) $(k-j_1,(\theta',e_1) \in [\![\sigma(\tau[I/i])]\!]_{\mathrm{E}}^{\rho_1,\sigma Z[I/i]}$
Unfold (4), assume $(b)(\theta',e) \Downarrow (v_2,T_2) \wedge |(|T_2) = j_2 \wedge j_2 \leq k$, we get:
(1)" $\mathtt{adap}(T_2) \leq \sigma Z_1[I/i]$.
(2)" $\forall x, \mathtt{depth}_x(T_2) \leq \rho(x)$.
(3)" $(k-j_2,v_2) \in [\![\sigma\tau[I/i]]\!]_{\mathrm{V}}$.
Pick $v = v_2$, $T = \mathtt{IApp}(T_1,T_2)$, $j = j_1 + j_2$, By operational semantics of IApp, we know: $(\theta,\sigma e[]) \Downarrow (v,T) \wedge |(|T) = j \wedge j \leq k$ (1) (2) (3) are proved by (1)' (2)' (3)' and (1)" (2)" (3)".

Case

$$\frac{\Delta;\Gamma;\rho \vdash_Z e : \square\,((\tau_1;0 \xrightarrow{\rho'';0} \tau_2)) \ (\star) \quad Z' = 1 + Z \qquad \rho' = 1 + \max(\rho, \rho'' + Z)}{\Delta;\Gamma;\rho' \vdash_{Z'} \delta(e) : \mathtt{real}}\ \textbf{PRIMITIVE}$$

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. $(\forall x \in \mathrm{Vars}.\mathtt{depth}_x(T) \leq \rho'(x))$
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:

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

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

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 Lemma5 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 10, 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: $\mathtt{depth}_x(T) = \mathtt{depth}_x(\delta(T')) = 1+\max(\mathtt{depth}_x(T'), \mathtt{adap}(T')+\mathsf{MAX}_{v\in\tau_1}\Big(\max(\mathtt{depth}_x(T_3(v)), \bot)\Big) \leq \rho'(x)$.
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.

Case $\dfrac{\Delta; \Gamma, f : (\tau_1; q \xrightarrow{\rho;\, Z} \tau_2), x : \tau_1; \rho[f : \infty, x : q] \vdash_Z e : \tau_2\ (1)}{\Delta; \Gamma; \rho' \vdash_{Z'} \mathtt{fix}\, f(x).e : (\tau_1; q \xrightarrow{\rho;\, Z} \tau_2)}$ **FIX**

Assume $\sigma \in [\![\Delta]\!]_{\mathrm{V}}$, $(k, \theta) \in [\![\sigma\Gamma]\!]_{\mathrm{V}}$.
TS: $(k, (\theta, \sigma\,\mathtt{fix}\, f(x).e)) \in [\![\sigma(\tau_1; q \xrightarrow{\rho; Z} \tau_2)]\!]_{\mathrm{E}}^{\rho', \sigma Z'}$.
By inversion, STS: $\forall v, T, j\ (\theta, \sigma\,\mathtt{fix}\, f(x).e \Downarrow v, T) \wedge (|T| = j) \wedge (j \leq k)$
1. $(\mathtt{adap}(T) \leq \sigma Z')$;
2. $(\forall x \in \mathrm{Vars}.\mathtt{depth}_x(T) \leq \rho'(x))$;
3. $((k - j), v) \in [\![\sigma(\tau_1; q \xrightarrow{\rho; Z} \tau_2)]\!]_{\mathrm{V}}$.
By E-FIX, let $v = (\sigma\,\mathtt{fix}\, f(x).e, \theta)$, we know:
(2). $(\theta, \sigma\,\mathtt{fix}\, f(x).e) \Downarrow ((\sigma\,\mathtt{fix}\, f(x).e, \theta), \sigma\,\mathtt{fix}\, f(x).e)$;
(3). $|\sigma\,\mathtt{fix}\, f(x).e| = j \wedge j < k$.
Suppose (2), (3), STS:
1. $\mathtt{adap}(\sigma\,\mathtt{fix}\, f(x).e) = 0 \leq \sigma Z'$;
2. $\forall x \in \mathrm{Vars}.\mathtt{depth}_x(\sigma\,\mathtt{fix}\, f(x).e) = \bot \leq \rho'(x)$;
3. $((k - j), (\sigma\,\mathtt{fix}\, f(x).e, \theta)) \in [\![\sigma(\tau_1; q \xrightarrow{\rho; Z} \tau_2)]\!]_{\mathrm{V}}$.
1. and 2. are proved by definition. 3., is proved by general theorem:
Set $k - j = k'$, $\forall m \leq k', (m, (\sigma\,\mathtt{fix}\, f(x).e, \theta)) \in [\![\sigma(\tau_1; q \xrightarrow{\rho; Z} \tau_2)]\!]_{\mathrm{V}}$.
Induction on m:

Subcase 1: $m = 0$,
TS: $\forall j' < 0$,
$(j', (\theta[x \mapsto v_m, f \mapsto (\sigma\,\mathtt{fix}\, f(x).e, \sigma\theta)], e)) \in [\![\sigma(\tau_2)]\!]_{\mathrm{E}}^{\rho[x:q, f:\infty], Z}$,
it is obviously true because $j' < 0 \notin \mathbb{N}$.

Subcase 2: $m = m' + 1 \leq k'$,
TS: $(m, (\sigma\,\mathtt{fix}\, f(x).e, \theta)) \in [\![\sigma(\tau_1; q \xrightarrow{\rho; Z} \tau_2)]\!]_{\mathrm{V}}$.
Pick $\forall j' < m' + 1$, $\forall (j', v_m) \in [\![\tau_1]\!]_{\mathrm{V}}$,
STS: (4). $(j', (\theta[x \mapsto v_m, f \mapsto (\sigma\,\mathtt{fix}\, f(x).e, \theta)], \sigma e)) \in [\![\sigma\tau_2]\!]_{\mathrm{E}}^{\rho[x:q, f:\infty], Z}$.
By sub ih, we have: (5). $(m', \sigma(\mathtt{fix}\, f(x).e, \theta)) \in [\![\sigma(\tau_1; q \xrightarrow{\rho; Z} \tau_2)]\!]_{\mathrm{V}}$.
Pick $\theta' = \theta[x \mapsto v_m, f \mapsto \sigma(\mathtt{fix}\, f(x).e, \theta)]$,
we know: (6). $(j', \theta') \in [\![\Gamma, f : \sigma(\tau_1; q \xrightarrow{\rho; Z} \tau_2), x : \sigma\tau_1]\!]_{\mathrm{V}}$ proved by:

a) $(k, \theta) \in [\![\Gamma]\!]_{\mathrm{V}}$, applying Lemma 13 on assumption, we get: $(j', \theta) \in [\![\Gamma]\!]_{\mathrm{V}}$.
b) $(j', v_m) \in [\![\sigma\tau_1]\!]_{\mathrm{V}}$, from the assumption.
c) $(j', (\mathtt{fix}\, f(x).e, \theta)) \in [\![\sigma(\tau_1; q \xrightarrow{\rho; Z} \tau_2)]\!]_{\mathrm{V}}$ from (5).

From above, (4) is proved by induction hypothesis on (1) and (6).

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

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. $(\forall x \in \text{Vars}.\texttt{depth}_x(T) \leq \rho'(x))$;
3. $(((k-j), v) \in [\![\sigma\tau_2]\!]_{\text{V}})$.

By ih on $\star$ and $\triangle$, we get: (1) $(k, (\theta, \sigma e_1)) \in [\![\sigma(\tau_1; q \xrightarrow{\rho; Z} \tau_2)]\!]_{\text{E}}^{\rho_1, \sigma Z_1}$.
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) $(\forall x \in \text{Vars}.\texttt{depth}_x(T_1) \leq \rho_1(x))$;
(4) $((k-j_1), v_1) \in [\![\sigma(\tau_1; q \xrightarrow{\rho; Z} \tau_2)]\!]_{\text{V}}$.
v_1 is a function by definition.
Let $v_1 = (\texttt{fix}\, f(x).e, \theta')$ (b) s.t. $T_1 = \texttt{fix}\, f(x).e$,
By inversion on (4), we know $\forall j' < (k - j_1) \wedge (j', v') \in [\![\sigma\tau_1]\!]_{\text{V}}$,
(5). $(j', (\theta'[x \mapsto v', f \mapsto (\texttt{fix}\, f(x).e, \theta')], e)) \in [\![\sigma\tau_2]\!]_{\text{E}}^{\rho[x:q, f:\infty], \sigma Z}$.
By ih on $\diamond$ and $\square$, we get: (6). $(k, (\theta, \sigma e_2)) \in [\![\sigma\tau_1]\!]_{\text{E}}^{\rho_2, \sigma Z_2}$.
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). $(\forall x \in \text{Vars}.\texttt{depth}_x(T_2) \leq \rho_2(x))$;
(9). $((k - j_2), v_2) \in [\![\sigma\tau_1]\!]_{\text{V}}$
Apply Lemma 13 on (9), we have $((k - j_2 - j_1 - 1), v_2) \in [\![\sigma\tau_1]\!]_{\text{V}}$
Pick $j' = k - j_1 - j_2 - 1, v' = v_2$, we have: $(k - j_1 - j_2 - 1, (\theta'[x \mapsto v_2, f \mapsto v_1], e)) \in [\![\sigma\tau_2]\!]_{\text{E}}^{\rho[x:q, f:\infty], \sigma Z}$ (10) from (5).
Inversion on (10), let $\theta'' = \theta'[x \mapsto v_2, f \mapsto v_1]$, pick any v'', T'', j'', s.t. $((\theta'', e) \Downarrow (v'', T''))$ $(d) \wedge (|T''| = j'') \wedge (j'' \leq k - j_1 - j_2 - 1)$, we have:
(11). $\texttt{adap}(T'') \leq \sigma Z$;
(12). $(\forall x \in \texttt{depth}_x(T'') \leq \rho[x : q, f : \infty](x))$;
(13). $(k - j_1 - j_2 - j'' - 1, v'') \in [\![\sigma\tau_2]\!]_{\text{V}}$.
Apply E-APP rule on $(a)(b)(c)(d)$ we have:

$$\frac{\theta, \sigma e_1 \Downarrow v_1, T_1\ (a) \qquad v_1 = (\texttt{fix}\, f(x).e, \theta')\ (b) \qquad \theta, \sigma e_2 \Downarrow v_2, T_2\ (c) \qquad \theta'[f \mapsto v_1, x \mapsto v_2], e \Downarrow v'', T''\ (d)}{\theta, \sigma(e_1\ e_2) \Downarrow v'', T_1\ T_2 \triangleright \texttt{fix}\, f(x).T''}$$

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

Case $\dfrac{\begin{array}{c}\Delta;\Gamma;\rho_1 \vdash_{Z_1} e_1 : \texttt{bool}\ (\star) \\ \Delta;\Gamma;\rho \vdash_Z e_2 : \tau\ (\diamond) \qquad \Delta;\Gamma;\rho \vdash_Z e_3 : \tau\ (\triangle) \\ Z' = Z_1 + Z \qquad \rho' = \max(\rho_1, Z_1 + \rho)\end{array}}{\Delta;\Gamma;\rho' \vdash_{Z'} \texttt{if}(e_1, e_2, e_3) : \tau}\ \textbf{IF}$

Assume $\sigma \in [\![\Delta]\!]_{\mathrm{V}}$, $(k, \theta) \in [\![\sigma\Gamma]\!]_{\mathrm{V}}$. TS: $(k, (\theta, \sigma\texttt{if}(e_1, e_2, e_3))) \in [\![\sigma\tau]\!]_{\mathrm{E}}^{\rho', \sigma Z'}$.
Inversion on it, $\forall v, T, j\ (\theta, \sigma\texttt{if}(e_1, e_2, e_3) \Downarrow v, T) \wedge (|T| = j) \wedge (j \leq k)$
STS: $(\texttt{adap}(T) \leq \sigma Z') \wedge (\forall x \in \mathrm{Vars}.\texttt{depth}_x(T) \leq \rho'(x)) \wedge ((k-j, v) \in [\![\sigma\tau]\!]_{\mathrm{V}})$.
By induction hypothesis on $\star$ and $(k-1, \theta) \in [\![\Gamma]\!]_{\mathrm{V}}$ by Lemma 13, we get: $(k-1, (\theta, e_1)) \in [\![\texttt{bool}]\!]_{\mathrm{E}}^{\rho_1, Z_1}$ (1).
Inversion on (1), pick $v_1, T_1, j_1.(\theta, \sigma e_1 \Downarrow v_1, T_1)\ (a) \wedge (|T_1| = j_1) \wedge (j_1 \leq k-1)$, we know: $(\texttt{adap}(T_1) \leq \sigma Z_1)\ (4) \wedge (\forall x. \in \texttt{depth}_x(T_1) \leq \rho_1(x))\ (5) \wedge ((k-1-j_1, v_1) \in [\![\texttt{bool}]\!]_{\mathrm{V}})$
By induction hypothesis on $\diamond$ and $(k-1-j_1, \theta) \in [\![\Gamma]\!]_{\mathrm{V}}$ by Lemma 13 on the assumption, we get: $(k-1-j_1, (\theta, \sigma e_2)) \in [\![\sigma\tau]\!]_{\mathrm{E}}^{\rho, \sigma Z}$ (2)
By induction hypothesis on $\triangle$ and $(k-1-j_1, \theta) \in [\![\Gamma]\!]_{\mathrm{V}}$, by Lemma 13 on the assumption, we get: $(k-1-j_1, (\theta, \sigma e_3)) \in [\![\sigma\tau]\!]_{\mathrm{E}}^{\rho, \sigma Z}$ (3)
Induction on v_1, we have two following subcases:

Subcase1: $v_1 = \texttt{true}$,
inversion on (2), pick any v_2, T_2, j_2, s.t., $(\theta, \sigma e_2 \Downarrow v_2, T_2)\ (b) \wedge (|T_2| = j_2) \wedge (j_2 \leq k-1-j_1)$,
we know: $(\texttt{adap}(T_2) \leq \sigma Z)\ (6) \wedge (\forall x \in \mathrm{Vars}.\texttt{depth}_x(T_2) \leq \rho(x))\ (7) \wedge ((k-1-j_1-j_2, v_2) \in [\![\sigma\tau]\!]_{\mathrm{V}})\ (8)$.
Apply E-IFT on $(a)(b)$, we get:

$$\frac{\theta, \sigma e_1 \Downarrow \texttt{true}, T_1\ (a) \qquad \theta, \sigma e_2 \Downarrow v_2, T_2\ (b)}{\theta, \sigma\texttt{if}(e_1, e_2, e_3) \Downarrow v_2, \texttt{if}^{\texttt{t}}(T_1, T_2)}$$

Pick $v = v_2, T = \texttt{if}^{\texttt{t}}(T_1, T_2), j = j_1 + j_2 + 1$ s.t.,
$\theta, \sigma\texttt{if}(e_1, e_2, e_3) \Downarrow v, T \wedge j = |T| \wedge j = j_1 + j_2 + 1 \leq k$.
The following 3 goals are proved:
1. $\texttt{adap}(T) = \texttt{adap}(\texttt{if}^{\texttt{t}}(T_1, T_2)) = \texttt{adap}(T_1) + \texttt{adap}(T_2)) \leq \sigma Z_1 + \sigma Z = \sigma Z'$ is proved by $(4), (6)$.
2. $\forall x \in \mathrm{Vars}$ s.t., $\texttt{depth}_x(T) = \max(\texttt{depth}_x(T_1), \texttt{adap}(T_1) + \texttt{depth}_x(T_2)) \leq \max(\rho_1(x), \sigma Z_1 + \rho(x)) = \rho'(x)$ is proved by $(5), (7)$.
3. $(k-j, v) = (k-1-j_1-j_2, v_2) \in [\![\sigma\tau]\!]_{\mathrm{V}}$ is proved by (8).

Subcase2: $v_1 = \texttt{false}$
inversion on (3), pick any v_3, T_3, j_3 s.t., $(\theta, \sigma e_3 \Downarrow v_3, T_3)\ (b) \wedge (|T_3| = j_3) \wedge (j_3 \leq k-1-j_1)$,
we know: $(\texttt{adap}(T_3) \leq \sigma Z)\ (9) \wedge (\forall x \in \mathrm{Vars}.\texttt{depth}_x(T_3) \leq \rho(x))\ (10) \wedge ((k-1-j_1-j_3, v_3) \in [\![\sigma\tau]\!]_{\mathrm{V}})\ (11)$.
Apply E-IFT on $(a)(b)$, we get:

$$\frac{\theta, \sigma e_1 \Downarrow \texttt{true}, T_1\ (a) \qquad \theta, \sigma e_3 \Downarrow v_3, T_3\ (b)}{\theta, \sigma\texttt{if}(e_1, e_2, e_3) \Downarrow v_3, \texttt{if}^{\texttt{t}}(T_1, T_3)}$$

Pick $v = v_3, T = \mathtt{if}^{\mathtt{t}}(T_1, T_3), j = j_1 + j_3 + 1$, s.t.
$\theta, \sigma\mathtt{if}(e_1, e_2, e_3) \Downarrow v, T \wedge j = |T| \wedge j = j_1 + j_3 + 1 \leq k$.
The following 3 goals are proved:
1. $\mathtt{adap}(T) = \mathtt{adap}(\mathtt{if}^{\mathtt{t}}(T_1, T_3)) = \mathtt{adap}(T_1) + \mathtt{adap}(T_3)) \leq \sigma Z_1 + \sigma Z = \sigma Z'$ proved by $(4), (9)$.
2. $\forall x \in \mathrm{Vars}.\mathtt{depth}_x(T) = \max(\mathtt{depth}_x(T_1), \mathtt{adap}(T_1) + \mathtt{depth}_x(T_3)) \leq \max(\rho_1(x), \sigma Z_1 + \rho(x)) = \rho'(x)$ proved by $(5), (10)$.
3. $(k - j, v) = (k - 1 - j_1 - j_3, v_3) \in [\![\tau]\!]_{\mathrm{V}}$ proved by (11).

Case $\dfrac{\Delta; \Gamma; \rho \vdash_Z e : \tau_1 \times \tau_2}{\Delta; \Gamma; \rho \vdash_Z \mathtt{fst}(e) : \tau_1}$ **FST**

Assume $\sigma \in [\![\Delta]\!]_{\mathrm{V}}$, $(k, \theta) \in [\![\sigma\Gamma]\!]_{\mathrm{V}}$. TS: $(k, (\theta, \sigma\mathtt{fst}(e))) \in [\![\sigma\tau_1]\!]_{\mathrm{E}}^{\rho, \sigma Z}$.
Unfold, pick any $v, T, j.(\theta, \sigma\mathtt{fst}(e) \Downarrow v, T) \wedge (|T| = j) \wedge (j \leq k)$,
STS: 1. $(\mathtt{adap}(T) \leq \sigma Z)$;
2. $(\forall x \in \mathrm{Vars}.\mathtt{depth}_x(T) \leq \rho(x))$;
3. $(k - j, v) \in [\![\sigma\tau_1]\!]_{\mathrm{V}}$.
By induction hypothesis, we get: (1). $(k - 1, (\theta, \sigma e)) \in [\![\sigma(\tau_1 \times \tau_2)]\!]_{\mathrm{E}}^{\rho, \sigma Z}$.
Inversion on (1), $\forall v', T', j'.(\theta, \sigma e \Downarrow v', T') \wedge (|T'| = j') \wedge (j' \leq k - 1)$,
We know: $(\mathtt{adap}(T') \leq \sigma Z)$ $(a) \wedge (\forall x \in \mathrm{Vars}.\mathtt{depth}_x(T') \leq \rho(x))$ $(b) \wedge ((k - 1 - j', v') \in [\![\sigma(\tau_1 \times \tau_2)]\!]_{\mathrm{V}})$ (c).
Pick any v_1, v_2, $v' = (v_1, v_2)$, inversion on (c), we have:
$(k - j', v_1) \in [\![\sigma\tau_1]\!]_{\mathrm{V}}$ $(d) \wedge (k - j', v_2) \in [\![\sigma\tau_2]\!]_{\mathrm{V}}$.
By E-FST, pick $v = v_1, j = j' + 1, T = \mathtt{fst}(T')$, the 3 goals are proved:
1. $\mathtt{adap}(T) = \mathtt{adap}(\mathtt{fst}(T')) = \mathtt{adap}(T') \leq \sigma Z$ by (a).
2. $\forall x.\mathtt{depth}_x(T) \implies \forall x.\mathtt{depth}_x(T') \leq \rho(x)$ by (b).
3. $(k - j, v) = (k - 1 - j', v_2) \in [\![\sigma\tau_1]\!]_{\mathrm{V}}$ by (d).

Case $\dfrac{\Delta; \Gamma; \rho \vdash_Z e : \tau_1 \times \tau_2}{\Delta; \Gamma; \rho \vdash_Z \mathtt{snd}(e) : \tau_2}$ **SND**

Assume $\sigma \in [\![\Delta]\!]_{\mathrm{V}}$, $(k, \theta) \in [\![\sigma\Gamma]\!]_{\mathrm{V}}$. TS: $(k, (\theta, \sigma\mathtt{snd}(e))) \in [\![\sigma\tau_2]\!]_{\mathrm{E}}^{\rho, \sigma Z}$.
Unfold, pick any $v, T, j.(\theta, \sigma\mathtt{snd}(e) \Downarrow v, T) \wedge (|T| = j) \wedge (j \leq k)$,
STS: 1. $(\mathtt{adap}(T) \leq \sigma Z)$;
2. $(\forall x \in \mathrm{Vars}.\mathtt{depth}_x(T) \leq \rho(x))$;
3. $(k - j, v) \in [\![\sigma\tau_1]\!]_{\mathrm{V}}$.
By induction hypothesis, we get: $(k - 1, (\theta, \sigma e)) \in [\![\sigma(\tau_1 \times \tau_2)]\!]_{\mathrm{E}}^{\rho, \sigma Z}$ (1).
Inversion on (1), $\forall v', T', j'.(\theta, \sigma e \Downarrow v', T') \wedge (|T'| = j') \wedge (j' \leq k - 1)$,
We know: $(\mathtt{adap}(T') \leq \sigma Z)$ $(a) \wedge (\forall x \in \mathtt{depth}_x(T') \leq \rho(x))$ $(b) \wedge ((k - 1 - j', v') \in [\![\sigma(\tau_1 \times \tau_2)]\!]_{\mathrm{V}})$ (c).
Pick any v_1, v_2, $v' = (v_1, v_2)$, inversion on (c), we have:
$(k - j', v_1) \in [\![\sigma\tau_1]\!]_{\mathrm{V}}$ $(d) \wedge (k - j', v_2) \in [\![\sigma\tau_2]\!]_{\mathrm{V}}$ (e).
By E-SND, pick $v = v_2, j = j' + 1, T = \mathtt{snd}(T')$, following 3 goals are proved:
1. $\mathtt{adap}(T) = \mathtt{adap}(\mathtt{snd}(T')) = \mathtt{adap}(T') \leq \sigma Z$ by (a)

2. $\forall x \in \text{Vars}.\texttt{depth}_x(T) \implies \forall x.\texttt{depth}_x(T') \leq \rho(x)$ proved by (b).
3. $(k-j, v) = (k-1-j', v_2) \in [\![\sigma\tau_2]\!]_{\text{V}}$ proved by (e).

Case $\dfrac{}{\Delta;\Gamma;\rho \vdash_Z \texttt{true} : \texttt{bool}}$ **TRUE**

Assume $\sigma \in [\![\Delta]\!]_{\text{V}}$, $(k,\theta) \in [\![\texttt{bool}]\!]_{\text{V}}$. TS: $(k,(\theta,\texttt{true})) \in [\![\texttt{bool}]\!]_{\text{E}}^{\rho,\sigma Z}$.
By inversion, STS: $\forall v, T, j.(\theta, \texttt{true} \Downarrow v, T) \wedge (|T| = j) \wedge (j \leq k)$
1. $(\texttt{adap}(T) \leq \sigma Z)$ 2. $(\forall x \in \text{Vars}.\texttt{depth}_x(T) \leq \rho(x))$ 3. $(k-j, v) \in [\![\texttt{bool}]\!]_{\text{V}}$
By E-TRUE, let $v = \texttt{true}$, $T = \texttt{true}$ and $j = |\texttt{true}| = 0$.
Following 3 goals are proved:
1. $\texttt{adap}(T) = 0 \leq \sigma Z$.
2. $\forall x \in \text{Vars}.\texttt{depth}_x(\texttt{true}) = 0 \leq \rho(x)$
3. $(k-j, v) = (k, \texttt{true}) \in [\![\texttt{bool}]\!]_{\text{V}}$.

Case $\dfrac{}{\Delta;\Gamma;\rho \vdash_Z \texttt{false} : \texttt{bool}}$ **FALSE**

Assume $\sigma \in [\![\Delta]\!]_{\text{V}}$, $(k,\theta) \in [\![\texttt{bool}]\!]_{\text{V}}$. TS: $(k,(\theta,\texttt{false})) \in [\![\texttt{bool}]\!]_{\text{E}}^{\rho,\sigma Z}$.
By inversion, STS: $\forall v, T, j.(\theta, \texttt{false} \Downarrow v, T) \wedge (|T| = j) \wedge (j \leq k)$
1. $(\texttt{adap}(T) \leq Z)$ 2. $(\forall x \in \text{Vars}.\texttt{depth}_x(T) \leq \rho(x))$ 3. $(k-j, v) \in [\![\texttt{bool}]\!]_{\text{V}}$
By E-TRUE, let $v = \texttt{false}$, $T = \texttt{false}$ and $j = |\texttt{false}| = 0$.
Following 3 goals are proved:
1. $\texttt{adap}(T) = 0 \leq \sigma Z$.
2. $\forall x \in \text{Vars}.\texttt{depth}_x(\texttt{false}) = 0 \leq \rho(x)$
3. $(k-j, v) = (k, \texttt{false}) \in [\![\texttt{bool}]\!]_{\text{V}}$.

Case $\dfrac{\Delta;\Gamma(x) = \tau \qquad 0 \leq \rho(x) \textbf{ or equiv. } \rho(x) \neq \bot}{\Delta;\Gamma;\rho \vdash_Z x : \tau}$ **VAR**

Assume $\sigma \in [\![\Delta]\!]_{\text{V}}$, $(k,\theta) \in [\![\sigma\Gamma]\!]_{\text{V}}$. TS: $(k,(\theta,\sigma x)) \in [\![\sigma\tau]\!]_{\text{E}}^{\rho,\sigma Z}$.
By inversion, STS: $\forall v, T, j.(\theta, \sigma x \Downarrow v, T) \wedge (|T| = j) \wedge (j \leq k)$
1. $(\texttt{adap}(T) \leq \sigma Z)$; 2. $(\forall x \in \text{Vars}.\texttt{depth}_x(T) \leq \rho(x))$; 3. $(k-j, v) \in [\![\tau]\!]_{\text{V}}$
By E-VAR, pick $v = \theta(\sigma x)$, $T = \sigma x$, $j = |T| = 0$, following 3 goals are proved:
1. $(\texttt{adap}(T) = \texttt{adap}(\sigma x) = 0 \leq \sigma Z)$
2. $\forall y \in \texttt{depth}_y(\sigma x)$,
$y = \sigma x$ $\texttt{depth}_y(\sigma x) = 0 \leq \rho(\sigma x)$
$y \neq \sigma x$ $\texttt{depth}_y(\sigma x) = \bot \leq \rho(y)$
3. $(k-j, v) = (k, \theta(\sigma x)) \in [\![\sigma\tau]\!]_{\text{V}}$.
By definition of $(k,\theta) \in [\![\Gamma]\!]_{\text{V}}$, we have: $(k, \theta(\sigma x)) \in [\![\Gamma(\sigma x)]\!]_{\text{V}}$, and $\Gamma(\sigma x) = \tau$.

Case $\dfrac{\rho \vdash \tau\ \texttt{wf}}{\Delta;\Gamma;\rho \vdash_Z \texttt{nil} : \texttt{list}\,\tau}$

Assume $\sigma \in [\![\Delta]\!]_{\text{V}}$, $(k,\theta) \in [\![\sigma\Gamma]\!]_{\text{V}}$. TS: $(k,(\theta,\texttt{nil})) \in [\![\sigma\texttt{list}\,\tau]\!]_{\text{E}}^{\rho,\sigma Z}$.

By inversion, STS: $\forall v, T, j.(\theta, \texttt{nil} \Downarrow v, T) \wedge (|T| = j) \wedge (j \leq k)$
1. $(\texttt{adap}(T) \leq \sigma Z)$; 2. $(\forall x \in \text{Vars}.\texttt{depth}_x(T) \leq \rho(x))$; 3. $(k - j, v) \in [\![\sigma\tau]\!]_{\text{V}}$
By E-NIL, we know : $v = \texttt{nil}$ and $T = \texttt{nil}$ and $|\texttt{nil}| = 0 \leq k$.
The following 3 goals are proved:
1. $(\texttt{adap}(T) = \texttt{adap}(\texttt{nil}) = 0 \leq \sigma Z)$, because σZ is not negative.
2. $\forall x \in \text{Vars}\texttt{depth}_x(\texttt{nil})$, $\texttt{depth}_x(\texttt{nil}) = \bot \leq \rho(x)$ proved from the definition of $\bot$.
3. $(k - 0, \texttt{nil}) \in [\![\sigma\texttt{list}\,\tau]\!]_{\text{V}}$, proved by inversion of the interpretation of $[\![\sigma\texttt{list}\,\tau]\!]_{\text{V}}$.

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

Assume $\sigma \in [\![\Delta]\!]_{\text{V}}$, $(k, \theta) \in [\![\sigma\Gamma]\!]_{\text{V}}$. TS: $(k, (\theta, \sigma\,\texttt{cons}(e_1, e_2))) \in [\![\sigma\texttt{list}\,\tau]\!]_{\text{E}}^{\rho', \sigma Z'}$.
By inversion, STS: $\forall v, T, j.(\theta, \sigma\,\texttt{cons}(e_1, e_2) \Downarrow v, T) \wedge (|T| = j) \wedge (j \leq k)$
1. $(\texttt{adap}(T) \leq \sigma Z')$; 2. $(\forall x \in \text{Vars}.\texttt{depth}_x(T) \leq \rho'(x))$; 3. $(k - j, v) \in [\![\sigma\texttt{list}\,\tau]\!]_{\text{V}}$
By ih on $(\star)$, we get: $(k, (\theta, \sigma e_1)) \in [\![\tau]\!]_{\text{E}}^{\rho_1, \sigma Z_1}$ (1).
By inversion, we pick v_1, T_1 and j_1 s.t. $\theta, \sigma e_1 \Downarrow v_1, T_1$ and $|T_1| = j_1$ and $j_1 \leq k$.
We know : $(\texttt{adap}(T_1) \leq \sigma Z_1)$ (2) $\wedge$ $(\forall x \in \text{Vars}.\texttt{depth}_x(T_1) \leq \rho_1(x))$ (3) $\wedge$ $(k - j_1, v_1) \in [\![\sigma\tau]\!]_{\text{V}}$ (4).
By ih on $(\diamond)$ and $(k - j_1, \theta) \in [\![\Gamma]\!]_{\text{V}}$ by Lemma 13, we get: $(k - j_1, (\theta, \sigma e_2)) \in [\![\sigma\texttt{list}\,\tau]\!]_{\text{E}}^{\rho_2, \sigma Z_2}$ (5).
By inversion, we pick v_2, T_2 and j_2, s.t. $\theta, \sigma e_2 \Downarrow v_2, T_2$ and $|T_2| = j_2$ and $j_2 \leq k - j_1$.
We know : $(\texttt{adap}(T_2) \leq \sigma Z_2)$ (6) $\wedge$ $(\forall x \in \text{Vars}.\texttt{depth}_x(T_2) \leq \rho_2(x))$ (7) $\wedge$ $(k - j_2, v_2) \in [\![\sigma\texttt{list}\,\tau]\!]_{\text{V}}$ (8).

$$\frac{\theta, \sigma e_1 \Downarrow v_1, T_1 \qquad \theta, \sigma e_2 \Downarrow v_2, T_2}{\theta, \sigma\,\texttt{cons}(e_1, e_2) \Downarrow \texttt{cons}(v_1, v_2), \texttt{cons}(T_1, T_2)}$$

By E-Cons, we set : $v = \texttt{cons}(v_1, v_2)$ and $T = \texttt{cons}(T_1, T_2)$ s.t. $j = |T| = j_1 + j_2 \leq k$.
3 following goals are proved:
1. $(\texttt{adap}(T) = \texttt{adap}(\texttt{cons}(T_1, T_2)) \leq \sigma Z')$, by (2) and (6).
2. $\forall x \in \text{Vars}.\texttt{depth}_x(\texttt{cons}(T_1, T_2)) = \max(\texttt{depth}_x(T_1), \texttt{depth}_x(T_2)) \leq \rho'(x)$ by (3) and (7).
3. $(k - j_1 - j_2, \texttt{cons}(v_1, v_2)) \in [\![\sigma\texttt{list}\,\tau]\!]_{\text{V}}$. By inversion of the definition of $[\![\sigma\texttt{list}\,\tau]\!]_{\text{V}}$, suffice to show: $(k - j_1 - j_2, v_1) \in [\![\sigma\tau]\!]_{\text{V}}$, which is proved by Lemma 13 on (4), $(k - j_1 - j_2, v_2) \in [\![\sigma\texttt{list}\,\tau]\!]_{\text{V}}$ is proved by (8).

Case $\dfrac{\Delta;\Gamma;\rho_1 \vdash_{Z_1} e_1 : \tau_1\ (\star) \qquad \Delta;\Gamma, x:\tau_1;\rho_2[x:q] \vdash_{Z_2} e_2 : \tau_2\ (\diamond) \quad \rho' = \max(\rho_2, \rho_1+q) \qquad Z' = \max(Z_2, Z_1+q)}{\Delta;\Gamma;\rho' \vdash_{Z'} \mathtt{let}\, x = e_1\, \mathtt{in}\, e_2 : \tau}$

Assume $\sigma \in [\![\Delta]\!]_{\mathrm{V}}$, $(k,\theta) \in [\![\sigma\Gamma]\!]_{\mathrm{V}}$ $(\triangle)$.
TS: $(k, (\theta, \sigma(\mathtt{let}\, x = e_1\, \mathtt{in}\, e_2))) \in [\![\sigma\tau]\!]_{\mathrm{E}}^{\rho', \sigma Z'}$.
By inversion, STS: $\forall v, T, j$,
s.t., $(\theta, \sigma(\mathtt{let}\, x = e_1\, \mathtt{in}\, e_2) \Downarrow v, T) \wedge (|T| = j) \wedge (j \le k)$,
1. $(\mathtt{adap}(T) \le \sigma Z')$;
2. $(\forall x \in \mathrm{Vars}.\mathtt{depth}_x(T) \le \rho'(x))$;
3. $(k-j, v) \in [\![\sigma\tau]\!]_{\mathrm{V}}$ By ih on $(\star)$, we get: $(k, (\theta, \sigma e_1)) \in [\![\sigma\tau_1]\!]_{\mathrm{E}}^{\rho_1, \sigma Z_1}$ (1).
By inversion, pick any v_1, T_1, j_1, s.t. $(\theta, \sigma e_1) \Downarrow (v_1, T_1) \wedge |T_1| = j_1 \wedge j_1 \le k$.
We know: $\mathtt{adap}(T_1) \le \sigma Z_1$ (2) $\wedge \forall x \in \mathrm{Vars}.\mathtt{depth}_x(T_1) \le \rho_1(x)$ (3) $\wedge (k - j_1, v_1) \in [\![\sigma\tau_1]\!]_{\mathrm{V}}$ (4).
By Lemma 13 on $(\triangle)$, we get : $(k - j_1, \theta) \in [\![\Gamma]\!]_{\mathrm{V}}$ (5).
Set $\theta' = \theta[x \mapsto v_1]$, we know: $(k - j_1, \theta') \in [\![\Gamma, x : \sigma\tau_1]\!]_{\mathrm{V}}$ (6) by (4) and (5).
By ih on $(\diamond)$ and (6), we know: $(k - j_1, (\theta', \sigma e_2)) \in [\![\sigma\tau]\!]_{\mathrm{E}}^{\rho_2[x:q], \sigma Z_2}$ (7).
By inversion (7), pick any v_2, T_2, j_2,
s.t., $(\theta', \sigma e_2) \Downarrow (v_2, T_2) \wedge |T_2| = j_2 \wedge j_2 \le k - j_1$.
We know: $(\mathtt{adap}(T_2) \le \sigma Z_2)$ (8)$\wedge(\forall x \in \mathrm{Vars}.\mathtt{depth}_x(T_2) \le \rho_2[x:q](x))$ (9)$\wedge$ $(k - j_1 - j_2, v_2) \in [\![\sigma\tau]\!]_{\mathrm{V}}$ (10).

$$\frac{\theta, \sigma e_1 \Downarrow v_1, T_1 \qquad \theta[x \mapsto v_1], \sigma e_2 \Downarrow v_2, T_2}{\theta, \sigma(\mathtt{let}\, x; q = e_1\, \mathtt{in}\, e_2) \Downarrow v_2, \mathtt{let}(x, T_1, T_2)}$$

By E-LET, pick $v = v_2$, $T = \sigma\, \mathtt{let}(x, T_1, T_2)$, s.t., $j = j_1 + j_2 \le k$,
the following 3 goals are proved:
1. $(\mathtt{adap}(T) = \mathtt{adap}(\sigma\, \mathtt{let}(x, T_1, T_2)) \le \sigma Z')$, proved by (2) ,(8)and (9).
2. $\forall y \in \mathrm{Vars}$, $\mathtt{depth}_y(\sigma\, \mathtt{let}(x, T_1, T_2)) \le \rho'(y)$ proved by (3) and (9).
3. $(k - j_1 - j_2, v_2) \in [\![\sigma\tau]\!]_{\mathrm{V}}$, proved by (10).

□

$$
\begin{array}{lll}
\text{Expr.} & e & ::= \quad x \mid e_1\ e_2 \mid \texttt{fix}\, f(x:\tau).e \mid (e_1, e_2) \mid \texttt{fst}(e) \mid \texttt{snd}(e) \mid \\
& & \quad \texttt{true} \mid \texttt{false} \mid \texttt{if}(e_1, e_2, e_3) \mid c \mid \delta(e) \mid \Lambda.e \mid e\ [] \\
& & \quad \mid \texttt{let}\, x : q = e_1\, \texttt{in}\, e_2 \mid \texttt{nil} \mid \texttt{cons}(e_1, e_2) \\
& & \quad \mid \quad \texttt{bernoulli}\ e \mid \texttt{uniform}\ e_1\ e_2 \\
\text{Value} & v & ::= \quad \texttt{true} \mid \texttt{false} \mid c \mid (\texttt{fix}\, f(x:\tau).e, \theta) \mid (v_1, v_2) \mid \texttt{nil} \mid \texttt{cons}(v_1, v_2) \mid \\
& & \quad (\Lambda.e, \theta)
\end{array}
$$

Figure 14: ADAPTFUN syntax

$$
\begin{array}{lll}
\text{Expr.} & e & ::= \quad x \mid e_1\ e_2 \mid \texttt{fix}\, f(x:\tau).e \mid (e_1, e_2) \mid \texttt{fst}(e) \mid \texttt{snd}(e) \mid \\
& & \quad \texttt{true} \mid \texttt{false} \mid \texttt{if}(e_1, e_2, e_3) \mid c \mid \delta(e) \mid \Lambda.i.e \mid e\ [I] \\
& & \quad \mid \texttt{let}\, x : q = e_1\, \texttt{in}\, e_2 \mid \texttt{nil} \mid \texttt{cons}(e_1, e_2) \\
& & \quad \mid \quad \texttt{bernoulli}\ e \mid \texttt{uniform}\ e_1\ e_2 \mid \mathsf{box}\ e \mid \mathsf{der}\ e \\
\text{Value} & v & ::= \quad \texttt{true} \mid \texttt{false} \mid c \mid (\texttt{fix}\, f(x:\tau).e, \theta) \mid (v_1, v_2) \mid \texttt{nil} \mid \texttt{cons}(v_1, v_2) \mid \\
& & \quad (\Lambda.e, \theta)
\end{array}
$$

Figure 15: ADAPTFUN core syntax

7 Bidirectional Type Checking soundness

$$
\begin{array}{lll}
\text{Expr.} & e & ::= \quad x \mid e_1\ e_2 \mid \texttt{fix}\, f(x:\tau).e \mid (e_1, e_2) \mid \texttt{fst}(e) \mid \texttt{snd}(e) \mid \\
& & \quad \texttt{true} \mid \texttt{false} \mid \texttt{if}(e_1, e_2, e_3) \mid c \mid \delta(e) \mid \Lambda.i.e \mid e\ [I] \\
& & \quad \mid \texttt{let}\, x : q = e_1\, \texttt{in}\, e_2 \mid \texttt{nil} \mid \texttt{cons}(e_1, e_2) \\
& & \quad \mid \quad \texttt{bernoulli}\ e \mid \texttt{uniform}\ e_1\ e_2 \mid \mathsf{box}\ e \mid \mathsf{der}\ e \mid (e : \tau, \rho, Z) \\
\text{Value} & v & ::= \quad \texttt{true} \mid \texttt{false} \mid c \mid (\texttt{fix}\, f(x:\tau).e, \theta) \mid (v_1, v_2) \mid \texttt{nil} \mid \texttt{cons}(v_1, v_2) \mid \\
& & \quad (\Lambda.e, \theta)
\end{array}
$$

Figure 16: ADAPTFUN algorithmic syntax

$$\frac{\Gamma(x) = \tau \qquad 0 \leq \rho(x) \text{ or equiv. } \rho(x) \neq \bot}{\Delta; \Gamma; \rho \vdash_Z x :^c \tau} \textbf{ c-var}$$

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

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

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

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

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

$$\frac{\Delta; \Gamma; \rho' \vdash_{Z'} e :^c : \tau' \qquad \rho' \leq \rho \qquad Z' \leq Z \qquad \Delta \models \tau' \equiv^c \tau}{\Delta; \Gamma; \rho \vdash_Z e :^c \tau} \textbf{ c-}\equiv$$

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

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

Figure 17: Core Typing rules, part 1

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

$$\frac{}{\Delta;\Gamma;\rho \vdash_Z c :^c \texttt{b}}\ \textbf{c-const} \qquad \frac{\Delta;\rho \vdash \tau\ \texttt{wf}}{\Delta;\Gamma;\rho \vdash_Z \texttt{nil} :^c \texttt{list}\,\tau}\ \textbf{c-nil}$$

$$\frac{\Delta;\Gamma;\rho \vdash_Z e :^c \Box\,((\tau_1; 0 \xrightarrow{\rho'';0} \tau_2)) \qquad Z' = 1 + Z \qquad \rho' = 1 + \max(\rho, \rho'' + Z)}{\Delta;\Gamma;\rho' \vdash_{Z'} \delta(e) :^c \texttt{real}}\ \textbf{c-}\delta$$

$$\frac{\Delta;\Box\,(\Gamma);\rho \vdash_Z e :^c \tau \qquad \delta \notin e \qquad \texttt{dom}(\Gamma') = \texttt{dom}(\rho')}{\Delta;\Box\,(\Gamma),\Gamma';\rho,\boxed{\rho'} \vdash_0 \mathsf{box}\ e :^c \Box\,(\tau)}\ \textbf{c-box}$$

$$\frac{\Delta;\Gamma;\rho \vdash_Z e :^c \Box\,(\tau)}{\Delta;\Gamma;\rho \vdash_Z \mathsf{der}\ e :^c \tau}\ \textbf{c-der}$$

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

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

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

$$\frac{\Delta;\Gamma;\rho \vdash_Z e :^c \forall i \overset{\rho_1;Z_1}{::} S.\,\tau \qquad \Delta \vdash I :: S \qquad \rho' = \max(\rho, Z + \rho_1) \qquad Z' = Z_1[I/i] + Z}{\Delta;\Gamma;\rho' \vdash_{Z'} e\ [I] :^c \tau[I/i]}\ \textbf{c-iapp}$$

Figure 18: Core Typing rules, part 2

$$\frac{}{\Delta \models \mathtt{b} \equiv^c \mathtt{b}}\ \textbf{core-eq-base} \qquad \frac{}{\Delta \models \mathtt{bool} \equiv^c \mathtt{bool}}\ \textbf{core-eq-bool}$$

$$\frac{\Delta \models \tau_1 \equiv^c \tau_1' \qquad \Delta \models \tau_2 \equiv^c \tau_2'}{\Delta \models \tau_1 \times \tau_2 \equiv^c \tau_1' \times \tau_2'}\ \textbf{core-eq-pair}$$

$$\frac{\begin{array}{c}\Delta \models \tau_1 \equiv^c \tau_1' \\ \Delta \models \tau_2 \equiv^c \tau_2' \qquad \Delta \models q = q' \qquad \Delta \models Z = Z' \qquad \Delta \models \rho = \rho'\end{array}}{\Delta \models \tau_1; q \xrightarrow{\rho; Z} \tau_2 \equiv^c \tau_1'; q' \xrightarrow{\rho'; Z'} \tau_2'}\ \textbf{core-eq-arrow}$$

$$\frac{\Delta \models I = I'}{\Delta \models \mathtt{int}[I] \equiv^c \mathtt{int}[I']}\ \textbf{core-eq-int} \qquad \frac{\Delta \models \tau \equiv^c \tau'}{\Delta \models \Box(\tau) \equiv^c \Box(\tau')}\ \textbf{core-eq-box}$$

$$\frac{\Delta \models \tau \equiv^c \tau'}{\Delta \models \mathtt{list}\,\tau \equiv^c \mathtt{list}\,\tau'}\ \textbf{core-eq-list}$$

$$\frac{i, \Delta \models \tau \equiv^c \tau' \qquad \Delta \models Z = Z' \qquad \Delta \models \rho = \rho'}{\Delta \models \forall i ::^{\rho; Z} S.\tau \equiv^c \forall i ::^{\rho'; Z'} S.\tau'}\ \textbf{core-eq-}\forall$$

Figure 19: Core equivalence Rules

$$\frac{\Gamma(x) = \tau \qquad 0 \leq \rho(x) \text{ or equiv. } \rho(x) \neq \bot}{\Delta; \Gamma; \rho \vdash_Z x \rightsquigarrow x : \tau} \textbf{ e-var}$$

$$\frac{\begin{array}{c}\Delta; \Gamma; \rho_1 \vdash_{Z_1} e_1 \rightsquigarrow e_1^* : (\tau_1; q \xrightarrow{\rho; Z} \tau_2) \qquad \Delta; \Gamma; \rho_2 \vdash_{Z_2} e_2 \rightsquigarrow e_2^* : \tau_1 \\ Z' = Z_1 + \max(Z, Z_2 + q) \qquad \rho' = \max(\rho_1, Z_1 + \max(\rho, \rho_2 + q))\end{array}}{\Delta; \Gamma; \rho' \vdash_{Z'} e_1\ e_2 \rightsquigarrow e_1^*\ e_2^* : \tau_2} \textbf{ e-app}$$

$$\frac{\Delta; \Gamma, f : (\tau_1; q \xrightarrow{\rho; Z} \tau_2), x : \tau_1; \rho[f : \infty, x : q] \vdash_Z e \rightsquigarrow e^* : \tau_2}{\Delta; \Gamma; \rho' \vdash_{Z'} \texttt{fix}\, f(x : \tau_1).e \rightsquigarrow \texttt{fix}\, f(x : \tau_1).e^* : (\tau_1; q \xrightarrow{\rho; Z} \tau_2)} \textbf{ e-fix}$$

$$\frac{\begin{array}{c}\Delta; \Gamma; \rho_1 \vdash_{Z_1} e_1 \rightsquigarrow e_1^* : \tau_1 \qquad \Delta; \Gamma; \rho_2 \vdash_{Z_2} e_2 \rightsquigarrow e_2^* : \tau_2 \\ \rho' = \max(\rho_1, \rho_2) \qquad Z' = \max(Z_1, Z_2)\end{array}}{\Delta; \Gamma; \rho' \vdash_{Z'} (e_1, e_2) \rightsquigarrow (e_1^*, e_2^*) : \tau_1 \times \tau_2} \textbf{ e-pair}$$

$$\frac{\Delta; \Gamma; \rho \vdash_Z e \rightsquigarrow e^* : \tau_1 \times \tau_2}{\Delta; \Gamma; \rho \vdash_Z \texttt{fst}(e) \rightsquigarrow \texttt{fst}(e^*) : \tau_1} \textbf{ e-fst}$$

$$\frac{\Delta; \Gamma; \rho \vdash_Z e \rightsquigarrow e^* : \tau_1 \times \tau_2}{\Delta; \Gamma; \rho \vdash_Z \texttt{snd}(e) \rightsquigarrow \texttt{snd}(e^*) : \tau_2} \textbf{ e-snd}$$

$$\frac{}{\Delta; \Gamma; \rho \vdash_Z \texttt{true} \rightsquigarrow \texttt{true} : \texttt{bool}} \textbf{ e-true}$$

$$\frac{}{\Delta; \Gamma; \rho \vdash_Z \texttt{false} \rightsquigarrow \texttt{false} : \texttt{bool}} \textbf{ e-false}$$

$$\frac{\begin{array}{c}\Delta; \Gamma; \rho' \vdash_{Z'} e \rightsquigarrow e^* : \tau' \\ \Delta \models \tau' <: \tau \qquad \rho' \leq \rho \qquad Z' \leq Z \qquad e' = \mathsf{coerce}_{\tau', \tau}\end{array}}{\Delta; \Gamma; \rho \vdash_Z e \rightsquigarrow e'\ e^* : \tau} \textbf{ e-subsumption}$$

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

$$\frac{\begin{array}{c}\Delta; \Gamma; \rho_1 \vdash_{Z_1} e_1 \rightsquigarrow e_1^* : \texttt{real} \\ \Delta; \Gamma; \rho_2 \vdash_{Z_2} e_2 \rightsquigarrow e_2^* : \texttt{real} \qquad Z = \max(Z_1, Z_2) \qquad \rho' = \max(\rho_1, \rho_2)\end{array}}{\Delta; \Gamma; \rho' \vdash_{Z'} \texttt{uniform}\ e_1\ e_2 \rightsquigarrow \texttt{uniform}\ e_1^*\ e_2^* : \texttt{real}} \textbf{ e-uniform}$$

Figure 20: Embedding rules, part 1

$$\frac{\begin{array}{c}\Delta;\Gamma;\rho_1 \vdash_{Z_1} e_1 \rightsquigarrow e_1^* : \texttt{bool} \\ \Delta;\Gamma;\rho \vdash_Z e_2 \rightsquigarrow e_2^* : \tau \qquad \Delta;\Gamma;\rho \vdash_Z e_3 \rightsquigarrow e_3^* : \tau \\ Z' = Z_1 + Z \qquad \rho' = \max(\rho_1, Z_1 + \rho)\end{array}}{\Delta;\Gamma;\rho' \vdash_{Z'} \texttt{if}(e_1, e_2, e_3) \rightsquigarrow \texttt{if}(e_1^*, e_2^*, e_3^*) : \tau}\ \textbf{e-if}$$

$$\frac{}{\Delta;\Gamma;\rho \vdash_Z c \rightsquigarrow c : \texttt{b}}\ \textbf{e-const} \qquad \frac{\Delta;\rho \vdash \tau\ \texttt{wf}}{\Delta;\Gamma;\rho \vdash_Z \texttt{nil} \rightsquigarrow \texttt{nil} : \texttt{list}\,\tau}\ \textbf{e-nil}$$

$$\frac{\begin{array}{c}\Delta;\Gamma;\rho \vdash_Z e \rightsquigarrow e^* : \Box\left((\tau_1; 0 \xrightarrow{\rho'';0} \tau_2)\right) \\ Z' = 1 + Z \qquad \rho' = 1 + \max(\rho, \rho'' + Z)\end{array}}{\Delta;\Gamma;\rho' \vdash_{Z'} \delta(e) \rightsquigarrow \delta(e^*) : \texttt{real}}\ \textbf{e-}\delta$$

$$\frac{\begin{array}{c}\Delta;\Gamma;\rho \vdash_Z e \rightsquigarrow e^* : \tau \qquad \delta \notin e \qquad \forall x_i \in \texttt{dom}(\Gamma).e_i = \mathsf{coerce}_{\Gamma(x_i),\Box(\Gamma(x_i))} \\ \forall x_i \in \texttt{dom}(\Gamma).\Delta \models \Gamma(x_i) <: \Box(\Gamma(x_i)) \qquad \texttt{dom}(\Gamma') = \texttt{dom}(\rho')\end{array}}{\Delta;\Gamma,\Gamma';\rho,\rho' \vdash_0 e \rightsquigarrow \texttt{let}\ \overline{y_i; 0 = (e_i\ x_i)}\ \texttt{in}(\texttt{box}\ e^*[\overline{y_i/x_i}]) : \Box(\tau)}\ \textbf{e-box}$$

$$\frac{\begin{array}{c}\Delta;\Gamma;\rho_1 \vdash_{Z_1} e_1 \rightsquigarrow e_1^* : \tau \\ \Delta;\Gamma;\rho_2 \vdash_{Z_2} e_2 \rightsquigarrow e_2^* : \texttt{list}\,\tau \qquad \rho' = \max(\rho_1, \rho_2) \qquad Z' = \max(Z_1, Z_2)\end{array}}{\Delta;\Gamma;\rho' \vdash_{Z'} \texttt{cons}(e_1, e_2) \rightsquigarrow \texttt{cons}(e_1^*, e_2^*) : \texttt{list}\,\tau}\ \textbf{e-cons}$$

$$\frac{\begin{array}{c}\Delta;\Gamma;\rho_1 \vdash_{Z_1} e_1 \rightsquigarrow e_1^* : \tau_1 \qquad \Delta;\Gamma, x : \tau_1;\rho_2[x : q] \vdash_{Z_2} e_2 \rightsquigarrow e_2^* : \tau \\ \rho' = \max(\rho_2, \rho_1 + q) \qquad Z' = \max(Z_2, Z_1 + q)\end{array}}{\Delta;\Gamma;\rho' \vdash_{Z'} \texttt{let}\,x; q = e_1\,\texttt{in}\,e_2 \rightsquigarrow \texttt{let}\,x; q = e_1^*\,\texttt{in}\,e_2^* : \tau}\ \textbf{e-let}$$

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

$$\frac{\begin{array}{c}\Delta;\Gamma;\rho \vdash_Z e \rightsquigarrow e^* : \forall i \overset{\rho_1;Z_1}{::} S.\tau \qquad \Delta \vdash I :: S \\ \rho' = \max(\rho, Z + \rho_1) \qquad Z' = Z_1[I/i] + Z\end{array}}{\Delta;\Gamma;\rho' \vdash_{Z'} e\ [] \rightsquigarrow e^*\ [I] : \tau[I/i]}\ \textbf{e-iapp}$$

Figure 21: Embedding rules, part 2

Lemma 15 (Coerse of subtyping in ADAPTFUN)**.** If $\Delta \models \tau <: \tau'$, then exists e in the Core calculus, such that $\Delta; \bot; \cdot \vdash_0 e :^c \tau; 0 \xrightarrow{\bot;0} \tau'$.

Proof. By induction on the subtyping derivation.

$$\frac{}{\Delta \models \Box((\tau_1; q \xrightarrow{\rho';Z} \tau_2)) <: (\Box(\tau_1)); 0 \xrightarrow{\rho';0} (\Box(\tau_2))}\ \textbf{sb-box-arrow}$$

TS: exists e, $\Delta; \rho; \cdot \vdash_0 e :^c \Box((\tau_1; q \xrightarrow{\rho';Z} \tau_2)); 0 \xrightarrow{\bot;0} (\Box(\tau_1)); 0 \xrightarrow{\rho';0} (\Box(\tau_2))$
? $e = \lambda x.\lambda y.\mathsf{box}((\mathsf{der}x)(\mathsf{der}y))$.

$$\frac{\Delta \models \tau_1 <: \tau_1' \qquad \Delta \models \tau_2 <: \tau_2'}{\Delta \models \tau_1 \times \tau_2 <: \tau_1' \times \tau_2'}\ \textbf{sb-pair}$$

$$\frac{}{\Delta \models \texttt{bool} <: \Box(\texttt{bool})}\ \textbf{sb-box-bool}$$

□

Lemma 16 (Reflexivity of algorithmic type equivalence)**.** Exists Φ,$\Delta \models \tau \equiv \tau \Rightarrow \Phi$ and $\Delta \models \Phi$ is provable.

Proof. By structural induction on the type τ.

Case: $\tau_1 \times \tau_2$
TS: $\Delta \models \tau_1 \times \tau_2 \equiv \tau_1 \times \tau_2 \Rightarrow \Phi$ and $\Delta \models \Phi$.
By induction on τ_1,we get: $\exists \Phi_1.\Delta \models \tau_1 \equiv \tau_1 \Rightarrow \Phi_1$ and $\Delta \models \Phi_1$ is provable.
By induction on τ_2,we get: $\exists \Phi_2.\Delta \models \tau_2 \equiv \tau_2 \Rightarrow \Phi_2$ and $\Delta \models \Phi_2$ is provable.
By the above statement and the rule **alg-eq-pair**, we conclude the following statement where $\Phi = \Phi_1 \wedge \Phi_2$.

$\Delta \models \tau_1 \times \tau_2 \equiv \tau_1 \times \tau_2 \Rightarrow \Phi$ and $\Delta \models \Phi$.

Case: $\tau_1; q \xrightarrow{\rho;Z} \tau_2$

TS: $\Delta \models \tau_1; q \xrightarrow{\rho;Z} \tau_2 \equiv \tau_1; q \xrightarrow{\rho;Z} \tau_2 \Rightarrow \Phi$ and $\Delta \models \Phi$.
By induction on τ_1,we get: $\exists \Phi_1.\Delta \models \tau_1 \equiv \tau_1 \Rightarrow \Phi_1$ and $\Delta \models \Phi_1$ is provable.
By induction on τ_2,we get: $\exists \Phi_2.\Delta \models \tau_2 \equiv \tau_2 \Rightarrow \Phi_2$ and $\Delta \models \Phi_2$ is provable.
By the above statement and the rule **alg-eq-arrow**, we conclude the following statement where $\Phi = \Phi_1 \wedge \Phi_2 \wedge q \leq q \wedge Z \leq Z \wedge \rho \leq \rho$.
$\Delta \models \tau_1; q \xrightarrow{\rho;Z} \tau_2 \equiv \tau_1; q \xrightarrow{\rho;Z} \tau_2 \Rightarrow \Phi$ and $\Delta \models \Phi$.

Case $\forall i ::^{\rho;Z} S.\tau$

TS: $\Delta \models \forall i \overset{\rho;Z}{::} S.\tau \equiv \forall i \overset{\rho;Z}{::} S.\tau \Rightarrow \Phi$ and $\Delta \models \Phi$.
By induction on τ where instantiated with $\Delta' = i, \Delta$, exists Φ' such that $\Delta' \models \tau \equiv \tau \Rightarrow \Phi'$ and $\Delta \models \Phi'$.
By the above statement and the rule **alg-eq-**$\forall$,we conclude the following statement where $\Phi = \forall I :: S.\Phi$.
$\Delta \models \forall i \overset{\rho;Z}{::} S.\tau \equiv \forall i \overset{\rho;Z}{::} S.\tau \Rightarrow \Phi$ and $\Delta \models \Phi$.

□

Theorem 17 (Soundness of algorithmic type equivalence)**.** If $\Delta \models \tau \equiv \tau' \Rightarrow \Phi$ and $\mathsf{FIV}(\tau, \tau') \subseteq \Delta$, and $\Delta \models \Phi$ is provable, then $\Delta \models \tau \equiv^c \tau'$.

Proof. By induction on the algorithmic type equivalence derivation.

Case: $\dfrac{}{\Delta \models \mathsf{b} \equiv \mathsf{b} \Rightarrow \top}$ **alg-eq-base**
Assume $\mathsf{FIV}(\mathsf{b})$, it is proved by the core equivalence rule **core-eq-base**.

Case: $\dfrac{\Delta \models \tau_1 \equiv \tau_1' \Rightarrow \Phi_1 \qquad \Delta \models \tau_2 \equiv \tau_2' \Rightarrow \Phi_2}{\Delta \models \tau_1 \times \tau_2 \equiv \tau_1' \times \tau_2' \Rightarrow \Phi_1 \wedge \Phi_2}$ **alg-eq-pair**
Assume $\mathsf{FIV}(\tau_1 \times \tau_2, \tau_1' \times \tau_2') \subseteq \Delta$ and $\Delta \models \Phi_1 \wedge \Phi_2$.
TS: $\Delta \models \tau_1 \times \tau_2 \equiv^c \tau_1' \times \tau_2'$.
From the assumption $\mathsf{FIV}(\tau_1 \times \tau_2, \tau_1' \times \tau_2') \subseteq \Delta$, we know that :
$\mathsf{FIV}(\tau_1, \tau_1') \subseteq \Delta$ and $\mathsf{FIV}(\tau_2, \tau_2') \subseteq \Delta$.
Similarly, we also infer from the assumption that :
$\Delta \models \Phi_1$ and $\Delta \models \Phi_2$.
By IH on the premises, we know that :
$\Delta \models \tau_1' \equiv^c$ and $\Delta \models \tau_2 \equiv^c \tau_2'$.
From the rule **core-eq-pair**, we conclude that : $\Delta \models \tau_1 \times \tau_2 \equiv^c \tau_1' \times \tau_2'$.

Case: $\dfrac{\Delta \models \tau_1 \equiv \tau_1' \Rightarrow \Phi_1 \qquad \Delta \models \tau_2 \equiv \tau_2' \Rightarrow \Phi_2}{\Delta \models \tau_1; q \xrightarrow{\rho;Z} \tau_2 \equiv \tau_1'; q' \xrightarrow{\rho';Z'} \tau_2' \Rightarrow \Phi_1 \wedge \Phi_2 \wedge q = q' \wedge Z = Z' \wedge \rho = \rho'}$ **alg-eq-arrow**
Assume $\mathsf{FIV}(\tau_1; q \xrightarrow{\rho;Z} \tau_2, \tau_1'; q' \xrightarrow{\rho';Z'} \tau_2') \subseteq \Delta$ and $\Delta \models \Phi_1 \wedge \Phi_2 \wedge q = q' \wedge Z = Z' \wedge \rho = \rho'$.
TS: $\Delta \models \tau_1; q \xrightarrow{\rho;Z} \tau_2 \equiv^c \tau_1'; q' \xrightarrow{\rho';Z'} \tau_2'$.
From the assumption we know that :
$\mathsf{FIV}(\tau_1, \tau_1') \subseteq \Delta$ and $\mathsf{FIV}(\tau_2, \tau_2') \subseteq \Delta$.
Similarly, we also infer from the assumption that :
$\Delta \models \Phi_1$ and $\Delta \models \Phi_2$ and $\Delta \models q = q' \wedge Z = Z' \wedge \rho = \rho'$.
By IH on the premises, we know that :
$\Delta \models \tau_1 \equiv^c \tau_1'$ and $\Delta \models \tau_2 \equiv^c \tau_2'$.
From the rule **core-eq-arrow**, we conclude that :

$$\Delta \models \tau_1; q \xrightarrow{\rho; Z} \tau_2 \equiv^c \tau_1'; q' \xrightarrow{\rho'; Z'} \tau_2'$$

.

$$\frac{}{\Delta \models \texttt{int}[I] \equiv \texttt{int}[I'] \Rightarrow I = I'} \textbf{ alg-eq-int}$$

Assume $\mathsf{FIV}(\texttt{int}[I], \texttt{int}[I']) \subseteq \Delta$ and $\Delta \models I = I'$.
TS: $\Delta \models \texttt{int}[I] \equiv^c \texttt{int}[I']$.
By the assumption and the rule **core-eq-int**, we conclude that :
$\Delta \models \texttt{int}[I] \equiv^c \texttt{int}[I']$.

$$\frac{\Delta \models \tau \equiv \tau' \Rightarrow \Phi}{\Delta \models \Box(\tau) \equiv \Box(\tau') \Rightarrow \Phi} \textbf{ alg-eq-box}$$

Assume $\mathsf{FIV}(\Box(\tau), \Box(\tau')) \subseteq \Delta$ and $\Delta \models \Phi$.
TS: $\Delta \models \Box(\tau) \equiv^c \Box(\tau')$.
From the assumption we know that :
$\mathsf{FIV}(\tau, \tau') \subseteq \Delta$.
By IH on the premise, we know that :
$\Delta \models \tau \equiv^c \tau'$
From the rule **core-eq-box**, we conclude that :
$\Delta \models \Box(\tau) \equiv^c \Box(\tau')$.

$$\frac{i, \Delta \models \tau \equiv \tau' \Rightarrow \Phi}{\Delta \models \forall i ::^{\rho; Z} S.\tau \equiv \forall i ::^{\rho'; Z'} S.\tau' \Rightarrow \forall I :: S.\Phi \wedge \rho = \rho' \wedge Z = Z'} \textbf{ alg-eq-}\forall$$

Assume $\mathsf{FIV}(\forall i ::^{\rho; Z} S.\tau, \forall i ::^{\rho'; Z'} S.\tau') \subseteq \Delta$ and $\Delta \models \forall I :: S.\Phi \wedge \rho = \rho' \wedge Z = Z'$.
TS: $\Delta \models \forall i ::^{\rho; Z} S.\tau \equiv^c \forall i ::^{\rho'; Z'} S.\tau'$.
From the assumption we know that :
$\mathsf{FIV}(\tau, \tau', i) \subseteq (i, \Delta)$ and $i, \Delta \models \Phi$

By IH on the premise where $\Delta' = i, \Delta$, we know that :
$\Delta' \models \tau \equiv^c \tau'$
BY the above statements and the rule **core-eq-**$\forall$, we conclude that :
$\Delta \models \forall i ::^{\rho; Z} S.\tau \equiv^c \forall i ::^{\rho'; Z'} S.\tau'$.

□

Theorem 18 (Completeness of the algorithmic type equivalence)**.** If $\Delta \models \tau \equiv^c \tau'$, then exists Φ such that $\Delta \models \tau \equiv \tau' \Rightarrow \Phi$ and $\Delta \models \Phi$.

Proof. By induction on the type equivalence derivation.

$$\frac{\Delta \models \tau_1 \equiv^c \tau_1' \qquad \Delta \models \tau_2 \equiv^c \tau_2'}{\Delta \models \tau_1 \times \tau_2 \equiv^c \tau_1' \times \tau_2'}\ \textbf{core-eq-pair}$$

TS: Exists Φ such that $\Delta \models \tau_1 \times \tau_2 \equiv \tau_1' \times \tau_2' \Rightarrow \Phi$ and $\Delta \models \Phi$. By IH on the premises, we know that
Exists Φ_1 st $\Delta \models \tau_1 \equiv \tau_1' \Rightarrow \Phi_1$ and $\Delta \models \Phi_1$.
Exists Φ_2 st $\Delta \models \tau_2 \equiv \tau_2' \Rightarrow \Phi_2$ and $\Delta \models \Phi_2$.
By the above statements and the algorithmic type equivalence rule **alg-eq-pair**, we conclude the following statement where $\Phi = \Phi_1 \wedge \Phi_2$.
$\Delta \models \tau_1 \times \tau_2 \equiv \tau_1' \times \tau_2' \Rightarrow \Phi$ and $\Delta \models \Phi$.

$$\frac{\Delta \models \tau_2 \equiv^c \tau_2' \qquad \Delta \models \tau_1 \equiv^c \tau_1' \qquad \Delta \models q = q' \qquad \Delta \models Z = Z' \qquad \Delta \models \rho = \rho'}{\Delta \models \tau_1; q \xrightarrow{\rho; Z} \tau_2 \equiv^c \tau_1'; q' \xrightarrow{\rho'; Z'} \tau_2'}\ \textbf{core-eq-arrow}$$

TS: Exists Φ such that $\Delta \models \tau_1; q \xrightarrow{\rho; Z} \tau_2 \equiv \tau_1'; q' \xrightarrow{\rho'; Z'} \tau_2' \Rightarrow \Phi$ and $\Delta \models \Phi$.
By IH on the premises, we know that
Exists Φ_1 st $\Delta \models \tau_1 \equiv \tau_1' \Rightarrow \Phi_1$ and $\Delta \models \Phi_1$.
Exists Φ_2 st $\Delta \models \tau_2 \equiv \tau_2' \Rightarrow \Phi_2$ and $\Delta \models \Phi_2$.
$\Delta \models q = q' \wedge Z = Z' \wedge \rho = \rho'$.
By the above statements and the algorithmic type equivalence rule **alg-eq-arrow**, we conclude the following statement where $\Phi = \Phi_1 \wedge \Phi_2 \wedge q = q' \wedge Z = Z' \wedge \rho = \rho'$.
$\Delta \models \tau_1; q \xrightarrow{\rho; Z} \tau_2 \equiv \tau_1'; q' \xrightarrow{\rho'; Z'} \tau_2' \Rightarrow \Phi$ and $\Delta \models \Phi$.

$$\frac{\Delta \models I = I'}{\Delta \models \texttt{int}[I] \equiv^c \texttt{int}[I']}\ \textbf{core-eq-int}$$

TS: Exists Φ such that $\Delta \models \texttt{int}[I] \equiv \texttt{int}[I'] \Rightarrow \Phi$ and $\Delta \models \Phi$.
By the algorithmic type equivalence rule **alg-eq-int**, we conclude the following statement where $\Phi = I = I'$.
$\Delta \models \texttt{int}[I] \equiv \texttt{int}[I'] \Rightarrow \Phi$ and $\Delta \models \Phi$.

$$\frac{i, \Delta \models \tau \equiv^c \tau' \qquad \Delta \models Z = Z' \qquad \Delta \models \rho = \rho'}{\Delta \models \forall i \overset{\rho; Z}{::} S.\tau \equiv^c \forall i \overset{\rho'; Z'}{::} S.\tau'}\ \textbf{core-eq-}\forall$$

TS: Exists Φ such that $\Delta \models \forall i \overset{\rho; Z}{::} S.\tau \equiv \forall i \overset{\rho'; Z'}{::} S.\tau' \Rightarrow \Phi$ and $\Delta \models \Phi$.
By IH on the first premise where $\Delta' = i, \Delta$, we know that
Exists Φ_1 st $\Delta' \models \tau \equiv \tau' \Rightarrow \Phi_1$ and $\Delta' \models \Phi_1$.
From the statement$\Delta' \models \Phi_1$, we conclude the following statement by generalizing the index variable i:
$\Delta \models \forall i.\Phi_1$.
From the other premises, we get:
$\Delta \models Z = Z' \wedge \rho = \rho'$.
By the above statements and the algorithmic type equivalence rule **alg-eq-**$\forall$,

we conclude the following statement where $\Phi = \forall i.\Phi_1 \wedge Z = Z' \wedge \rho = \rho'$.
$\Delta \models \forall i \overset{\rho;Z}{::} S.\tau \equiv \forall i \overset{\rho';Z'}{::} S.\tau' \Rightarrow \Phi$ and $\Delta \models \Phi$.

□

Theorem 19 (Type are preserved via embedding)**.** If $\Delta;\Gamma;\rho \vdash_Z e \rightsquigarrow e' : \tau$, then $\Delta;\Gamma;\rho \vdash_Z e^* :^c \tau$ and $\Delta;\Gamma;\rho \vdash_Z e : \tau$.

Proof. By induction on the embedding derivation.

Case:

$$\frac{\Gamma(x) = \tau \qquad 0 \leq \rho(x) \textbf{ or equiv. } \rho(x) \neq \bot}{\Delta;\Gamma;\rho \vdash_Z x \rightsquigarrow x : \tau} \textbf{ e-var}$$

By the rule **var**, we know: $\Delta;\Gamma;\rho \vdash_Z x : \tau$.
By the rule **c-var**, we know: $\Delta;\Gamma;\rho \vdash_Z x :^c \tau$.

Case:

$$\frac{\begin{array}{c}\Delta;\Gamma;\rho_1 \vdash_{Z_1} e_1 \rightsquigarrow e_1^* : (\tau_1; q \xrightarrow{\rho;Z} \tau_2)\ (\star) \qquad \Delta;\Gamma;\rho_2 \vdash_{Z_2} e_2 \rightsquigarrow e_2^* : \tau_1\ (\diamond) \\ Z' = Z_1 + \max(Z, Z_2 + q) \qquad \rho' = \max(\rho_1, Z_1 + \max(\rho, \rho_2 + q))\end{array}}{\Delta;\Gamma;\rho' \vdash_{Z'} e_1\ e_2 \rightsquigarrow e_1^*\ e_2^* : \tau_2} \textbf{ e-app}$$

TS: $\Delta;\Gamma;\rho' \vdash_{Z'} e_1^*\ e_2^* :^c \tau_2$ and $\Delta;\Gamma;\rho' \vdash_{Z'} e_1\ e_2 : \tau_2$.
By induction on $(\star)$, we get:$\Delta;\Gamma;\rho_1 \vdash_{Z_1} e_1^* :^c (\tau_1; q \xrightarrow{\rho;Z} \tau_2)$ and $\Delta;\Gamma;\rho_1 \vdash_{Z_1} e_1 : (\tau_1; q \xrightarrow{\rho;Z} \tau_2)$.
By induction on $(\diamond)$, we get:$\Delta;\Gamma;\rho_2 \vdash_{Z_2} e_2^* :^c \tau_1$ and $\Delta;\Gamma;\rho_2 \vdash_{Z_2} e_2 : \tau_1$.
By the above statements, the premises, and the rule **app**, we conclude that : $\Delta;\Gamma;\rho' \vdash_{Z'} e_1\ e_2 : \tau_2$.
By the above statements, the premises, and the rule **c-app**, we conclude that : $\Delta;\Gamma;\rho' \vdash_{Z'} e_1^*\ e_2^* :^c \tau_2$.

Case:

$$\frac{\Delta;\Gamma, f : (\tau_1; q \xrightarrow{\rho;Z} \tau_2), x : \tau_1; \rho[f : \infty, x : q] \vdash_Z e \rightsquigarrow e^* : \tau_2}{\Delta;\Gamma;\rho' \vdash_{Z'} \texttt{fix}\, f(x : \tau_1).e \rightsquigarrow \texttt{fix}\, f(x : \tau_1).e^* : (\tau_1; q \xrightarrow{\rho;Z} \tau_2)} \textbf{ e-fix}$$

TS: $\Delta;\Gamma;\rho' \vdash_{Z'} \texttt{fix}\, f(x : \tau_1).e^* :^c (\tau_1; q \xrightarrow{\rho;Z} \tau_2)$ and $\Delta;\Gamma;\rho' \vdash_{Z'} \texttt{fix}\, f(x : \tau_1).e : (\tau_1; q \xrightarrow{\rho;Z} \tau_2)$.
By induction on the premise instantiated with $\Gamma, f : (\tau_1; q \xrightarrow{\rho;Z} \tau_2), x : \tau_1$ and $\rho[f : \infty, x : q]$, we get:
$\Delta;\Gamma, f : (\tau_1; q \xrightarrow{\rho;Z} \tau_2), x : \tau_1; \rho[f : \infty, x : q] \vdash_Z e^* :^c \tau_2$

$\Delta; \Gamma, f : (\tau_1; q \xrightarrow{\rho; Z} \tau_2), x : \tau_1; \rho[f : \infty, x : q] \vdash_Z e : \tau_2$
By the above statements, the premises, and the rule **fix**, we conclude that : $\Delta; \Gamma; \rho' \vdash_{Z'} \texttt{fix}\, f(x : \tau_1).e^* :^c (\tau_1; q \xrightarrow{\rho; Z} \tau_2)$.
By the above statements, the premises, and the rule **c-app**, we conclude that : $\Delta; \Gamma; \rho' \vdash_{Z'} \texttt{fix}\, f(x : \tau_1).e : (\tau_1; q \xrightarrow{\rho; Z} \tau_2)$.

Case:

$$\frac{\Delta \models \tau' <: \tau\ (\diamond) \qquad \begin{array}{c}\Delta; \Gamma; \rho' \vdash_{Z'} e \rightsquigarrow e^* : \tau'\ (\star) \\ \rho' \leq \rho\ (\clubsuit) \qquad Z' \leq Z\ (\heartsuit) \qquad e' = \mathsf{coerce}_{\tau', \tau}\ (\spadesuit)\end{array}}{\Delta; \Gamma; \rho \vdash_Z e \rightsquigarrow e'\ e^* : \tau}\ \textbf{e-subsumption}$$

TS: $\Delta; \Gamma; \rho \vdash_Z e'\ e^* :^c \tau$ and $\Delta; \Gamma; \rho \vdash_Z e : \tau$.
By induction on $(\star)$, we get:
$\Delta; \Gamma; \rho' \vdash_{Z'} e^* :^c \tau'$ (1) and $\Delta; \Gamma; \rho' \vdash_{Z'} e : \tau'$ (2) .
By (2),$(\diamond)$, $(\clubsuit)$, $(\heartsuit)$ and the rule **subtype**, we conclude that ;

$$\Delta; \Gamma; \rho \vdash_Z e : \tau$$

By Lemma 15 and $(\diamond)$, we have : $\Delta; \cdot; \bot \vdash_0 \mathsf{coerce}_{\tau', \tau} :^c \tau'; 0 \xrightarrow{\bot; 0} \tau$
From the rule **c-app**, we know:

$$\frac{\begin{array}{c}\Delta; \cdot; \bot \vdash_0 \mathsf{coerce}_{\tau', \tau} :^c \tau'; 0 \xrightarrow{\bot; 0} \tau \qquad \Delta; \Gamma; \rho' \vdash_{Z'} e^* :^c \tau' \\ Z' = 0 + \max(0, Z' + 0) \qquad \rho' = \max(0, 0 + \max(\bot, \rho' + 0))\end{array}}{\Delta; \Gamma; \rho' \vdash_{Z'} \mathsf{coerce}_{\tau', \tau}\ e^* :^c \tau}\ \textbf{c-app}$$

By the reflexivity of type equivalence, we know: $\Delta \models \tau \equiv^c \tau$.
By the above statements and $(\clubsuit)$, $(\heartsuit)$ and the rule **c-**$\equiv$, we conclude that:

$$\frac{\Delta; \Gamma; \rho' \vdash_{Z'} \mathsf{coerce}_{\tau', \tau}\ e^* :^c \tau \qquad \rho' < \rho \qquad Z' < Z \qquad \Delta \models \tau \equiv^c \tau}{\Delta; \Gamma; \rho \vdash_Z e :^c : \tau}\ \textbf{c-}\equiv$$

Case:

$$\frac{\begin{array}{c}\Delta; \Gamma; \rho \vdash_Z e \rightsquigarrow e^* : \square((\tau_1; 0 \xrightarrow{\rho''; 0} \tau_2)) \\ Z' = 1 + Z \qquad \rho' = 1 + \max(\rho, \rho'' + Z)\end{array}}{\Delta; \Gamma; \rho' \vdash_{Z'} \delta(e) \rightsquigarrow \delta(e^*) : \texttt{real}}\ \textbf{e-}\delta$$

TS: $\Delta; \Gamma; \rho' \vdash_{Z'} \delta(e^*) :^c \texttt{real}$ and $\Delta; \Gamma; \rho' \vdash_{Z'} \delta(e) : \texttt{real}$.
By induction on the first premise, we get: $\Delta; \Gamma; \rho \vdash_Z e^* :^c \square((\tau_1; 0 \xrightarrow{\rho''; 0} \tau_2))$ (1) and $\Delta; \Gamma; \rho \vdash_Z e : \square((\tau_1; 0 \xrightarrow{\rho''; 0} \tau_2))$ (2) .
By the above statement and the premises, and the rule δ, we conclude that: $\Delta; \Gamma; \rho' \vdash_{Z'} \delta(e) : \texttt{real}$.
By the above statement and the premises, and the rule **c-**δ, we conclude that: $\Delta; \Gamma; \rho' \vdash_{Z'} \delta(e^*) :^c \texttt{real}$.

Case:

$$\frac{\Delta;\Gamma;\rho \vdash_Z e \rightsquigarrow e^* : \tau \qquad \delta \notin e \qquad \forall x_i \in \mathtt{dom}(\Gamma).e_i = \mathsf{coerce}_{\Gamma(x_i),\Box\,(\Gamma(x_i))} \quad \forall x_i \in \mathtt{dom}(\Gamma).\Delta \models \Gamma(x_i) <: \Box\,(\Gamma(x_i))}{\Delta;\Gamma,\Gamma';\rho,\rho' \vdash_0 e \rightsquigarrow \mathtt{let}\ \overline{y_i; 0 = (e_i\ x_i)}\ \mathtt{in}(\mathsf{box}\ e^*[\overline{y_i/x_i}]) : \Box\,(\tau)}\ \textbf{e-box}$$

TS: $\Delta;\Gamma,\Gamma';\rho,\rho' \vdash_0 \mathtt{let}\ \overline{y_i; 0 = (e_i\ x_i)}\ \mathtt{in}(\mathsf{box}\ e^*[\overline{y_i/x_i}]) :^c \Box\,(\tau)$ and $\Delta;\Gamma,\Gamma';\rho,\rho' \vdash_0 e : \Box\,(\tau)$.
By induction on the first premise, we get: $\Delta;\Gamma;\rho \vdash_Z e^* :^c \tau$ (1) and $\Delta;\Gamma;\rho \vdash_Z e : \tau$ (2) .
By (2), the premises, and the rule **box**, we conclude that :
$\Delta;\Gamma,\Gamma';\rho,\rho' \vdash_0 e : \Box\,(\tau)$.
By the rule **c-let**

$$\frac{\Delta;\Gamma;\rho_1 \vdash_{Z_1} e_1 :^c \tau_1 \qquad \Delta;\Gamma,x:\tau_1;\rho_2[x:q] \vdash_{Z_2} e_2 :^c \tau_2 \quad \rho' = \max(\rho_2,\rho_1+q) \qquad Z' = \max(Z_2,Z_1+q)}{\Delta;\Gamma;\rho' \vdash_{Z'} \mathtt{let}\ x;q = e_1\ \mathtt{in}\ e_2 :^c \tau}\ \textbf{c-let}$$

By initiating with the goal we need to prove, we have:

$$\frac{\Delta;\Gamma,\Gamma';\rho_1 \vdash_0 e_i\ x_i :^c \Box\,(\Gamma(x_i)) \quad \Delta;\Box\,(\Gamma_1),\Gamma,\Gamma';\rho_2,\overline{[y_i:0]} \vdash_0 (\mathsf{box}\ e^*[\overline{y_i/x_i}]) :^c \Box\,(\tau) \quad \rho,\rho' = \max(\rho_2,\rho_1+0) \qquad Z' = \max(0,0+0)}{\Delta;\Gamma,\Gamma';\rho,\rho' \vdash_0 \mathtt{let}\ \overline{y_i; 0 = (e_i\ x_i)}\ \mathtt{in}(\mathsf{box}\ e^*[\overline{y_i/x_i}]) :^c \Box\,(\tau)}\ \textbf{c-let}$$

where $\mathtt{dom}(\rho_1) = dom(\Gamma,\Gamma')$, $\Gamma_1 = \overline{[y_i : \Gamma(x_i)]}$
STS:

$$\Delta;\Box\,(\Gamma_1),\Gamma,\Gamma';\rho_2,\overline{[y_i:0]} \vdash_0 (\mathsf{box}\ e^*[\overline{y_i/x_i}]) :^c \Box\,(\tau)$$

By using the **c-box**

$$\frac{\Delta;\Box\,(\Gamma);\rho \vdash_Z e :^c \tau \qquad \delta \notin e \qquad \mathtt{dom}(\Gamma') = \mathtt{dom}(\rho')}{\Delta;\Box\,(\Gamma),\Gamma';\rho,\rho' \vdash_0 \mathsf{box}\ e :^c \Box\,(\tau)}\ \textbf{c-box}$$

$$\frac{\Delta;\Box\,(\Gamma_1);\overline{[y_i:0]} \vdash_Z e^*[\overline{y_i/x_i}] :^c \tau \qquad \delta \notin e \qquad \mathtt{dom}(\Gamma,\Gamma') = \mathtt{dom}(\rho_2)}{\Delta;\Box\,(\Gamma_1),\Gamma,\Gamma';\rho_2,\overline{[y_i:0]} \vdash_0 (\mathsf{box}\ e^*[\overline{y_i/x_i}]) :^c \Box\,(\tau)}\ \textbf{c-box}$$

STS:

$$\Delta;\Box\,(\Gamma_1);\overline{[y_i:0]} \vdash_Z e^*[\overline{y_i/x_i}] :^c \tau$$

We extend the context and dmap from (1), we get :
$\Delta;\Box\,(\Gamma_1),\Gamma;\rho,\overline{[y_i:0]} \vdash_Z e^* :^c \tau$.
By substitution on all the variable x_i in Γ, we conclude that
$\Delta;\Box\,(\Gamma_1);\overline{[y_i:0]} \vdash_Z e^*[\overline{y_i/x_i}] :^c \tau$

Case:

$$\frac{\begin{array}{c}\Delta;\Gamma;\rho_1 \vdash_{Z_1} e_1 \rightsquigarrow e_1^* : \texttt{bool} \\ \Delta;\Gamma;\rho \vdash_Z e_2 \rightsquigarrow e_2^* : \tau \qquad \Delta;\Gamma;\rho \vdash_Z e_3 \rightsquigarrow e_3^* : \tau \\ Z' = Z_1 + Z \qquad \rho' = \max(\rho_1, Z_1 + \rho)\end{array}}{\Delta;\Gamma;\rho' \vdash_{Z'} \texttt{if}(e_1, e_2, e_3) \rightsquigarrow \texttt{if}(e_1^*, e_2^*, e_3^*) : \tau}\ \textbf{e-if}$$

TS: $\Delta;\Gamma;\rho' \vdash_{Z'} \texttt{if}(e_1^*, e_2^*, e_3^*) :^c \tau$ and $\Delta;\Gamma;\rho' \vdash_{Z'} \texttt{if}(e_1, e_2, e_3) : \tau$.
By induction on the first premise, we get:
$\Delta;\Gamma;\rho_1 \vdash_{Z_1} e_1^* :^c \texttt{bool}$ and $\Delta;\Gamma;\rho_1 \vdash_{Z_1} e_1 : \texttt{bool}$
By induction on the second premise, we get:
$\Delta;\Gamma;\rho \vdash_Z e_2^* :^c \tau$ and $\Delta;\Gamma;\rho \vdash_Z e_2 : \tau$
By induction on the third premise, we get:
$\Delta;\Gamma;\rho \vdash_Z e_3^* :^c \tau$ and $\Delta;\Gamma;\rho \vdash_Z e_3 : \tau$
By the above statements and the rule **if**, we conclude that: $\Delta;\Gamma;\rho' \vdash_{Z'} \texttt{if}(e_1, e_2, e_3) : \tau$.
By the above statements and the rule **c-if**, we conclude that: $\Delta;\Gamma;\rho' \vdash_{Z'} \texttt{if}(e_1^*, e_2^*, e_3^*) :^c \tau$.

Case:

$$\frac{\begin{array}{c}\Delta;\Gamma;\rho_1 \vdash_{Z_1} e_1 \rightsquigarrow e_1^* : \tau_1 \qquad \Delta;\Gamma, x : \tau_1;\rho_2[x : q] \vdash_{Z_2} e_2 \rightsquigarrow e_2^* : \tau \\ \rho' = \max(\rho_2, \rho_1 + q) \qquad Z' = \max(Z_2, Z_1 + q)\end{array}}{\Delta;\Gamma;\rho' \vdash_{Z'} \texttt{let}\, x; q = e_1\, \texttt{in}\, e_2 \rightsquigarrow \texttt{let}\, x; q = e_1^*\, \texttt{in}\, e_2^* : \tau}\ \textbf{e-let}$$

TS: $\Delta;\Gamma;\rho' \vdash_{Z'} \texttt{let}\, x; q = e_1^*\, \texttt{in}\, e_2^* :^c \tau$ and $\Delta;\Gamma;\rho' \vdash_{Z'} \texttt{let}\, x; q = e_1\, \texttt{in}\, e_2 : \tau$.
By induction on the first premise, we get:
$\Delta;\Gamma;\rho_1 \vdash_{Z_1} e_1^* :^c \tau_1$ and $\Delta;\Gamma;\rho_1 \vdash_{Z_1} e_1 : \tau_1$
By induction on the second premise, we get:
$\Delta;\Gamma, x : \tau_1;\rho_2[x : q] \vdash_{Z_2} e_2^* :^c \tau$ and $\Delta;\Gamma, x : \tau_1;\rho_2[x : q] \vdash_{Z_2} e_2 : \tau$
By the above statements and the rule **let**, we conclude that: $\Delta;\Gamma;\rho' \vdash_{Z'} \texttt{let}\, x; q = e_1\, \texttt{in}\, e_2 : \tau$.
By the above statements and the rule **c-let**, we conclude that: $\Delta;\Gamma;\rho' \vdash_{Z'} \texttt{let}\, x; q = e_1^*\, \texttt{in}\, e_2^* :^c \tau$.

□

Theorem 20 (Completeness of embedding)**.** If $\Delta;\Gamma;\rho \vdash_Z e : \tau$, then $\exists e'$ such that $\Delta;\Gamma;\rho \vdash_Z e \rightsquigarrow e' : \tau$.

Proof. By induction on the typing derivation.

Case:

$$\frac{\Delta;\Gamma;\rho \vdash_Z e : \tau\ (\star) \qquad \forall x \in \mathtt{dom}(\Gamma), \Gamma(x) <: \square(\Gamma(x))\ (\diamond) \qquad \delta \notin e}{\Delta;\Gamma,\Gamma';\rho,\boxed{\rho'} \vdash_Z e : \square(\tau)}\ \textbf{box}$$

By induction on $(\star)$, we know that: $\exists e^*.\Delta;\Gamma;\rho \vdash_Z e \rightsquigarrow e^* : \tau$.
By Lemma15 on $(\diamond)$, we know : exists $e_i = \mathsf{coerce}_{\Gamma(x_i),\square(\Gamma(x_i))}$ for all variable x_i.
By the rule **e-box**, we conclude

$$\frac{\Delta;\Gamma;\rho \vdash_Z e \rightsquigarrow e^* : \tau \qquad \delta \notin e \qquad \forall x_i \in \mathtt{dom}(\Gamma).e_i = \mathsf{coerce}_{\Gamma(x_i),\square(\Gamma(x_i))} \qquad \forall x_i \in \mathtt{dom}(\Gamma).\Delta \models \Gamma(x_i) <: \square(\Gamma(x_i))}{\Delta;\Gamma,\Gamma';\rho,\rho' \vdash_0 e \rightsquigarrow \mathtt{let}\ \overline{y_i; 0 = (e_i\ x_i)}\ \mathtt{in}(\mathsf{box}\ e^*[\overline{y_i/x_i}]) : \square(\tau)}\ \textbf{e-box}$$

Case:

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

By induction on $(\star)$, we know that: $\exists e^*.\Delta;\Gamma;\rho' \vdash_{Z'} e \rightsquigarrow e^* : \tau'$.
By Lemma15 on $(\diamond)$, we know : exists $e_i = \mathsf{coerce}_{\tau',\tau}$.
By the rule **e-subsumption**, we conclude

$$\frac{\Delta;\Gamma;\rho' \vdash_{Z'} e \rightsquigarrow e^* : \tau'\ (\star) \qquad \Delta \models \tau' <: \tau\ (\diamond) \qquad \rho' \leq \rho \qquad Z' \leq Z \qquad e' = \mathsf{coerce}_{\tau',\tau}}{\Delta;\Gamma;\rho \vdash_Z e \rightsquigarrow e'\ e^* : \tau}\ \textbf{e-subsumption}$$

Case:

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

By induction on $(\star)$, we know that: $\exists e_1^*.\Delta;\Gamma;\rho_1 \vdash_{Z_1} e_1 \rightsquigarrow e_1^* : (\tau_1; q \xrightarrow{\rho;Z} \tau_2)$.
By induction on $(\diamond)$, we know that: $\exists e_2^*.\Delta;\Gamma;\rho_2 \vdash_{Z_2} e_2 \rightsquigarrow e_2^* : \tau_1$.
By the rule **e-app**, we conclude

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

Case

$$\frac{\Delta;\Gamma;\rho \vdash_Z e : \forall i \overset{\rho_1;Z_1}{::} S.\tau\ (\star) \qquad \Delta \vdash I :: S \qquad \rho' = \max(\rho, Z + \rho_1) \qquad Z' = Z_1[I/i] + Z}{\Delta;\Gamma;\rho' \vdash_{Z'} e\ [] : \tau[I/i]}\ \textbf{iapp}$$

By induction on $(\star)$, we know that: $\exists e^*.\Delta;\Gamma;\rho \vdash_{Z_1} e \rightsquigarrow e^* : \forall i ::^{\rho_1,Z_1} S.\tau$.
By the rule **e-iapp**, we conclude

$$\frac{\Delta;\Gamma;\rho \vdash_Z e \rightsquigarrow e^* : \forall i ::^{\rho_1,Z_1} S.\tau \qquad \Delta \vdash I :: S \qquad \rho' = \max(\rho, Z + \rho_1) \qquad Z' = Z_1[I/i] + Z}{\Delta;\Gamma;\rho' \vdash_{Z'} e\ [] \rightsquigarrow e^*\ [I] : \tau[I/i]}\ \textbf{e-iapp}$$

□

Theorem 21 (Invariant of algorithmic type checking)**.** []

1. If $\Delta;\Gamma;\rho \vdash_Z e \downarrow \tau \Rightarrow \Phi$ and $\mathsf{FIV}(\Gamma,\tau) \subseteq \mathtt{dom}(\Delta)$, then $\mathsf{FIV}(\rho, Z, \Phi) \in \mathtt{dom}(\Delta)$.
2. If $\Delta;\Gamma;\rho \vdash_Z e \uparrow \tau \Rightarrow \Phi$, $\mathsf{FIV}(\Gamma) \subseteq \mathtt{dom}(\Delta)$, then $\mathsf{FIV}(\rho, Z, \tau, \Phi) \in \mathtt{dom}(\Delta)$.

Theorem 22 (Soundness of algorithmic typechecking)**.** []

1. If $\Delta;\Gamma;\rho \vdash_Z e \downarrow \tau \Rightarrow \Phi$ and $\mathsf{FIV}(\Gamma,\tau) \subseteq \mathtt{dom}(\Delta)$ and $\Delta \models \Phi$, then $\Delta;\Gamma;\rho \vdash_Z |e| :^c \tau$.
2. If $\Delta;\Gamma;\rho \vdash_Z e \uparrow \tau \Rightarrow \Phi$, $\mathsf{FIV}(\Gamma) \subseteq \mathtt{dom}(\Delta)$ and $\Delta \models \Phi$, then $\Delta;\Gamma;\rho \vdash_Z |e| :^c \tau$.

Proof. By simultaneous induction on the algorithmic typing derivation.
Proof of statement (1).

Case:

$$\frac{\Delta;\Box(\Gamma);\rho \vdash_Z e \downarrow \tau \Rightarrow \Phi \qquad \delta \notin e \qquad \forall x \in \mathtt{dom}(\Gamma').\rho'(x) = \bot \qquad \mathtt{dom}(\Gamma') = \mathtt{dom}(\rho')}{\Delta;\Gamma',\Box(\Gamma);\rho,\boxed{\rho'} \vdash_Z \mathsf{BOX}\ e \downarrow \Box(\tau) \Rightarrow \Phi}\ \textbf{alg-box-}\downarrow$$

The main assumption assumes $\mathsf{FIV}(\Gamma,\Gamma',\tau) \subseteq \mathtt{dom}(\Delta)$ (1) and $\Delta \models \Phi$ (2).
TS: $\Delta;\Gamma',\Box(\Gamma);\rho' \vdash_Z \mathsf{box}\ |e| :^c \Box(\tau)$
From (1), we infer that : $\mathsf{FIV}(\Gamma,\tau) \subseteq \mathtt{dom}(\Delta)$ (3).
By IH1 on the first premise using (2),(3), we know:
$\Delta;\Box(\Gamma);\rho \vdash_Z |e| :^c \tau$.
By the rule **c-box**,

$$\frac{\Delta;\Box(\Gamma);\rho \vdash_Z |e| :^c \tau \qquad \delta \notin e \qquad \mathtt{dom}(\Gamma') = \mathtt{dom}(\rho')}{\Delta;\Box(\Gamma),\Gamma';\rho,\rho' \vdash_0 \mathsf{box}\ |e| :^c \Box(\tau)}\ \textbf{c-box}$$

Case:

$$\frac{\Delta;\Gamma;\rho_1 \vdash_{Z_1} e_1 \uparrow \texttt{bool} \Rightarrow \Phi_1 \qquad \Delta;\Gamma;\rho_2 \vdash_{Z_2} e_2 \downarrow \tau \Rightarrow \Phi_2 \qquad \Delta;\Gamma;\rho_3 \vdash_{Z_3} e_3 \downarrow \tau \Rightarrow \Phi_3 \qquad Z' = Z_1 + \max(Z_2, Z_3) \qquad \rho' = \max(\rho_1, Z_1 + \max(\rho_2, \rho_3))}{\Delta;\Gamma;\rho' \vdash_{Z'} \texttt{if}(e_1, e_2, e_2) \downarrow \tau \Rightarrow \Phi_1 \wedge \Phi_2 \wedge \Phi_3}\ \textbf{alg-if-}\downarrow$$

The main assumption assumes $\mathsf{FIV}(\Gamma, \tau) \subseteq \texttt{dom}(\Delta)$ (1) and $\Delta \models \Phi_1 \wedge \Phi_2 \wedge \Phi_3$ (2).
TS: $\Delta;\Gamma;\rho' \vdash_{Z'} \texttt{if}(|e_1|, |e_2|, |e_2|) :^c \tau$
From (1), we infer that : $\mathsf{FIV}(\Gamma, \texttt{bool}) \subseteq \texttt{dom}(\Delta)$ (3) .
Similarly, we infer that $\mathsf{FIV}(\Gamma, \tau) \subseteq \texttt{dom}(\Delta)$ (4) .
From (2), we know that $\Delta \models \Phi_1$ (6), $\Delta \models \Phi_2$ (7) and $\Delta \models \Phi_3$ (8).
By IH2 on the first premise with (3),(6), we know:
$\Delta;\Gamma;\rho_1 \vdash_{Z_1} |e_1| :^c \texttt{bool}$ (a)
By IH1 on the second premise with (4),(7), we know:
$\Delta;\Gamma;\rho_2 \vdash_{Z_2} |e_2| :^c \tau$ (b)
By IH1 on the third premise with (4),(8), we know:
$\Delta;\Gamma;\rho_3 \vdash_{Z_3} |e_3| :^c \tau$ (c)
By the rule **c-**$\equiv$ on the (b), (c), we get:
$\Delta;\Gamma;\max(\rho_2, \rho_3) \vdash_{\max(Z_2,Z_3)} |e_2| :^c \tau$ $(\star)$
$\Delta;\Gamma;\max(\rho_2, \rho_3) \vdash_{\max(Z_2,Z_3)} |e_3| :^c \tau$ $(\diamond)$
By the rule **c-if** with $(a), (\star), (\diamond)$, we conclude that :

$$\frac{\Delta;\Gamma;\rho_1 \vdash_{Z_1} |e_1| :^c \texttt{bool}\ (a) \qquad \Delta;\Gamma;\max(\rho_2, \rho_3) \vdash_{\max(Z_2,Z_3)} |e_2| :^c \tau\ (\star) \qquad \Delta;\Gamma;\max(\rho_2, \rho_3) \vdash_{\max(Z_2,Z_3)} |e_3| :^c \tau\ (\diamond) \qquad Z' = Z_1 + \max(Z_2, Z_3) \qquad \rho' = \max(\rho_1, Z_1 + \max(\rho_2, \rho_3))}{\Delta;\Gamma;\rho' \vdash_{Z'} \texttt{if}(|e_1|, |e_2|, |e_3|) :^c \tau}\ \textbf{c-if}$$

Case:

$$\frac{\Delta;\Gamma;\rho_1 \vdash_{Z_1} e_1 \downarrow \tau \Rightarrow \Phi_1 \qquad \Delta;\Gamma;\rho_2 \vdash_{Z_2} e_2 \downarrow \texttt{list}\,\tau \Rightarrow \Phi_2 \qquad Z = \max(Z_1, Z_2) \qquad \rho = \max(\rho_1, \rho_2)}{\Delta;\Gamma;\rho \vdash_{Z} \texttt{cons}(e_1, e_2) \downarrow \texttt{list}\,\tau \Rightarrow \Phi_1 \wedge \Phi_2}\ \textbf{alg-cons-}\downarrow$$

The main assumption assumes $\mathsf{FIV}(\Gamma, \tau) \subseteq \texttt{dom}(\Delta)$ (1) and $\Delta \models \Phi_1 \wedge \Phi_2$ (2).
TS: $\Delta;\Gamma;\rho \vdash_{Z} \texttt{cons}(|e_1|, |e_2|) :^c \texttt{list}\,\tau$
From (1), we infer that : $\mathsf{FIV}(\Gamma, \tau) \subseteq \texttt{dom}(\Delta)$ (3) .
From (2), we know that $\Delta \models \Phi_1$ (5), $\Delta \models \Phi_2$ (6).
By IH1 on the first premise with (3),(5), we know:
$\Delta;\Gamma;\rho_1 \vdash_{Z_1} |e_1| :^c \tau$ (a)
By IH1 on the second premise with (3),(6), we know:
$\Delta;\Gamma;\rho_2 \vdash_{Z_2} |e_2| :^c \texttt{list}\,\tau$ (b)

By the rule **c-cons** ,we conclude that :

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

Case:

$$\frac{\Delta;\Gamma;\rho_1 \vdash_{Z_1} e_1 \uparrow \tau_1 \Rightarrow \Phi \qquad \Delta;\Gamma,x:\tau_1;\rho_2 \vdash_{Z_2} e_2 \downarrow \tau \Rightarrow \Phi_2 \qquad Z = \max(Z_2, Z_1+q) \qquad \rho = \max(\rho_2 \setminus x, \rho_1 + q)}{\Delta;\Gamma;\rho \vdash_{Z} \texttt{let}\, x : q = e_1 \,\texttt{in}\, e_2 \downarrow \tau \Rightarrow \Phi_1 \wedge \Phi_2 \wedge \rho_2(x) \leq q}\ \textbf{alg-let-}\downarrow$$

The main assumption assumes $\mathsf{FIV}(\Gamma,\tau) \subseteq \texttt{dom}(\Delta)$ (1) and $\Delta \models \Phi_1 \wedge \Phi_2 \wedge \rho_2(x) \leq q$ (2).
TS: $\Delta;\Gamma;\rho \vdash_Z \texttt{let}\, x;q = |e_1|\,\texttt{in}\,|e_2| :^c \tau$
From (1), we infer that : $\mathsf{FIV}(\Gamma,\tau_1) \subseteq \texttt{dom}(\Delta)$ (3).
Similarly, we infer that $\mathsf{FIV}(\Gamma, x:\tau_1, \tau) \subseteq \texttt{dom}(\Delta)$ (4)
From (2), we know that $\Delta \models \Phi_1$ (5), $\Delta \models \Phi_2$ (6) and $\Delta \models \rho_2(x) \leq q$ (7).
By IH2 on the first premise with (3),(5), we know:
$\Delta;\Gamma;\rho_1 \vdash_{Z_1} |e_1| :^c \tau_1$ (a)
By IH1 on the second premise with (4),(6), we know:
$\Delta;\Gamma;\rho_2 \vdash_{Z_2} |e_2| :^c \tau$ (b)
Set $\rho_2 = \rho_2', [x:\rho_2(x)]$ and $\rho_2' = \rho_2 \setminus x$.
We rewrite (b) as follows:
$\Delta;\Gamma;\rho_2', [x:\rho_2(x)] \vdash_{Z_2} |e_2| :^c \tau$ (c)
By the rule **c-**$\equiv$ and $\rho_2(x) \leq q$ on (c), we conclude that:

$$\Delta;\Gamma;\rho_2', [x:q] \vdash_{Z_2} |e_2| :^c \tau \ (d)$$

BY the rule **c-let**, we conclude the following statement.

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

Case:

$$\frac{\Delta;\Gamma;\rho \vdash_Z e \uparrow \tau \Rightarrow \Phi \qquad \Delta \models \tau \equiv \tau' \Rightarrow \Phi'}{\Delta;\Gamma;\rho \vdash_Z e \downarrow \tau' \Rightarrow \Phi \wedge \Phi'}\ \textbf{alg-}\uparrow\textbf{-}\downarrow$$

The main assumption assumes $\mathsf{FIV}(\Gamma,\tau') \subseteq \texttt{dom}(\Delta)$ (1) and $\Delta \models \Phi_1 \wedge \Phi'$ (2).
TS: $\Delta;\Gamma;\rho \vdash_Z |e| :^c \tau'$
From (1), we infer that : $\mathsf{FIV}(\Gamma) \subseteq \texttt{dom}(\Delta)$ (3) .
From (2), we know that $\Delta \models \Phi$ (4), $\Delta \models \Phi'$ (5).

By IH2 on the first premise with (3),(4), we know:
$\Delta; \Gamma; \rho \vdash_Z |e| :^c \tau$ (a)
By the rule **c-≡**, we conclude that:

$$\frac{\Delta; \Gamma; \rho \vdash_Z |e| :^c \tau \qquad \rho \leq \rho \qquad Z \leq Z \qquad \Delta \models \tau \equiv^c \tau'}{\Delta; \Gamma; \rho \vdash_Z |e| :^c \tau'} \textbf{c-}\equiv$$

Case:

$$\frac{i :: S, \Delta; \Gamma; \rho \vdash_Z e \downarrow \tau \Rightarrow \Phi}{\Delta; \Gamma; \rho \vdash_0 \Lambda.e \downarrow \forall i \overset{\rho;Z}{::} S.\tau \Rightarrow \forall i.\Phi} \textbf{alg-iabs-}\downarrow$$

The main assumption assumes $\mathsf{FIV}(\Gamma, \forall i \overset{\rho;Z}{::} S.\tau) \subseteq \mathtt{dom}(\Delta)$ (1) and $\Delta \models \forall i.\Phi$ (2).

TS: $\Delta; \Gamma; \rho \vdash_0 \Lambda.|e| :^c \forall i \overset{\rho;Z}{::} S.\tau$
From (1), we can infer that : $i \notin \mathsf{FIV}(\Gamma)$. Can we ?
From (1), we infer that : $\mathsf{FIV}(\Gamma, \tau) \subseteq i, \mathtt{dom}(\Delta)$ (3) .
From (2), we know that $i :: S, \Delta \models \Phi$ (4)
By IH1 on the first premise with (3),(4), we know:
$i, \Delta; \Gamma; \rho \vdash_Z |e| :^c \tau$ (a)
By the rule **c-iLam**,we conclude :

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

Proof of statement (2).

Case:

$$\frac{\Delta; \Gamma; \rho \vdash_Z e \downarrow \tau \Rightarrow \Phi \qquad \Delta \vdash \tau \ \mathtt{wf} \qquad \mathsf{FIV}(\tau, \rho, Z) \in \mathtt{dom}(\Delta) \ (\star)}{\Delta; \Gamma; \rho \vdash_Z (e : \tau, \rho, Z) \uparrow \tau \Rightarrow \Phi} \textbf{alg-anno-}\uparrow$$

The main assumption assumes $\mathsf{FIV}(\Gamma) \subseteq \mathtt{dom}(\Delta)$ (1) and $\Delta \models \Phi$ (2).
TS: $\Delta; \Gamma; \rho \vdash_Z |(e : \tau, \rho, Z)| :^c \tau$
From (1) and $(\star)$, we infer that : $\mathsf{FIV}(\Gamma, \tau) \subseteq \mathtt{dom}(\Delta)$ (3) .
By IH1 on the first premise with (3),(2), we conclude:

$$\Delta; \Gamma; \rho \vdash_Z |e| :^c \tau$$

Case:

$$\frac{\begin{array}{c}\Delta; \Gamma; \rho \vdash_Z e \uparrow \forall i \overset{\rho_1;Z_1}{::} S.\tau \Rightarrow \Phi \qquad \Delta \vdash I :: S \\ \rho' = \max(\rho, Z + \rho_1) \qquad Z' = Z_1[I/i] + Z\end{array}}{\Delta; \Gamma; \rho' \vdash_{Z'} e[I] \uparrow \tau[I/i] \Rightarrow \Phi} \textbf{alg-iapp}\uparrow$$

The main assumption assumes $\mathsf{FIV}(\Gamma) \subseteq \mathtt{dom}(\Delta)$ (1) and $\Delta \models \Phi$ (2).
TS: $\Delta; \Gamma; \rho' \vdash_{Z'} |e[I]| :^c \tau[I/i]$
By IH2 on the first premise with (1),(2), we conclude:
$\Delta; \Gamma; \rho \vdash_Z |e| :^c \forall i \overset{\rho_1; Z_1}{::} S.\tau$ (a)

By the above statement and the premises, we conclude:

$$\frac{\Delta; \Gamma; \rho \vdash_Z |e| :^c \forall i \overset{\rho_1; Z_1}{::} S.\tau \qquad \Delta \vdash I :: S \qquad \rho' = \max(\rho, Z + \rho_1) \qquad Z' = Z_1[I/i] + Z}{\Delta; \Gamma; \rho' \vdash_{Z'} |e|\ [I] :^c \tau[I/i]}\ \textbf{c-iapp}$$

Case:

$$\frac{\Delta; \Gamma; \rho_1 \vdash_{Z_1} e_1 \uparrow \tau_1; q \xrightarrow{\boxed{\rho}; Z} \tau_2 \Rightarrow \Phi_1 \qquad \Delta; \Gamma; \rho_2 \vdash_{Z_2} e_2 \downarrow \tau_1 \Rightarrow \Phi_2 \qquad Z' = Z_1 + \max(Z, Z_2 + q) \qquad \rho' = \max(\rho_1, Z_1 + \max(\rho, \rho_2 + q))}{\Delta; \Gamma; \rho' \vdash_{Z'} e_1\ e_2 \uparrow \tau_2 \Rightarrow \Phi_1 \wedge \Phi_2}\ \textbf{alg-app-}\uparrow$$

The main assumption assumes $\mathsf{FIV}(\Gamma) \subseteq \mathtt{dom}(\Delta)$ (1) and $\Delta \models \Phi$ (2).
TS: $\Delta; \Gamma; \rho' \vdash_{Z'} |e_1|\ |e_2| :^c \tau_2$
By IH2 on the first premise with (1),(2), we conclude:
$\Delta; \Gamma; \rho_1 \vdash_{Z_1} |e_1| :^c (\tau_1; q \xrightarrow{\rho; Z} \tau_2)$ (a)
By theorem 21 on the first premise and (1), we know:

$$\mathsf{FIV}(\rho_1, Z_1, (\tau_1; q \xrightarrow{\rho; Z} \tau_2), \Phi_1) \in \mathtt{dom}(\Delta)\ (3)$$

From (1) and (3), we know: $\mathsf{FIV}(\Gamma, \tau_1) \in \mathtt{dom}(\Delta)$ (4).
By IH1 on the second premise vis (4), (2), we conclude:
$\Delta; \Gamma; \rho_2 \vdash_{Z_2} |e_2| :^c \tau_1$ (b)
By the above statements and premises, using the rule **c-app**, we conclude that

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

Case:

$$\frac{\Delta; \Gamma; \rho \vdash_Z e \uparrow \square\,(\tau_1; 0 \xrightarrow{\rho''; 0} \tau_2) \Rightarrow \Phi \qquad Z' = Z + 1 \qquad \rho' = 1 + \max(\rho, \rho'' + Z)}{\Delta; \Gamma; \rho' \vdash_{Z'} \delta e \uparrow \mathtt{real} \Rightarrow \Phi}\ \textbf{alg-}\delta\textbf{-}\uparrow$$

The main assumption assumes $\mathsf{FIV}(\Gamma) \subseteq \mathtt{dom}(\Delta)$ (1) and $\Delta \models \Phi$ (2).
TS: $\Delta; \Gamma; \rho' \vdash_{Z'} \delta|e| :^c \mathtt{real}$

$$\frac{\Delta; \rho \vdash \tau\ \mathtt{wf}}{\Delta; \Gamma; \rho \vdash_Z \mathtt{nil} :^c \mathtt{list}\,\tau}\ \textbf{c-nil} \qquad \frac{\Delta; \Gamma; \rho \vdash_Z e :^c \Box\,((\tau_1; 0 \xrightarrow{\rho''; 0} \tau_2)) \quad Z' = 1 + Z \quad \rho' = 1 + \max(\rho, \rho'' + Z)}{\Delta; \Gamma; \rho' \vdash_{Z'} \delta(e) :^c \mathtt{real}}\ \textbf{c-}\delta$$

Case

$$\frac{\Delta; \Gamma; \rho \vdash_Z e \uparrow \tau_1 \times \tau_2 \Rightarrow \Phi}{\Delta; \Gamma; \rho \vdash_Z \mathtt{fst}(e) \uparrow \tau_1 \Rightarrow \Phi}\ \textbf{alg-fst-}\uparrow$$

Case:

$$\frac{\Gamma(x) = \tau \qquad \rho = [x : 0] \cup \bot}{\Delta; \Gamma; \rho \vdash_Z x \uparrow \tau \Rightarrow \top}\ \textbf{alg-var-}\uparrow$$

□

8 Examples Detail

9 Examples

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 $q_k(x) = x(k) \cdot \text{sign}\big(\sum_{i\in[k]} x(i) \cdot \ln\frac{1+a_i}{1-a_i}\big)$ where $x \in \{-1,+1\}^{k+1}$.
  {In the line above, $\text{sign}(y) = \begin{cases} +1 & \text{if } y \geq 0 \\ -1 & \text{otherwise} \end{cases}$.}
  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: $a_{k+1} \in [-1,+1]$

Two-rounds:

$$
\begin{array}{l}
\texttt{let } g : \bot = \texttt{fix } f(j : \texttt{int}).\lambda k : \texttt{int}.\lambda db : \texttt{tdb}. \\
\quad \texttt{if}\big((j < k), \\
\qquad \texttt{let } a : 0 = \delta\ (\lambda x : \texttt{list int}.(\mathsf{get}\ x\ j) \cdot (\mathsf{get}\ x\ k))\ \texttt{in} \\
\qquad\quad (a, j) :: (f\ (j+1)\ k) \\
\qquad\quad , [])\ \texttt{in} \\
\quad \texttt{fix } \mathsf{twoRound}(k : \texttt{int}).\lambda db : \texttt{list list int}. \\
\qquad \texttt{let } l : 1 = g\ 0\ k\ bd\ \texttt{in} \\
\qquad \texttt{let } q : 1 = \lambda x : \texttt{list int}.\mathsf{sign} \\
\qquad (\mathsf{foldl}\ (\lambda acc : \texttt{real}.\lambda ai : \texttt{real} \cdot \texttt{int}.\big(acc\ + (\mathsf{get}\ x\ (\mathsf{snd}\ ai)) \cdot \mathsf{log}(\tfrac{1+(\mathsf{fst}\ ai)}{1-(\mathsf{fst}\ ai)})\big)\ 0.0\ l))\ \texttt{in} \\
\qquad \delta(q)
\end{array}
$$

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

Require: Mechanism $\mathcal{M}$ with a hidden state $X \in [N]^n$ 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 $p = \texttt{uniform}(0, 1)$

define $q(x) = \texttt{bernoulli}(p)$.

define $qc(x) = \texttt{bernoulli}(p)$.

let $a = \mathcal{M}(q)$

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 $X = X \setminus I$

return X.

```
    let updtSC =
    fix f(z : unit).λsc : list real.λa : real.λp : real.λq : □(int → int).
    λI : list ∫.λi : int.λn : int.
     if((i < n),
      if((in i I),
         let x : 0 = (get sc i) + (a − p) ∗ (q i − p) in
         let sc′ : 0 = updt sc i x in
           f () sc′ a p q I (i + 1) n
      , f () sc a p q I (i + 1) n)
     , sc) in

    let updtSCC =
    fix f(z : unit).λscc : list real.λa : real.λp : real.λqc : □(int → int).
    λi : int.λcr : int.
     if((i < cr),
        let x : 0 = (nth scc i) + (a − p) ∗ (qc i − p) in
        let scc′ : 0 = updt scc i x in
          f () scc′ a p qc (i + 1) cr
     , scc) in

        let updtI =
        fix f(z : unit).λmaxScc : real.λsc : list real.λi : int.λn : int.
         if((i < n),
          if(((nth scc i) > maxScc),
             i :: (f () maxScc sc (i + 1) n)
             f () maxScc sc (i + 1) n)
         , []) in

fix multiRound(z : unit).Λk.Λj.λk : int[k].λj : int[j].λsc : list real.
λscc : list real.λil : list int.λn : int.λcr : int.λdb : list int.
 if((j < k),
 let p : k − j = uniform 0 1 in
  let q : k − j = λx.bernoulli p in
  let qc : k − j = λc.bernoulli p in
  let qj : k − j = restrict q db
  let a : k − j − 1 = δ(q) in
    let sc′ : k − j − 1 = updtSC () sc a p q il 0 n in
    let scc′ : k − j − 1 = updtSCC () scc a p qc 0 cr in
    let maxScc : k − j − 1 = foldl (λacc : real.λa : real.if(acc < a, a, acc)) 0 scc′ in
    let il′ : k − j − 1 = updtI () maxScc sc 0 n in
    let db′ : k − j − 1 = db \ il′ in
      a :: (multiRound () [k] [j + 1] k (j + 1) sc′ scc′ il′ n cr db′)
 , [])
```

10 vectors

$$
\begin{array}{lcll}
\max(\bot, q) & = & q & \\
\max(q, \bot) & = & q & \\
\max(\infty, q) & = & \infty & \\
\max(q, \infty) & = & \infty &
\end{array}
$$

$$
\begin{array}{lcll}
\bot + q & = & \bot & \\
q + \bot & = & \bot & \\
\infty + q & = & \infty & \text{if } q \neq \bot \\
q + \infty & = & \infty & \text{if } q \neq \bot
\end{array}
$$

$$
\begin{array}{l}
\bot \leq q \\
q \leq \infty
\end{array}
$$

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

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

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

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

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

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

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

$$\frac{\theta, e_1 \Downarrow v_1, T_1 \qquad \theta, e_2 \Downarrow v_2, T_2}{\theta, (e_1, e_2) \Downarrow (v_1, v_2), (T_1, T_2)} \qquad \frac{\theta, e \Downarrow (v_1, v_2), T}{\theta, \texttt{fst}(e) \Downarrow v_1, \texttt{fst}(T)}$$

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

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

$$\frac{}{\theta, \texttt{nil} \Downarrow \texttt{nil}, \texttt{nil}} \qquad \frac{\theta, e_1 \Downarrow v_1, T_1 \qquad \theta, e_2 \Downarrow v_2, T_2}{\theta, \texttt{cons}(e_1, e_2) \Downarrow \texttt{cons}(v_1, v_2), \texttt{cons}(T_1, T_2)}$$

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

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

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

Figure 22: Big-step semantics with provenance, vector

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

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

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

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

Figure 23: Adaptivity of a trace and depth of variable x in a trace

Two-rounds:

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

Algorithm 3 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 $q_{k+1}(x) = \text{sign}\big(\sum_{i \in [k]} x(i) \times \ln \frac{1+a_i}{1-a_i}\big)$ where $x \in \{-1, +1\}^{k+1}$.

{In the line above, $\text{sign}(y) = \begin{cases} +1 & \text{if } y \geq 0 \\ -1 & \text{otherwise} \end{cases}$.}

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 4 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 $p = \texttt{uniform}(0, 1)$

define $q(x) = \texttt{bernoulli}(p)$.

define $qc(x) = \texttt{bernoulli}(p)$.

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.

```
        let updtSC =
        fix f(z : unit).λsc : list real.λa : real.λp : real.λq.
        λI : list int.λi : int.λn : int.
         if((i < n),
          if((in i I),
             let x : 0 = (depth sc i) + (a − p) ∗ (q{i} − p) in
             let sc′ : 0 = updt sc i x in
               f () sc′ a p q I (i + 1) n
          , f () sc a p q I (i + 1) n)
         , sc) in

        let updtSCC =
        fix f(z : unit).λscc : list real.λa : real.λp : real.λqc
        λi : int.λcr : int.
         if((i < cr),
             let x : 0 = (nth scc i) + (a − p) ∗ (qc{i} − p) in
             let scc′ : 0 = updt scc i x in
               f () scc′ a p qc (i + 1) cr
         , scc) in

      let updtI =
      fix f(z : unit).λmaxScc : real.λsc : list real.λi : int.λn : int.
       if((i < n),
        if(((nth scc i) > maxScc),
          i :: (f () maxScc sc (i + 1) n)
          f () maxScc sc (i + 1) n)
       , []) in

fix multiRound(z : unit).Λk.Λj.λk : int[k].λj : int[j].λsc : list real.
λscc : list real.λil : list int.λn : int.λcr : int.λd : list int.
 if((j < k),
 let p : k − j = uniform 0 1 in
  let q : k − j = dict(attr_i → bernoulli p^{i∈1...n}) in
  let qc : k − j = dict(attr_i → bernoulli p^{i∈1...n}) in
  let qj : k − j = dict(attr_i → if((in e_i d), q{e_i}, else 0)^{i∈1...n}) in
  let a : k − j − 1 = δ(q_j) in
    let sc′ : k − j − 1 = updtSC () sc a p qj il 0 n in
    let scc′ : k − j − 1 = updtSCC () scc a p qc 0 cr in
    let maxScc : k − j − 1 = foldI (λacc : real.λa : real.if(acc < a, a, acc)) 0 scc′ in
    let il′ : k − j − 1 = updtI () maxScc sc 0 n in
    let d′ : k − j − 1 = d \ il′ in
      a :: (multiRound () [k] [j + 1] k (j + 1) sc′ scc′ il′ n cr d′)
 , [])
```

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

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

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

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

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

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

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

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

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

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

Figure 24: Typing rules, part 1

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

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

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

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

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

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

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

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

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

Figure 25: 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 \mathrm{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 26: Logical relation with step-indexing

Theorem 23 (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 $\dfrac{\Delta; \Gamma, f : (\tau_1; \xrightarrow{q;\, Z} \tau_2), x : \tau_1; \rho, [x : q] \vdash_Z e : \tau_2 \ (1)}{\Delta; \Gamma; \rho \vdash_0 \texttt{fix}\, f(x).e : \tau_1; \xrightarrow{q;\, Z} (\tau_2)}\ \textbf{fix}$

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. $(\forall x \in \mathrm{Vars}.\texttt{depth}_x(T) \leq \rho(x))$;
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. $\texttt{adap}(\sigma\,\texttt{fix}\,f(x).e) = 0 \le 0$;
2. $\forall x \in \text{Vars}.\texttt{depth}_x((\sigma\,\texttt{fix}\,f(x).e, \theta)) = \bot \le \rho(x)$;
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 $k - j = k'$, $\forall m \le k', (m, (\sigma\,\texttt{fix}\,f(x).e, \theta)) \in [\![\sigma(\tau_1; \xrightarrow{q;Z} \tau_2)]\!]_{\text{V}}$.
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: $m = m' + 1 \le k'$,
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: $(\theta[x \mapsto v_m, f \mapsto (\sigma\,\texttt{fix}\,f(x).e, \theta)], \sigma e) \Downarrow v, T \wedge |T| = j'' \wedge \texttt{adap}(T) \le \sigma Z \wedge \texttt{depth}_x(T) \le q \wedge (j' - j'', v) \in [\![\sigma\tau_2]\!]_{\text{V}}$ (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:

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

proved by:

a) $(k, \theta) \in [\![\Gamma]\!]_{\text{V}}$, applying Lemma 13 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: $(\theta[x \mapsto v_m, f \mapsto (\sigma\,\texttt{fix}\,f(x).e, \theta)], \sigma e) \Downarrow v, T \wedge |T| = j'' \le j' \wedge \texttt{adap}(T) \le \sigma Z \wedge \forall x \in \text{Vars}.\ \texttt{depth}_x(T) \le \rho[x:q](x) \wedge (j' - j'', v) \in [\![\sigma\tau_2]\!]_{\text{V}}$. (4) is proved by the above statements.

Case $\dfrac{\Delta; \Gamma; \rho_1 \vdash_{Z_1} e_1 : \tau_1; \xrightarrow{q;Z} \tau_2 \ (\star) \qquad \Delta; \Gamma; \rho_2 \vdash_{Z_2} e_2 : \tau_1 \ (\diamond) \quad Z' = Z_1 + \max(Z, Z_2 + q) \qquad \rho' = \max(\rho_1, Z_1 + \rho_2)}{\Delta; \Gamma; \rho' \vdash_{Z'} e_1\ e_2 : \tau_2}$ **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. $(\mathtt{adap}(T) \leq \sigma Z')$;
2. $(\forall x \in \mathrm{Vars}.\mathtt{depth}_x(T) \leq \rho'(x))$;
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) $(\mathtt{adap}(T_1) \leq \sigma Z_1)$;
(3) $(\forall x \in \mathrm{Vars}.\mathtt{depth}_x(T_1) \leq \rho_1(x))$;
(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 $v_1 = (\mathtt{fix}\, f(x).e, \theta')$ (b) s.t. $T_1 = (\mathtt{fix}\, f(x).e, \theta)$
By inversion on (4), we know: $\forall j' < (k - j_1) \wedge (j', v') \in [\![\sigma\tau_1]\!]_{\mathrm{V}}$,
$(\theta'[x \mapsto v', f \mapsto (\mathtt{fix}\, f(x).e, \theta')], e) \Downarrow v'', T''$ $(d) \wedge |T''| = j'' \leq j' \wedge \mathtt{adap}(T') \leq \sigma Z \wedge \mathtt{depth}_x(T') \leq \sigma q \wedge (j' - j'', v'') \in [\![\sigma\tau_2]\!]_{\mathrm{V}}$ (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). $(\mathtt{adap}(T_2) \leq \sigma Z_2)$;
(8). $(\forall x \in \mathrm{Vars}.\mathtt{depth}_x(T_2) \leq \rho_2(x))$;
(9). $((k - j_2), v_2) \in [\![\sigma\tau_1]\!]_{\mathrm{V}}$
Apply Lemma 13 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). $\mathtt{adap}(T'') \leq \sigma Z$;
(12). $\mathtt{depth}_x(T'') \leq q$;
(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:

$$\frac{\theta, \sigma e_1 \Downarrow v_1, T_1\ (a) \qquad v_1 = (\mathtt{fix}\, f(x).e, \theta')\ (b) \qquad \theta, \sigma e_2 \Downarrow v_2, T_2\ (c) \qquad \theta'[f \mapsto v_1, x \mapsto v_2], e \Downarrow v'', T''\ (d)}{\theta, \sigma(e_1\ e_2) \Downarrow v'', T_1\ T_2 \triangleright \mathtt{fix}\, f(x).T''}$$

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

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

Case

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

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

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

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

STS1: $\texttt{adap}(T) = \texttt{adap}(\delta(T')) \leq \sigma Z'$
Unfold $\texttt{adap}(\delta(T'))$, STS:
$1 + \texttt{adap}(T') + \mathsf{MAX}_{v\in\tau_1}\Big(\max\big(\texttt{adap}(T_3(v)), \texttt{depth}_x(T_3(v))\big)\Big) \leq \sigma Z'$.
where $v_1 = (\texttt{fix}\, f(x : \tau_1).e_1, \theta_1) = \mathsf{extract}(T')$ and $\theta_1[f \mapsto v_1, x \mapsto v], e_1 \Downarrow v'', T_3(v)$
By Lemma5 based on our assumption $\sigma\theta, \sigma e \Downarrow v', T'$, we know that $v_1 = \mathsf{extract}(T) = v'$.
Unfold (c), we know : $(k-j', (\texttt{fix}\, f(x : \tau_1).e_1, \theta_1)) \in [\![(\tau_1; 0 \xrightarrow{\rho'';0} \tau_2)]\!]_{\text{V}}$ (d) and $\delta \notin (\texttt{fix}\, f(x:\tau).e_1, \theta_1)$
Unfold (d), we get : $\forall j_1 < (k-j').(j_1, v_a) \in [\![\tau_1]\!]_{\text{V}}$, $(j_1, (e_1, \theta_1[f \mapsto v_1, x \mapsto v_a])) \in [\![\tau_2]\!]_{\text{E}}^{\rho''[x:0,f:\infty],0}$ (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: $\texttt{adap}(T_3(v_a)) \leq 0(f)$

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

From (f), we know $\forall v_a \in [\![\tau_1]\!]_{\mathrm{V}}.\mathtt{adap}(T_3(v_a)) = 0$.
By Lemma 10, 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: $\mathtt{depth}_x(T) = \mathtt{depth}_x(\delta(T')) = 1+\max(\mathtt{depth}_x(T'), \mathtt{adap}(T')+\mathsf{MAX}_{v\in\tau_1}\Big(\max(\mathtt{depth}_x(T_3(v)), \bot)$
$\leq \rho'(x)$.
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.

□

References

[1] Ezgi Cicek, Deepak Garg, and Umut Acar. Refinement types for incremental computational complexity. In *Proc. ESOP*, 2015.

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